



Video 3.1

React.js: Introduction

Chris Murphy

Review

- **JavaScript:** a general-purpose, easy-to-use programming language
- **DOM:** representation of structure of HTML page, which can be manipulated using JavaScript
- **jQuery:** library that simplifies accessing/using the DOM

What is React?

- JavaScript library for building user interfaces
- HTML page is composed of recyclable, interactive **'components'** that have a lifecycle during which the state of the component changes
- Highly efficient because of notion of **VirtualDOM**
- Created and maintained by Facebook
- Used in production by many well known companies

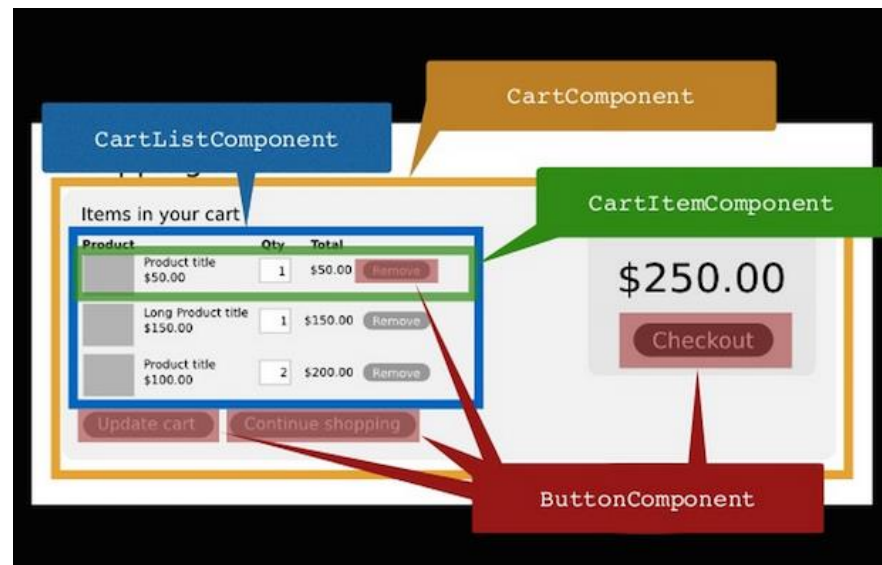
- Netflix
- WhatsApp, Instagram
- Atlassian (BitBucket, HipChat, Jira)
- Codecademy
- Airbnb
- Pinterest
- Dropbox
- PayPal
- Reddit
- Salesforce
- Squarespace
- New York Times
- Treehouse
- eBay
- Trulia
- Expedia
- Visa
- Wolfram Alpha

Why React?

- **Modularity:** organize code into reusable components that can work together
- **Lifecycle maintenance:** modifying component based on state; event listeners; simplified conditional rendering
- **JSX:** write HTML within JavaScript

Components

- Building blocks of React
- Make up the nodes included in the VirtualDOM
- Include and maintain a **state** that changes with events
- Each component maintains state independently
- Applications can be configured to respond to component level events



VirtualDOM

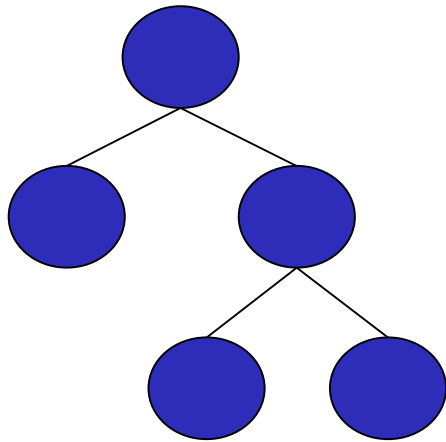
- Node tree that represents HTML elements, their attributes, and content as objects and properties
- **Selectively** renders and re-renders **subtrees** of nodes based on state changes
- Efficient because it does the least amount of DOM manipulation to update components
- Provides a layer of abstraction to the developer, providing simpler programming model and high performance

Normal DOM – How it Works

- When a node is updated, the browser updates (re-renders) **all** nodes

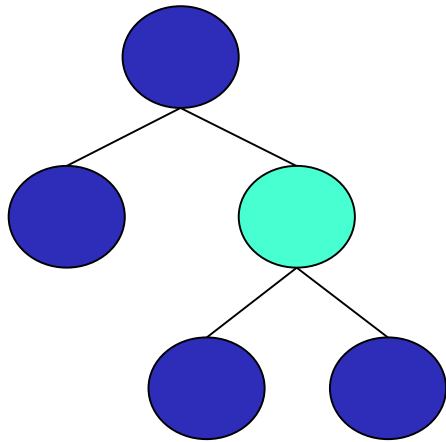
Normal DOM – How it Works

- When a node is updated, the browser updates (re-renders) **all** nodes



Normal DOM – How it Works

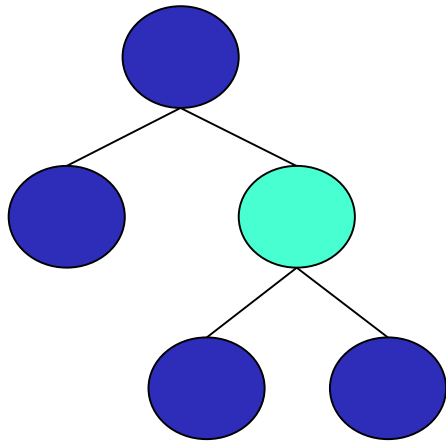
- When a node is updated, the browser updates (re-renders) **all** nodes



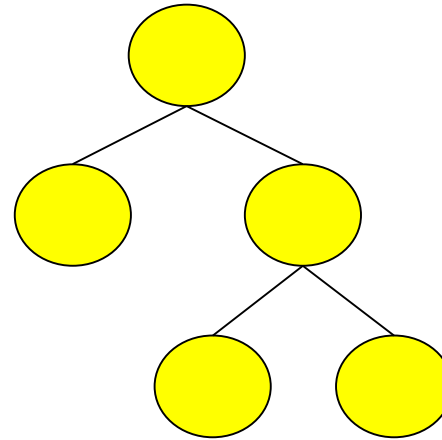
*Change has been
made to any given
node*

Normal DOM – How it Works

- When a node is updated, the browser updates (re-renders) **all** nodes



Change has been made to any given node



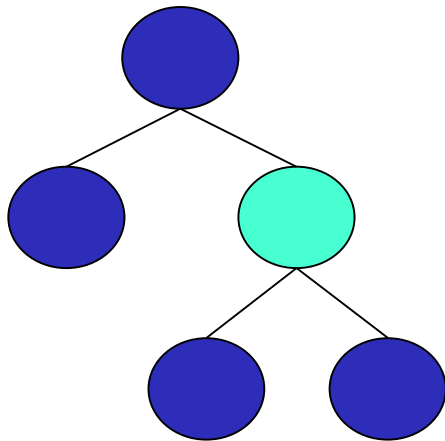
*Re-render **all** nodes to reflect the change*

VirtualDOM – How it Works

- When a node is updated, two things occur:
 - **'diff'** to determine which nodes within DOM have changed
 - **'reconciliation'** to update the nodes that are affected

VirtualDOM – How it Works

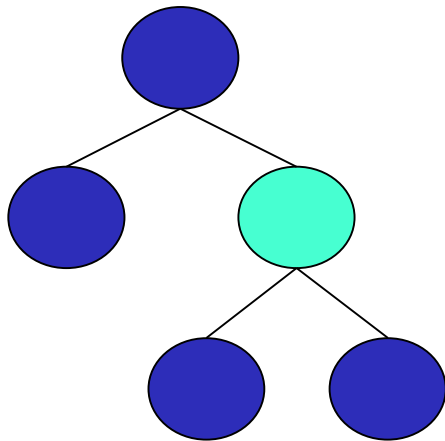
- When a node is updated, two things occur:
 - **'diff'** to determine which nodes within DOM have changed
 - **'reconciliation'** to update the nodes that are affected



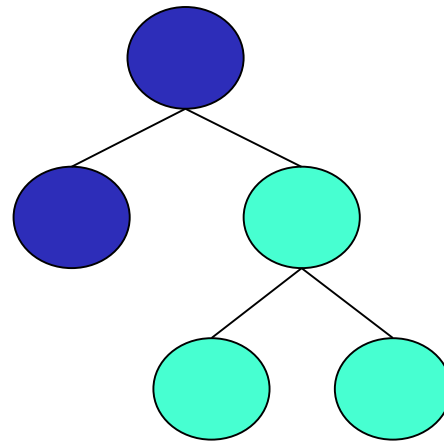
*Identify nodes that
have changed
(**'diff'**)*

VirtualDOM – How it Works

- When a node is updated, two things occur:
 - **'diff'** to determine which nodes within DOM have changed
 - **'reconciliation'** to update the nodes that are affected



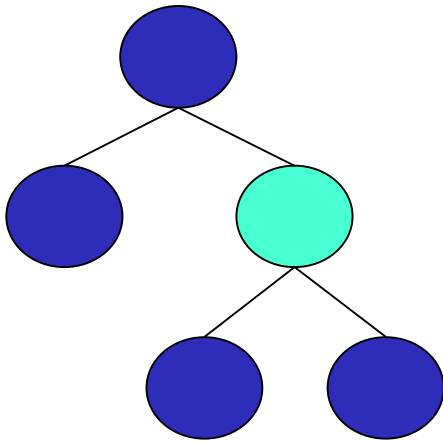
*Identify nodes that
have changed
(**'diff'**)*



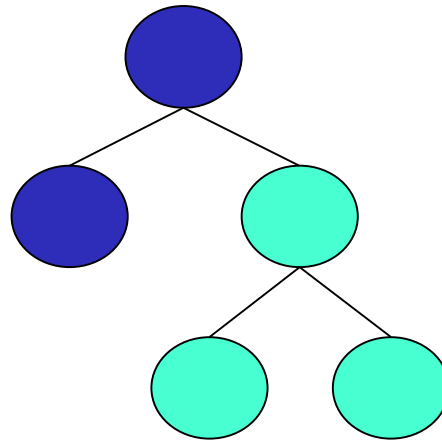
*Identify nodes that are
affected by the
change
(**'reconciliation'**)*

VirtualDOM – How it Works

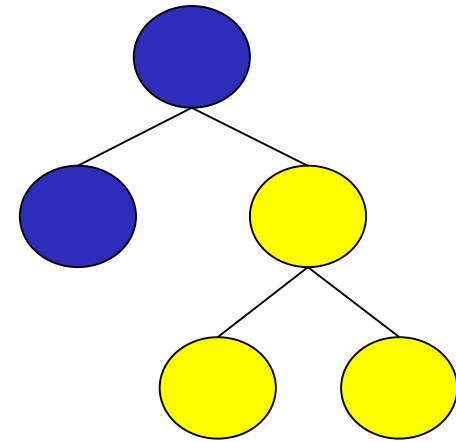
- When a node is updated, two things occur:
 - **'diff'** to determine which nodes within DOM have changed
 - **'reconciliation'** to update the nodes that are affected



Identify nodes that have changed
(**'diff'**)



Identify nodes that are affected by the change
(**'reconciliation'**)



Re-render **ONLY** the nodes that were affected by change

Developing with React

1. Within the page's HTML, allocate a position on the page in which the desired React component will be rendered, e.g. a **div**
2. Create a React component in JavaScript
 - Establish an initial state
 - Define any events that could change the component's state over its lifecycle
 - Define the function to render the HTML
3. Drop the component into position allocated in Step 1

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```


Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

Getting Started

- Create a **div** in the HTML to represent the location where the React component will be placed
- Write JavaScript code to create and display component in **div**

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id="container"></div>
    <script type="text/jsx">
      <!-- Insert React code here -->
    </script>
  </body>
</html>
```

JSX

- JSX – JavaScript XML Syntax Transform
- Allows user to write HTML-like tags within JavaScript
- Converts text (HTML) to React code

Rendering Elements using JSX

```
<div id="container"></div>
<script type='text/jsx'>

  ReactDOM.render (
    <h1>Hello, World!</h1>,
    document.getElementById('container')
  );

</script>
```

Rendering Elements using JSX

```
<div id="container"></div>
```

```
<script type='text/jsx'>
```

```
  ReactDOM.render (
```

```
    <h1>Hello, World!</h1>,
```

```
    document.getElementById('container')
```

```
  );
```

```
</script>
```

Rendering Elements using JSX

```
<div id="container"></div>  
<script type='text/jsx'>
```

```
  ReactDOM.render (
```

```
    <h1>Hello, World!</h1>,  
    document.getElementById('container')
```

```
  );
```

```
</script>
```

Rendering Elements using JSX

```
<div id="container"></div>
<script type='text/jsx'>

  ReactDOM.render (
    <h1>Hello, World!</h1>,
    document.getElementById('container')
  );

</script>
```

Rendering Elements using JSX

```
<div id="container"></div>
<script type='text/jsx'>

  ReactDOM.render (
    <h1>Hello, World!</h1>,
    document.getElementById('container')
  );

</script>
```

Rendering Elements using JSX

```
<div id="container"></div>
<script type='text/jsx'>

  ReactDOM.render (
    <h1>Hello, World!</h1>,
    document.getElementById('container')
  );

</script>
```

Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render(
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```

Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render(
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```


Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render(
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```

Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render (
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```

Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render (
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```

Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render(
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```

Hello, React!

```
<!DOCTYPE html>
<html>
  <head>
    <title>ReactJS Example</title>
    <script src="react.js"></script>
    <script src="react-dom.js"></script>
  </head>
  <body>
    <div id='container'></div>
    <script type='text/jsx'>
      ReactDOM.render(
        <h1> Hello, React! </h1>,
        document.getElementById('container')
      );
    </script>
  </body>
</html>
```

Looking Ahead

- Defining React components
- Reacting to user events
- Interaction between React components
- Developing large applications with React

Looking Ahead

- Defining React components
- Reacting to user events
- Interaction between React components
- Developing large applications with React



Video 3.2
React Components
Chris Murphy

Review

- React allows us to create custom components and insert them into the VirtualDOM in our HTML page
- This allows for selective rendering and modular development of dynamic behavior

React Components

- Components are JavaScript objects based off the **React.Component** prototype
- Components define properties, event-based state variables, and callback functions
- A component's **render()** function is used to render its HTML
- VirtualDOM manages each component's lifecycle and calls its **render()** function as needed

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```


Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

- Once we create the custom component, we can render it in the same way as HTML elements

```
ReactDOM.render (
  <HelloComponent />,
  document.getElementById('container')
);
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

- Once we create the custom component, we can render it in the same way as HTML elements

```
ReactDOM.render (
  <HelloComponent />,
  document.getElementById('container')
);
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

- Once we create the custom component, we can render it in the same way as HTML elements

```
ReactDOM.render (
  <HelloComponent />,
  document.getElementById('container')
);
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

- Once we create the custom component, we can render it in the same way as HTML elements

```
ReactDOM.render (
  <HelloComponent />,
  document.getElementById('container')
);
```

Creating React Components

- **React.createClass()** allows us to define a component
- This function takes an object containing the component's specifications as an argument:

```
var HelloComponent = React.createClass ({
  render: function() {
    return (
      <h1>Hello, React!</h1>
    );
  }
});
```

- Once we create the custom component, we can render it in the same way as HTML elements

```
ReactDOM.render (
  <HelloComponent />,
  document.getElementById('container')
);
```

Creating Custom Components – ES6

- ES6 is a more recent version of JavaScript syntax
- We can define a **class** instead of a single object

```
class HelloComponent extends React.Component {  
  render() {  
    return (  
      <h1>Hello, React!</h1>  
    );  
  }  
}
```

```
ReactDOM.render(  
  <HelloComponent />,  
  document.getElementById('container')  
);
```


Creating Custom Components – ES6

- ES6 is a more recent version of JavaScript syntax
- We can define a **class** instead of a single object

```
class HelloComponent extends React.Component {  
  render() {  
    return (  
      <h1>Hello, React!</h1>  
    );  
  }  
}
```

```
ReactDOM.render(  
  <HelloComponent />,  
  document.getElementById('container')  
);
```

Creating Custom Components – ES6

- ES6 is a more recent version of JavaScript syntax
- We can define a **class** instead of a single object

```
class HelloComponent extends React.Component {  
  render() {  
    return (  
      <h1>Hello, React!</h1>  
    );  
  }  
}
```

```
ReactDOM.render(  
  <HelloComponent />,  
  document.getElementById('container')  
);
```

Creating Custom Components – ES6

- ES6 is a more recent version of JavaScript syntax
- We can define a **class** instead of a single object

```
class HelloComponent extends React.Component {  
  render() {  
    return (  
      <h1>Hello, React!</h1>  
    );  
  }  
}
```

```
ReactDOM.render(  
  <HelloComponent />,  
  document.getElementById('container')  
);
```

Creating Custom Components – ES6

- ES6 is a more recent version of JavaScript syntax
- We can define a **class** instead of a single object

```
class HelloComponent extends React.Component {  
  render() {  
    return (  
      <h1>Hello, React!</h1>  
    );  
  }  
}
```

```
ReactDOM.render(  
  <HelloComponent />,  
  document.getElementById('container')  
);
```

Creating Custom Components – ES6

- ES6 is a more recent version of JavaScript syntax
- We can define a **class** instead of a single object

```
class HelloComponent extends React.Component {  
  render() {  
    return (  
      <h1>Hello, React!</h1>  
    );  
  }  
}
```

```
ReactDOM.render(  
  <HelloComponent />,  
  document.getElementById('container')  
);
```

React Component Attributes

- **Properties**

- Attributes and values that are set when the component is created
- Should never be modified after initialization

React Component Attributes

- **Properties**

- Attributes and values that are set when the component is created
- Should never be modified after initialization

- **State**

- Attributes and values that represent the current state of the component, based on what it does/represents
- Can be modified during the component's lifecycle

React Component Attributes

- **Properties**

- Attributes and values that are set when the component is created
- Should never be modified after initialization

- **State**

- Attributes and values that represent the current state of the component, based on what it does/represents
- Can be modified during the component's lifecycle

- Both properties and state can be used when rendering the component

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render (  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```


Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name} !</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component Properties

- Should always be assigned upon object creation, never modified afterward
- Component accesses its properties through **this.props**

```
ReactDOM.render(  
  <HelloUser name="Maria" />,  
  document.getElementById('container')  
);
```

```
class HelloUser extends React.Component {  
  render() {  
    return (  
      <h1>Hello {this.props.name}!</h1>  
    );  
  }  
}
```

Component State

- The set of variables that can change during the component's lifecycle
- Should be initialized in the **constructor**
- Component accesses its state through **this.state**

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```



```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props) ;
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```



```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return {this.state.numViews};
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

```
class TimesViewed extends React.Component {
  constructor(props) {
    super(props);
    var timesViewed = 0;
    if (localStorage.timesViewed) {
      timesViewed = localStorage.timesViewed;
    }
    timesViewed++;
    this.state = { numViews: timesViewed };
    localStorage.timesViewed = timesViewed;
  }

  render() {
    return <b>{this.state.numViews}</b>;
  }
}
```

```
ReactDOM.render(
  <TimesViewed />,
  document.getElementById('container')
);
```

Component Lifecycle

- The React VirtualDOM invokes callback functions on components during their lifecycle
- These functions fall into three categories:
 - Mounting
 - Updating
 - Unmounting
- You can optionally implement these for controlling the component

Component Lifecycle: Mounting

- Called when a component is being created and added to the VirtualDOM
- **constructor**: creates component, initializes state based on properties
- **componentWillMount**: invoked before component is added to VirtualDOM
- **componentDidMount**: invoked after component has been added to VirtualDOM and has been rendered

Component Lifecycle: Updating

- Called when a component's props or state is changing and the component is re-rendered
- **componentWillReceiveProps:** invoked before receiving new props, e.g. when its parent component re-renders
- **shouldComponentUpdate:** can be used to determine whether to re-render
- **componentWillUpdate:** invoked before re-rendering after change to state
- **componentDidUpdate:** invoked after being re-rendered

Component Lifecycle: Unmounting

- Called when a component is being removed from the VirtualDOM
- **componentWillUnmount**: invoked before component is removed from VirtualDOM and destroyed

Summary

- React components are JavaScript objects that can be used as HTML elements in the VirtualDOM
- A component's **render ()** function is used to render its HTML
- A component's **properties** are assigned when it is created
- A component's **state** can change during its lifecycle
- A component's **lifecycle functions** are invoked depending on relevant activities



Video 3.3
React Events
Chris Murphy

Review

- React allows us to insert JavaScript elements/components into VirtualDOM
- We can create additional components using the **React.Component** class as a base
- Components have two types of attributes
 - **Properties:** set at initialization and immutable thereafter
 - **State:** change in response to user events
- Component callback functions can be bound to HTML events

Changing Component State

- A component's state typically changes in response to some user action or "event"
- We can **bind** an event to a callback function within a React component
- That component can then change state using its **setState** function
- This will automatically re-render the component and any other affected component



Button Click



Guest



react-click.html



Count: 0

Click Me!

Count: 1

Click Me!


```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
    </div>
    );
  }
};
```



```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
    </div>
    );
  }
};

```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

```

class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};

```



```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  incrementCount() { // callback function
    this.setState( { count: this.state.count + 1; } );
  }

  render() { // invoked when setState is called
    return (
      <div> Count: { this.state.count }
      <button type="button"
        onClick={ this.incrementCount.bind(this); } >
        Increment
      </button>
      </div>
    );
  }
};
```

Java 👍 Like

JavaScript 👍 Like



Java  Like



JavaScript  Like



Java  Unlike



JavaScript  Like



Java 👍 Unlike

JavaScript 👍 Like

Java 👍 Unlike

JavaScript 👍 Unlike

Java 👍 Unlike

JavaScript 👍 Unlike

Java 👍 Like

JavaScript 👍 Unlike

```
ReactDOM.render (  
  <div>  
    <LikeButton name="Java" />  
    <LikeButton name="JavaScript" />  
  </div>,  
  document.getElementById('container'));
```

ReactDOM.render (

```
<div>  
  <LikeButton name="Java" />  
  <LikeButton name="JavaScript" />  
</div>,  
document.getElementById('container')) ;
```

```
ReactDOM.render (  
  <div>  
    <LikeButton name="Java" />  
    <LikeButton name="JavaScript" />  
  </div>,  
  document.getElementById('container'));
```

```
ReactDOM.render (  
  <div>  
    <LikeButton name="Java" />  
    <LikeButton name="JavaScript" />  
  </div>,  
  document.getElementById('container'));
```

```
ReactDOM.render(  
  <div>  
    <LikeButton name="Java" />  
    <LikeButton name="JavaScript" />  
  </div>,  
  document.getElementById('container'));
```

```
ReactDOM.render (  
  <div>  
    <LikeButton name="Java" />  
    <LikeButton name="JavaScript" />  
  </div>,  
  document.getElementById('container'));
```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```
class LikeButton extends React.Component {
```

```
  constructor(props) {  
    super(props);  
    this.state = { liked : false };  
  }
```

```
  toggle() {  
    this.setState( {liked: !this.state.liked} );  
  }
```

```
  render() {  
    var name = this.props.name;  
    var txt = this.state.liked ? 'Unlike' : 'Like';  
    var myColor = this.state.liked ? 'red' : 'black';  
    var weight = this.state.liked ? 'bold' : 'normal';  
    return (  
      <p><span style={{color:myColor, fontWeight:weight }}>  
        {name} </span>  
        <span onClick={this.toggle.bind(this)}>  
          {'\ud83d\udc4d' + txt}  
        </span> </p> );  
  }
```

```
};
```




```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
    }
  };

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```
class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};
```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



```
class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};
```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
    }
  };

```




```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
    }
  };

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
    }
  };

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
    }
  };

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
    )
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={ {color:myColor, fontWeight:weight } }>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```

```

class LikeButton extends React.Component {
  constructor(props) {
    super(props);
    this.state = { liked : false };
  }

  toggle() {
    this.setState( {liked: !this.state.liked} );
  }

  render() {
    var name = this.props.name;
    var txt = this.state.liked ? 'Unlike' : 'Like';
    var myColor = this.state.liked ? 'red' : 'black';
    var weight = this.state.liked ? 'bold' : 'normal';
    return (
      <p><span style={{color:myColor, fontWeight:weight }}>
        {name} </span>
        <span onClick={this.toggle.bind(this)}>
          {'\ud83d\udc4d' + txt}
        </span> </p> );
  }
};

```



react-mouse.html

Look at me!



Look at me! 



Look at me!





react-mouse.html



Look at me!





react-mouse.html

Look at me!



```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```



```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```



```
class MyText extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { bold : false, color : 'black' };  
  }  
  
  handleMouseOver() {  
    this.setState( { bold : true } );  
  }  
  
  handleMouseOut() {  
    this.setState( { bold: false } );  
  }  
  
  handleClick() {  
    if (this.state.color == 'red' )  
      this.setState( { color: 'black' } );  
    else  
      this.setState( { color: 'red' } );  
  }  
  
  . . .  
}
```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```
class MyText extends React.Component {  
  
  . . .  
  
  render () {  
    var myColor = this.state.color;  
    var weight = this.state.bold ? 'bold' : 'normal' ;  
    return (  
      <span style={ {color:myColor, fontWeight:weight} }  
        onClick={this.handleClick.bind(this)}  
          onMouseOver={this.handleMouseOver.bind(this)}  
          onMouseOut={this.handleMouseOut.bind(this)} >  
        {this.props.text}  
      </span>  
    );  
  }  
};  
  
ReactDOM.render(  
  <div><MyText text="Look at me!" /></div>,  
  document.getElementById('container')  
) ;
```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```
class MyText extends React.Component {  
  
  . . .  
  
  render () {  
    var myColor = this.state.color;  
    var weight = this.state.bold ? 'bold' : 'normal' ;  
    return (  
      <span style={ {color:myColor, fontWeight:weight} }  
        onClick={this.handleClick.bind(this)}  
          onMouseOver={this.handleMouseOver.bind(this)}  
          onMouseOut={this.handleMouseOut.bind(this)} >  
        {this.props.text}  
      </span>  
    );  
  }  
};  
  
ReactDOM.render(  
  <div><MyText text="Look at me!" /></div>,  
  document.getElementById('container')  
) ;
```



```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
        onMouseOver={this.handleMouseOver.bind(this)}
        onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
            onMouseOut={this.handleMouseOut.bind(this)} >
        {this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
{this.props.text}
      </span>
    );
  }
};

ReactDOM.render(
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

```

class MyText extends React.Component {

  . . .

  render () {
    var myColor = this.state.color;
    var weight = this.state.bold ? 'bold' : 'normal' ;
    return (
      <span style={ {color:myColor, fontWeight:weight} }
        onClick={this.handleClick.bind(this)}
          onMouseOver={this.handleMouseOver.bind(this)}
          onMouseOut={this.handleMouseOut.bind(this)} >
{this.props.text}
      </span>
    );
  }
};

ReactDOM.render (
  <div><MyText text="Look at me!" /></div>,
  document.getElementById('container')
);

```

Summary

- We can bind user events in HTML elements to callback functions in React components
- When we invoke a component's **setState** function, the **render** function will automatically be called and the component's appearance can change accordingly



Video 3.4

React Component Interaction

Chris Murphy

Review

- We can bind user events in HTML elements to callback functions in React components
- When we invoke a component's **setState** function, the **render** function will automatically be called and the component's appearance can change accordingly
- Re-rendering one component may necessitate re-rendering others as well



- anteater
- bear
- cat
- dog
- elephant
- fox

← → ↻ 🔍 react-list.html

- anteater
- bear
- cat
- elephant

← → ↻ 🔍 react-list.html

- anteater
- elephant

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {
```

```
  constructor(props) {
```

```
    super(props);
```

```
    var allItems = [ "Anteater", "Bear", "Cat",  
                    "Dog", "Elephant" ];
```

```
    this.state = { initialItems: allItems,  
                  currentItems: allItems };
```

```
  }
```

```
  filterList(input) { // callback function
```

```
    var updatedList = this.state.initialItems;
```

```
    updatedList = updatedList.filter(function(item) {  
      return item.search(input.target.value) !== -1;  
    });
```

```
    this.setState( { currentItems: updatedList } );
```

```
  }
```

```
  . . .
```

```
class FilteredList extends React.Component {  
  
  constructor(props) {  
    super(props);  
    var allItems = [ "Anteater", "Bear", "Cat",  
                    "Dog", "Elephant" ];  
    this.state = { initialItems: allItems,  
                  currentItems: allItems };  
}  
  
  filterList(input) { // callback function  
    var updatedList = this.state.initialItems;  
  
    updatedList = updatedList.filter(function(item) {  
      return item.search(input.target.value) !== -1;  
    });  
  
    this.setState( { currentItems: updatedList } );  
  }  
  
  . . .
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                   "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                   currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```



```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```

class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .

```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```



```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```

class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .

```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```
class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .
}
```

```

class FilteredList extends React.Component {

  constructor(props) {
    super(props);
    var allItems = [ "Anteater", "Bear", "Cat",
                    "Dog", "Elephant" ];
    this.state = { initialItems: allItems,
                  currentItems: allItems };
  }

  filterList(input) { // callback function
    var updatedList = this.state.initialItems;

    updatedList = updatedList.filter(function(item) {
      return item.search(input.target.value) !== -1;
    });

    this.setState( { currentItems: updatedList } );
  }

  . . .

```

```
class FilteredList extends React.Component {  
  
  constructor(props) {  
    super(props);  
    var allItems = [ "Anteater", "Bear", "Cat",  
                    "Dog", "Elephant" ];  
    this.state = { initialItems: allItems,  
                  currentItems: allItems };  
  }  
  
  filterList(input) { // callback function  
    var updatedList = this.state.initialItems;  
  
    updatedList = updatedList.filter(function(item) {  
      return item.search(input.target.value) !== -1;  
    });  
  
    this.setState( { currentItems: updatedList } );  
  }  
  
  . . .  
}
```

```

class FilteredList extends React.Component {
    . . .
    render() {
        return (
            <div><input type="text" placeholder="Search"
                onChange={this.filterList.bind(this)} />
                <ListItems items={this.state.currentItems} />
            </div>
        );
    }
};

class ListItems extends React.Component {
    render() {
        return (
            <ul> { this.props.items.map(function(item) {
                return <li key={item}> {item} </li> } ) } </ul>
        );
    }
};

ReactDOM.render(<FilteredList />,
    document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render () {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
      </div>
    );
  }
};

class ListItems extends React.Component {
  render () {
    return (
      <ul> { this.props.items.map(function (item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```



```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
      </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
        <ListItems items={this.state.currentItems} />
      </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```



```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```

```

class FilteredList extends React.Component {
  . . .
  render() {
    return (
      <div><input type="text" placeholder="Search"
        onChange={this.filterList.bind(this)} />
      <ListItems items={this.state.currentItems} />
    </div>
    );
  }
};

class ListItems extends React.Component {
  render() {
    return (
      <ul> { this.props.items.map(function(item) {
        return <li key={item}> {item} </li> } ) } </ul>
    );
  }
};

ReactDOM.render(<FilteredList />,
  document.getElementById('container'));

```



```

class FilteredList extends React.Component {
    . . .
    render() {
        return (
            <div><input type="text" placeholder="Search"
                onChange={this.filterList.bind(this)} />
                <ListItems items={this.state.currentItems} />
            </div>
        );
    }
};

class ListItems extends React.Component {
    render() {
        return (
            <ul> { this.props.items.map(function(item) {
                return <li key={item}> {item} </li> } ) } </ul>
        );
    }
};

ReactDOM.render(<FilteredList />,
    document.getElementById('container'));

```



TO-DO LIST

TO-DO LIST

Add

TO-DO LIST

- learn JavaScript

TO-DO LIST

- learn JavaScript
- find happiness

TO-DO LIST

- ~~learn JavaScript~~
- find happiness

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```



```
class TodoApp extends React.Component {
  constructor(props) {
    super(props) ;
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = { items: [], text: ' ', id: 0 };
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```



```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                   id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                   id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem) ,
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```



```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = {items: [], text: ' ', id: 0};
  }

  handleChange(e) {
    this.setState( { text: e.target.value } );
  }

  handleSubmit(e) {
    e.preventDefault(); // so as not to reload the page
    var newItem = { text: this.state.text,
                    id: this.state.id };
    this.setState( {
      items: this.state.items.concat(newItem),
      text: ' ',
      id: this.state.id + 1
    });
  }
}
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
        <input onChange={this.handleChange.bind(this)}
          value={this.state.text} />
        <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```



```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
        <input onChange={this.handleChange.bind(this)}
          value={this.state.text} />
        <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .

  render() {
    return (
      <div>
        <h3>TO-DO LIST</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit.bind(this)}>
          <input onChange={this.handleChange.bind(this)}
            value={this.state.text} />
          <button>Add</button>
        </form>
      </div>
    );
  }
};
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {  
  . . .  
  render() {  
    return (  
      . . .  
      <TodoList items={this.state.items} />  
      . . .  
    )  
  }  
}
```

```
class TodoList extends React.Component {  
  render() {  
    return (  
      <ul>  
        {this.props.items.map(function (item) {  
          return  
            <TodoItem id={item.id} text={item.text}/>  
          })}  
      </ul>  
    );  
  }  
}
```

```
class TodoApp extends React.Component {  
  . . .  
  render() {  
    return (  
      . . .  
      <TodoList items={this.state.items} />  
      . . .  
    );  
  }  
}
```

```
class TodoList extends React.Component {  
  render() {  
    return (  
      <ul>  
        {this.props.items.map(function (item) {  
          return  
            <TodoItem id={item.id} text={item.text}/>  
          })}  
      </ul>  
    );  
  }  
}
```

```
class TodoApp extends React.Component {  
  . . .  
  render() {  
    return (  
      . . .  
      <TodoList items={this.state.items} />  
      . . .  
    );  
  }  
}
```

```
class TodoList extends React.Component {  
  render() {  
    return (  
      <ul>  
        {this.props.items.map(function (item) {  
          return  
            <TodoItem id={item.id} text={item.text}/>  
          })}  
      </ul>  
    );  
  }  
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    )
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        })}
      </ul>
    );
  }
}
```



```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        { this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        }) }
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
          })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
          })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text}/>
        })}
      </ul>
    );
  }
}
```

```
class TodoApp extends React.Component {
  . . .
  render() {
    return (
      . . .
      <TodoList items={this.state.items} />
      . . .
    );
  }
}
```

```
class TodoList extends React.Component {
  render() {
    return (
      <ul>
        {this.props.items.map(function (item) {
          return
            <TodoItem id={item.id} text={item.text} />
        })}
      </ul>
    );
  }
}
```



```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}
```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
    );
  }
}
```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props) ;
    this.state = { amDone: false };
}

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
    );
  }
}
```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}
```

```

class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}

```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
    );
  }
}
```

```

class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}

```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}
```



```

class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
    );
  }
}

```

```

class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}

```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
  }
}
```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
      );
    }
  }
}
```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration: line}}>
        {this.props.text} </li>
      );
  }
}
```

```
class TodoItem extends React.Component {
  constructor(props) {
    super(props);
    this.state = { amDone: false };
  }

  handleClick() {
    this.setState( { amDone: !this.state.amDone} );
  }

  render() {
    var line = this.state.amDone ? 'line-through' : 'none';
    return (
      <li key={this.props.id}
        onClick={this.handleClick.bind(this)}
        style={{textDecoration:line}}>
        {this.props.text} </li>
    );
  }
}
```

Review

- React allows us to create reusable, modularized components that can be combined to form web applications
- React handles re-rendering of components based on the structure of VirtualDOM



Video 3.5

React Component Interaction – part 2

Chris Murphy

Review

- React allows us to create reusable, modularized components that can be combined to form web applications
- React handles re-rendering of components based on the structure of VirtualDOM

Enter two numbers to multiply:

The product is 0.

Enter two numbers to multiply:

The product is 0.

Enter two numbers to multiply:

The product is 72.

Enter two numbers to multiply:

The product is 752.

Enter two numbers to multiply:

The product is 7802.

Enter two numbers to multiply:

83.6 94

The product is 7858.4.

react-multiply.html

Enter two numbers to multiply:

The product is 7858.4.

react-multiply.html

Enter two numbers to multiply:

The product is 7858.4.

Enter two numbers to multiply:

The product is 7858.4.

react-multiply.html

Enter two numbers to multiply:

83.6

94

The product is 7858.4.

react-multiply.html

Enter two numbers to multiply:

The product is 7858.4.

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {
```

```
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }
```

```
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props) ;  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

```
class Multiplier extends React.Component {

  constructor(props) {
    super(props);
    this.state = { input1: 0, input2: 0, product: 0 };
    this.multiply = this.multiply.bind(this);
  }

  multiply(id, val) {
    if (id == 1) {
      this.setState( { input1: val,
                       product: val * this.state.input2 } );
    }
    else if (id == 2) {
      this.setState( { input2: val,
                       product: this.state.input1 * val } );
    }
  }
}
```

...


```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
  
}
```

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {

  constructor(props) {
    super(props);
    this.state = { input1: 0, input2: 0, product: 0 };
    this.multiply = this.multiply.bind(this);
  }

  multiply(id, val) {
    if (id == 1) {
      this.setState( { input1: val,
                       product: val * this.state.input2 } );
    }
    else if (id == 2) {
      this.setState( { input2: val,
                       product: this.state.input1 * val } );
    }
  }
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...


```
class Multiplier extends React.Component {

  constructor(props) {
    super(props);
    this.state = { input1: 0, input2: 0, product: 0 };
    this.multiply = this.multiply.bind(this);
  }

  multiply(id, val) {
    if (id == 1) {
      this.setState( { input1: val,
                       product: val * this.state.input2 } );
    }
    else if (id == 2) {
      this.setState( { input2: val,
                       product: this.state.input1 * val } );
    }
  }
}
```

...

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
  
  ...  
}
```

```
class Multiplier extends React.Component {  
  
  constructor(props) {  
    super(props);  
    this.state = { input1: 0, input2: 0, product: 0 };  
    this.multiply = this.multiply.bind(this);  
  }  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}
```

...

```
class Multiplier extends React.Component {

  constructor(props) {
    super(props);
    this.state = { input1: 0, input2: 0, product: 0 };
    this.multiply = this.multiply.bind(this);
  }

  multiply(id, val) {
    if (id == 1) {
      this.setState( { input1: val,
                       product: val * this.state.input2 } );
    }
    else if (id == 2) {
      this.setState( { input2: val,
                       product: this.state.input1 * val } );
    }
  }
}
```

...

```
class Multiplier extends React.Component {

  constructor(props) {
    super(props);
    this.state = { input1: 0, input2: 0, product: 0 };
    this.multiply = this.multiply.bind(this);
  }

  multiply(id, val) {
    if (id == 1) {
      this.setState( { input1: val,
                       product: val * this.state.input2 } );
    }
    else if (id == 2) {
      this.setState( { input2: val,
                       product: this.state.input1 * val } );
    }
  }
}
```

...

```
class Multiplier extends React.Component {  
  
    . . .  
  
    render() {  
        return (  
            <div>  
                <NumberInputField id="1" action={this.multiply}/>  
                <NumberInputField id="2" action={this.multiply}/>  
                <OutputField product={this.state.product}/>  
            </div>  
        );  
    }  
}
```

```
class Multiplier extends React.Component {  
  
    . . .  
  
    render() {  
        return (  
            <div>  
                <NumberInputField id="1" action={this.multiply}/>  
                <NumberInputField id="2" action={this.multiply}/>  
                <OutputField product={this.state.product}/>  
            </div>  
        );  
    }  
}
```

```
class Multiplier extends React.Component {  
  
    . . .  
  
    render() {  
        return (  
            <div>  
                <NumberInputField id="1" action={this.multiply}/>  
                <NumberInputField id="2" action={this.multiply}/>  
                <OutputField product={this.state.product}/>  
            </div>  
        );  
    }  
}
```



```
class Multiplier extends React.Component {  
  
    . . .  
  
    render() {  
        return (  
            <div>  
                <NumberInputField id="1" action={this.multiply}/>  
                <NumberInputField id="2" action={this.multiply}/>  
                <OutputField product={this.state.product}/>  
            </div>  
        );  
    }  
}
```

```
class Multiplier extends React.Component {  
  
    . . .  
  
    render() {  
        return (  
            <div>  
                <NumberInputField id="1" action={this.multiply}/>  
                <NumberInputField id="2" action={this.multiply}/>  
                <OutputField product={this.state.product}/>  
            </div>  
        );  
    }  
}
```

```
class Multiplier extends React.Component {  
  
  . . .  
  
  render() {  
    return (  
      <div>  
        <NumberInputField id="1" action={this.multiply}/>  
        <NumberInputField id="2" action={this.multiply}/>  
        <OutputField product={this.state.product}/>  
      </div>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {  
  
    . . .  
  
    render() {  
        return (  
            <div>  
                <NumberInputField id="1" action={this.multiply}/>  
                <NumberInputField id="2" action={this.multiply}/>  
                <OutputField product={this.state.product}/>  
            </div>  
        );  
    }  
}
```

```
class Multiplier extends React.Component {
  . . .
  <NumberInputField id="1" action={this.multiply}/>
  . . .
}
```

```
class NumberInputField extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
  }

  handleChange(e) {
    this.props.action(this.props.id, e.target.value);
  }

  render() {
    return (
      <input onChange={this.handleChange}></input>
    );
  }
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```



```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {
  . . .
  <NumberInputField id="1" action={this.multiply}/>
  . . .
}
```

```
class NumberInputField extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
  }
```

```
handleChange(e) {
```

```
  this.props.action(this.props.id, e.target.value);
```

```
}
```

```
render() {
```

```
  return(
```

```
    <input onChange={this.handleChange}></input>
```

```
  );
```

```
}
```

```
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return(  
      <input onChange={this.handleChange}></input>  
    );  
  }  
}
```

```
class Multiplier extends React.Component {
  . . .
  <NumberInputField id="1" action={this.multiply}/>
  . . .
}
```

```
class NumberInputField extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
  }

  handleChange(e) {
    this.props.action(this.props.id, e.target.value);
  }

  render() {
    return (
      <input onChange={this.handleChange}></input>
    );
  }
}
```

```
class Multiplier extends React.Component {
  . . .
  <NumberInputField id="1" action={this.multiply}/>
  . . .
}
```

```
class NumberInputField extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
  }

  handleChange(e) {
    this.props.action(this.props.id, e.target.value);
  }

  render() {
    return (
      <input onChange={this.handleChange}></input>
    );
  }
}
```

```
class Multiplier extends React.Component {  
  . . .  
  <NumberInputField id="1" action={this.multiply}/>  
  . . .  
}
```

```
class NumberInputField extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
  }  
  
  handleChange(e) {  
    this.props.action(this.props.id, e.target.value);  
  }  
  
  render() {  
    return (  
      <input onChange={this.handleChange} ></input>  
    );  
  }  
}
```



```
class Multiplier extends React.Component {
  . . .
  <NumberInputField id="1" action={this.multiply}/>
  . . .
}
```

```
class NumberInputField extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
  }

  handleChange(e) {
    this.props.action(this.props.id, e.target.value);
  }

  render() {
    return (
      <input onChange={this.handleChange}></input>
    );
  }
}
```

```
class Multiplier extends React.Component {  
  
  . . .  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}  
  
. . .
```

```
class Multiplier extends React.Component {  
  
  . . .  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
  
  . . .  
}
```

```
class Multiplier extends React.Component {  
  
  . . .  
  
  multiply(id, val) {  
    if (id == 1) {  
      this.setState( { input1: val,  
                      product: val * this.state.input2 } );  
    }  
    else if (id == 2) {  
      this.setState( { input2: val,  
                      product: this.state.input1 * val } );  
    }  
  }  
}  
  
. . .
```

```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .  
}
```

```
class OutputField extends React.Component {  
  render() {  
    return(  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```

```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .
```

```
class OutputField extends React.Component {  
  render() {  
    return(  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```

```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .
```

```
class OutputField extends React.Component {  
  render() {  
    return(  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```

```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .  
}
```

```
class OutputField extends React.Component {  
  render() {  
    return(  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```



```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .  
}
```

```
class OutputField extends React.Component {  
  render() {  
    return (  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```

```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .  
}
```

```
class OutputField extends React.Component {  
  render() {  
    return (  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```

```
class Multiplier extends React.Component {  
  . . .  
  <OutputField product={this.state.product}/>  
  . . .  
}
```

```
class OutputField extends React.Component {  
  render() {  
    return(  
      <div>The product is {this.props.product}.  
      </div>  
    );  
  }  
}
```

```
ReactDOM.render(<Multiplier/>,  
  document.getElementById('container'));
```

```
class Multiplier extends React.Component {
  . . .
  <OutputField product={this.state.product}/>
  . . .
}
```

```
class OutputField extends React.Component {
  render() {
    return (
      <div>The product is {this.props.product}.
      </div>
    );
  }
}
```

```
ReactDOM.render(<Multiplier/>,
  document.getElementById('container'));
```

Review

- React allows us to create reusable, modularized components that can be combined to form web applications
- Components can communicate with each other via callback methods that are set as props



Video 3.6
React and APIs
Chris Murphy

Review

- React allows us to create modular JavaScript components that we can use in our HTML pages to develop web applications
- How can we take advantage of the modular nature of the Web?

Web development in the old days

```
query.c — code
Add License

19
20 main(int argc, char *argv[]) {
21     entry entries[10000];
22     register int x,m=0;
23     char *cl;
24
25     printf("Content-type: text/html%c%c",10,10);
26
27     if(strcmp(getenv("REQUEST_METHOD"),"GET")) {
28         printf("This script should be referenced with a METHOD of GET.\n");
29         printf("If you don't understand this, see this ");
30         printf("<A
. HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html\">forms
. overview</A>.%c",10);
31         exit(1);
32     }
33
34     cl = getenv("QUERY_STRING");
35     if(cl == NULL) {
36         printf("No query information to decode.\n");
37         exit(1);
38     }
39     for(x=0;cl[x] != '\0';x++) {
40         m=x;
41         getword(entries[x].val,cl,'&');
42         plustospace(entries[x].val);
43         unescape_url(entries[x].val);
44         getword(entries[x].name,entries[x].val,'=');
45     }
46
47     printf("<H1>Query Results</H1>");
48     printf("You submitted the following name/value pairs:<p>%c",10);
49     printf("<ul>%c",10);
50
51     for(x=0; x <= m; x++)
52         printf("<li> <code>%s = %s</code>%c",entries[x].name,
53             entries[x].val,10);
54     printf("</ul>%c",10);
55 }
```


Designing software for the Web

- How can we make this better?
- Use design paradigms with the following goals:
 - Break up code into distinct components
 - The components interact with each other in a “standard” manner
 - Makes it easy to change any component
 - Less dependency and coupling

Service-Oriented Architecture

- Letter by Jeff Bezos to all employees at Amazon (circa 2002)
- “All teams will henceforth expose their data and functionality through service interfaces.”
- “The only communication allowed is via service interface calls over the network.”
- “All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.”

Software as a Service (SaaS)

- Extends the idea of Service-oriented Architecture
- SaaS – entire software runs as a service
- Examples
 - Google Docs (as opposed to Microsoft Word)
 - Gmail (as opposed to email clients like Outlook)
 - Cloud9 (as opposed to Eclipse)

Software as a Service: Advantages

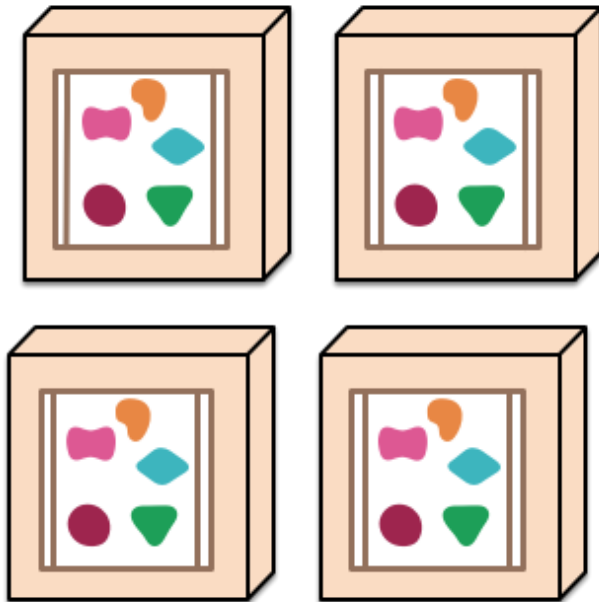
- No user installation is needed
- Less likely to lose data
- Users can collaborate with each other
- Upgrading software and datasets is easy
- Software is centralized in a single environment (don't need to worry about versions of operating systems, etc.)
- User's computer (hardware and software specifications) don't matter – usually all that's needed is a web browser

Microservices

A monolithic application puts all its functionality into a single process...



... and scales by replicating the monolith on multiple servers

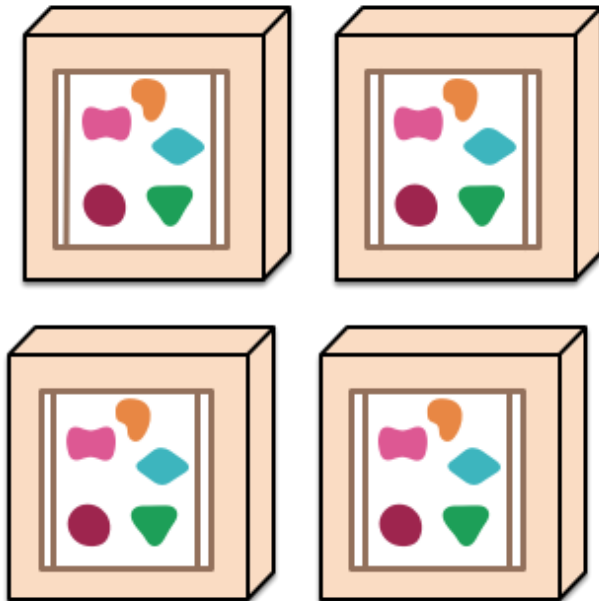


Microservices

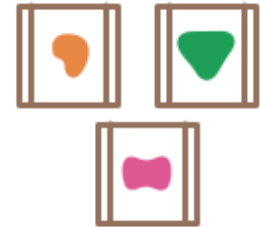
A monolithic application puts all its functionality into a single process...



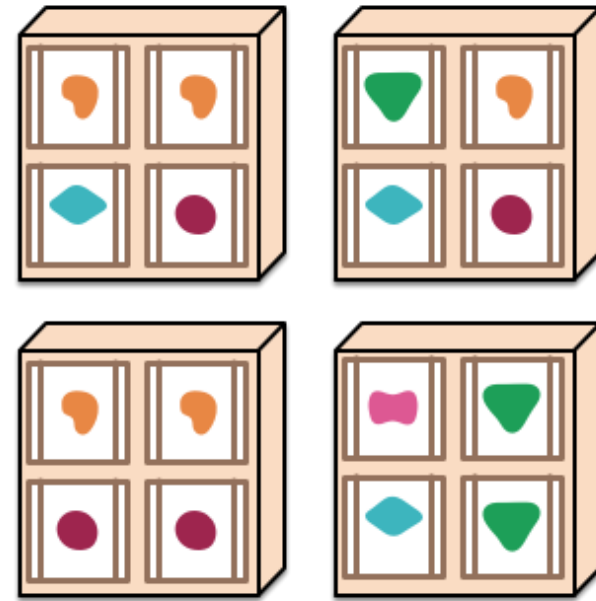
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



REST

- How do well-designed web servers behave?
- Define a set of rules and conventions
- **REST:** REpresentational State Transfer
- Collection of *resources* on which specific operations can be performed
- URI names *resources*, not pages or actions
- Self-contained - which resource, what to do with it, server doesn't need to maintain state between requests

What is an API?

- An API is an **Application Programming Interface**
- In web programming, an API is a URL or a set of URLs that returns pure data to requests
- APIs can be used to incorporate data and functionality from other sources in your webapp

The New York Times

- *The New York Times* is an American daily newspaper company
- *The New York Times* provides a variety of APIs that provide article data, books data, movies data, and more
- You can see more about available APIs and request access at <https://developer.nytimes.com/>

Our Goal

- Created a webapp that creates a dashboard of clickable *New York Times* article images



<https://www.nytimes.com/2017/03/23/us/taxas-aandm-speaking-policy-richard-spencer.html>

Getting Started

- Request a developer key for the Article Search API at <https://developer.nytimes.com/>
- You can test out the API and what the returned data looks like at https://developer.nytimes.com/article_search_v2.json

Structure

- First, let's create a new React component named ArticlesGrid to store the data in the page

```
class ArticlesGrid extends React.Component {  
  constructor (props) {  
    super(props);  
    this.state = {  
      articles: []  
    };  
  }  
  . . .
```

Structure

- First, let's create a new React component named ArticlesGrid to store the data in the page

```
class ArticlesGrid extends React.Component {  
  constructor (props) {  
    super(props);  
    this.state = {  
      articles: []  
    };  
  }  
  . . .  
}
```

Structure

- First, let's create a new React component named ArticlesGrid to store the data in the page

```
class ArticlesGrid extends React.Component {  
  constructor (props) {  
    super (props) ;  
    this.state = {  
      articles: []  
    };  
  }  
  . . .  
}
```

Structure

- First, let's create a new React component named ArticlesGrid to store the data in the page

```
class ArticlesGrid extends React.Component {  
  constructor (props) {  
    super(props);  
    this.state = {  
      articles: []  
    };  
  }  
  
  . . .
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }.bind(this));  
  }  
  
  . . .  
}
```


Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }.bind(this));  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }.bind(this));  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }).bind(this);  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {
  . . .

  componentDidMount () {
    var url=
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';

    $.getJSON(url, function(data, status) {
      return this.setState({articles: this.parse(data)});
    }).bind(this) );
  }

  . . .
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }).bind(this);  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {
  . . .

  componentDidMount () {
    var url=
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';

    $.getJSON(url, function(data, status) {
      return this.setState({articles: this.parse(data)});
    }).bind(this);
  }

  . . .
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }).bind(this);  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {
  . . .

  componentDidMount () {
    var url=
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';

    $.getJSON(url, function(data, status) {
      return this.setState ({articles: this.parse(data)});
    }).bind(this);
  }

  . . .
}
```


Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }.bind(this));  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    }.bind(this));  
  }  
  
  . . .  
}
```

Getting Data

- Let's start by getting some data from the API with a search term.
- We will do this with jQuery AJAX

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  componentDidMount () {  
    var url=  
      'http://api.nytimes.com/svc/search/v2/articlesearch.json?'  
      + 'api-key=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx';  
  
    $.getJSON(url, function(data, status) {  
      return this.setState({articles: this.parse(data)});  
    } .bind(this));  
  }  
  
  . . .  
}
```

```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        . . .
      }
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "At Washington last week the spokesman...
```

```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        . . .
      }
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "At Washington last week the spokesman...
```

```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        . . .
      }
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "At Washington last week the spokesman...
```

```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        . . .
      }
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "At Washington last week the spokesman...
```

```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        . . .
      }
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "At Washington last week the spokesman...
```



```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        ". . ."
      }
    ]
  }
}
```

```
{
  "response": {
    "meta": {
      "hits": 4251,
      "time": 36,
      "offset": 0
    },
    "docs": [
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "Having invited Senator LA FOLLETTE to make a
statement of his cause and case in today's TIMES, we would
refer to it only with becoming courtesy. Its great lack must
be obvious to every reader. This is its extreme indefiniteness.
The Senator uses w...",
        "lead_paragraph": "Having invited Senator LA FOLLETTE
to make a statement of his cause and case in today's TIMES,
we would refer to it only with becoming courtesy. Its great
lack must be obvious to every reader. This is its extreme
indefiniteness. The Senator uses with vehemence a great many
terms without defining one of them.",
        . . .
      }
      {
        "web_url": "https://query.nytimes.com/gst/...",
        "snippet": "At Washington last week the spokesman..."
      }
    ]
  }
}
```

Parsing Data

- The *NY Times* API will return a lot of raw data!
- Let's parse the data into something we can use
- For each article, we'll keep the image URL, title, and URL to original article if a large image exists

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```



```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }
}

isXL (image) {
  return image.subtype === 'xlarge';
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```



```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

```

class ArticlesGrid extends React.Component {
  . . .

  parse(results) {
    if( !results || !results.response ) return [];
    var articles = results.response.docs;
    var parsedArticles = [];
    for (var i = 0; i < articles.length; i++) {
      var article = articles[i];
      if (article.multimedia.find(this.isXL)) {
        parsedArticles.push({
          id:      article._id,
          title:   article.headline.main || 'Untitled',
          imageURL: article.multimedia.find(this.isXL).url || '#',
          webURL:  article.web_url || '#'
        });
      }
    }
    return parsedArticles;
  }

  isXL (image) {
    return image.subtype === 'xlarge';
  }
}

```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {  
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;  
  return (  
    <div className='article'>  
      <a className='article-link' href={article.webURL}>  
        <img className='article-image'  
          title={article.title}  
          src={imgURL} />  
      </a>  
    </div>  
  );  
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {  
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;  
  return (  
    <div className='article'>  
      <a className='article-link' href={article.webURL}>  
        <img className='article-image'  
          title={article.title}  
          src={imgURL} />  
      </a>  
    </div>  
  );  
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {  
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;  
  return (  
    <div className='article'>  
      <a className='article-link' href={article.webURL}>  
        <img className='article-image'  
          title={article.title}  
          src={imgURL} />  
      </a>  
    </div>  
  );  
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {  
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;  
  return (  
    <div className='article'>  
      <a className='article-link' href={article.webURL}>  
        <img className='article-image'  
          title={article.title}  
          src={imgURL} />  
      </a>  
    </div>  
  );  
}
```


Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```

Displaying Data

- Now that we have the data, let's make a component to display each article!
- This is shorthand for a component that only has a “render” function

```
var Article = function({article}) {
  var imgURL = 'https://static01.nyt.com/' + article.imageURL;
  return (
    <div className='article'>
      <a className='article-link' href={article.webURL}>
        <img className='article-image'
          title={article.title}
          src={imgURL} />
      </a>
    </div>
  );
}
```


Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  render () {  
    return this.state.articles && (  
      <div className='articles'>  
        { this.state.articles.map( function (article) {  
          return <Article article={article} key={article._id} />;  
        }) }  
      </div>  
    );  
  }  
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {  
  . . .  
  
  render () {  
    return this.state.articles && (  
      <div className='articles'>  
        { this.state.articles.map( function (article) {  
          return <Article article={article} key={article._id} />;  
        }) }  
      </div>  
    );  
  }  
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        }) }
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        }) }
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        })}
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        }) }
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        }})
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        })}
      </div>
    );
  }
}
```


Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        }) }
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        })}
      </div>
    );
  }
}
```

Displaying Data

- Finally, we can render the Articles Grid to tie it all together.

```
class ArticlesGrid extends React.Component {
  . . .

  render () {
    return this.state.articles && (
      <div className='articles'>
        { this.state.articles.map( function (article) {
          return <Article article={article} key={article._id} />;
        })}
      </div>
    );
  }
}
```

Summary

- Service Oriented Architecture and Software as a Service (SaaS) simplify the creation of web applications by allowing for combinations of online components
- We can access services via RESTful APIs and develop React components to make use of them



Video 3.7

React App Development

Chris Murphy

Review

- React allows us to create web applications by developing reusable, modular components
- So far, we've seen how to define components within the HTML pages
- How do we develop larger applications with multiple components?

Node.js - Introduction

- **Node.js** is a free, open source platform and framework built in JavaScript
- Includes suite of tools that allows user to prepare JavaScript (and thus React) applications for deployment
- Utilizes Node.js Package Manager (**npm**) to install programs and manage dependencies

Node.js - Benefits

- Instead of including all JavaScript code in a `<script>` tag, now we can separate the components into different files to make code more modular
- Node.js allows us to incorporate dependencies of the code within the current file

```
var React = require('react');  
var ReactDOM = require('react-dom');  
  
import MyComponent from './MyComponent.js';
```


Node.js - Benefits

- Instead of including all JavaScript code in a `<script>` tag, now we can separate the components into different files to make code more modular
- Node.js allows us to incorporate dependencies of the code within the current file

```
var React = require('react');  
var ReactDOM = require('react-dom');  
  
import MyComponent from './MyComponent.js';
```

Node.js - Benefits

- Instead of including all JavaScript code in a `<script>` tag, now we can separate the components into different files to make code more modular
- Node.js allows us to incorporate dependencies of the code within the current file

```
var React = require('react');  
var ReactDOM = require('react-dom');  
  
import MyComponent from './MyComponent.js';
```

Node.js - Installation

- Navigate to <https://nodejs.org/en/download/> and download and install Node.js and appropriate packages
- Although **npm** always comes with a Node.js installation, be sure to update the version to the most recent with the following command.

```
npm install npm -g
```

Creating a React App - Considerations

- Including dependencies (React, React-DOM libraries, etc.)
- Making code compatible with browsers that only support older versions of JavaScript
- Transforming JSX into JavaScript
- Modularity: implementing modules in separate files, bundling them as dependencies

Creating a React App with Node.js - Setup

- Fortunately, there exists a package (through npm) that takes all of the above into consideration when creating a React app
 - Incorporates Babel for JSX and ES6 transformation
 - Incorporates Webpack for bundling

```
npm install -g create-react-app
```

Creating a React App with Node.js - Setup

- Fortunately, there exists a package (through npm) that takes all of the above into consideration when creating a React app
 - Incorporates Babel for JSX and ES6 transformation
 - Incorporates Webpack for bundling

```
npm install -g create-react-app
```

- To create new React app, run the following command in the desired parent directory of new application

```
create-react-app my-app
```

Anatomy of a React App

- **package.json**: information about app, lists of dependencies, shortcuts for scripts
- **public**: directory containing HTML files, images, other static web content
- **src**: directory containing JavaScript and CSS files

Starting React App with Node.js

- You can start the default app as follows:

```
cd my-app/  
npm start
```


Starting React App with Node.js

- You can start the default app as follows:

```
cd my-app/  
npm start
```

- This will start a web server that listens for incoming HTTP requests on port 3000 on your computer

Starting React App with Node.js

- You can start the default app as follows:

```
cd my-app/  
npm start
```

- This will start a web server that listens for incoming HTTP requests on port 3000 on your computer
- You can access the web server by accessing <http://localhost:3000/> from your computer

localhost:3000



Welcome to React

To get started, edit `src/App.js` and save to reload.

Incorporating Components

- We can now create separate JavaScript files for each component.
- **src/Counter.js** would look like this:

```
var React = require('react');

class Counter extends React.Component {
  constructor(props) { . . . }

  incrementCount() { . . . }

  render() { . . . }
};

export default Counter;
```

Incorporating Components

- We can now create separate JavaScript files for each component.
- **src/Counter.js** would look like this:

```
var React = require('react');

class Counter extends React.Component {
  constructor(props) { . . . }

  incrementCount() { . . . }

  render() { . . . }
};

export default Counter;
```

Incorporating Components

- We can now create separate JavaScript files for each component.
- **src/Counter.js** would look like this:

```
var React = require('react');  
  
class Counter extends React.Component {  
  constructor(props) { . . . }  
  
  incrementCount() { . . . }  
  
  render() { . . . }  
};  
  
export default Counter;
```

Incorporating Components

- We can now create separate JavaScript files for each component.
- **src/Counter.js** would look like this:

```
var React = require('react');

class Counter extends React.Component {
  constructor(props) { . . . }

  incrementCount() { . . . }

  render() { . . . }
};

export default Counter;
```

Incorporating Components into the App

- Edit **src/App.js** as follows:

```
import Counter from './Counter.js';
```

```
class App extends Component {  
  render() {  
    return (  
      <div className="App">  
        <Counter />  
      </div>  
    );  
  }  
}
```


Incorporating Components into the App

- Edit `src/App.js` as follows:

```
import Counter from './Counter.js';
```

```
class App extends Component {  
  render() {  
    return (  
      <div className="App">  
        <Counter />  
      </div>  
    );  
  }  
}
```

Incorporating Components into the App

- Edit **src/App.js** as follows:

```
import Counter from './Counter.js';
```

```
class App extends Component {  
  render() {  
    return (  
      <div className="App">  
        <Counter />  
      </div>  
    );  
  }  
}
```



React App



Guest



localhost:3000



Count: 0

Increment



Good Dogs



Princess: Corgi [X](#)



Riley: Husky [X](#)

Add Dog

Name

Image

Breed

Submit



Good Dogs



Princess: Corgi [X](#)



Riley: Husky [X](#)

Add Dog

Name

Image

Breed

localhost:3000

Good Dogs



Princess: Corgi [X](#)



Riley: Husky [X](#)

Add Dog

Name

Image

Breed

Good Dogs



Princess: Corgi [X](#)



Riley: Husky [X](#)



Cooper: Catahoula Leopard [X](#)

Add Dog

Name

Image

Good Dogs



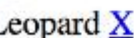
Princess: Corgi



Riley: Husky



Cooper: Catahoula Leopard



Add Dog

Name

Image



Good Dogs



Riley: Husky [X](#)



Cooper: Catahoula Leopard [X](#)

Add Dog

Name

Image

Breed

Submit

Good Dogs



Riley: Husky [X](#)



Cooper: Catahoula Leopard [X](#)

Add Dog

Name

Image

Breed

Submit

App

Dogs

Good Dogs



Riley: Husky [X](#)



Cooper: Catahoula Leopard [X](#)

App

Add Dog

Name

Image

Breed

Good Dogs



Riley: Husky X



Cooper: Catahoula Leopard X

Dogs

DogItem

App

DogItem

Add Dog

Name

Image

Breed

Good Dogs



Riley: Husky X



Cooper: Catahoula Leopard X

Dogs

DogItem

App

DogItem

Add Dog

Name

Cooper

Image

https://upload.wikimedia

Breed

Catahoula Leopard

Submit

AddDog

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
```

```
    return (
```

```
      <div className="App">
```

```
        <Dogs dogs={this.state.dogs}
```

```
          onDelete={this.handleDeleteDog.bind(this)} />
```

```
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
        <hr />
```

```
      </div>
```

```
    );
```

```
  }
```

```
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';

class App extends Component {
  constructor() { . . . }

  handleAddDog(dog) { . . . }

  handleDeleteDog(name) { . . . }

  render() {
    return (
      <div className="App">
        <Dogs dogs={this.state.dogs}
          onDelete={this.handleDeleteDog.bind(this)} />
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
        <hr />
      </div>
    );
  }
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }

  handleAddDog(dog) { . . . }

  handleDeleteDog(name) { . . . }

  render() {
    return (
      <div className="App">
        <Dogs dogs={this.state.dogs}
          onDelete={this.handleDeleteDog.bind(this)} />
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
        <hr />
      </div>
    );
  }
};
```



```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';

class App extends Component {
  constructor() { . . . }

  handleAddDog(dog) { . . . }

  handleDeleteDog(name) { . . . }

  render() {
    return (
      <div className="App">
        <Dogs dogs={this.state.dogs}
          onDelete={this.handleDeleteDog.bind(this)} />
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
        <hr />
      </div>
    );
  }
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }

  handleAddDog(dog) { . . . }

  handleDeleteDog(name) { . . . }

  render() {
    return (
      <div className="App">
        <Dogs dogs={this.state.dogs}
          onDelete={this.handleDeleteDog.bind(this)} />
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
        <hr />
      </div>
    );
  }
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
```

```
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
```

```
    return (
```

```
      <div className="App">
```

```
        <Dogs dogs={this.state.dogs}
```

```
          onDelete={this.handleDeleteDog.bind(this)} />
```

```
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
        <hr />
```

```
      </div>
```

```
    );
```

```
  }
```

```
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }

  handleAddDog(dog) { . . . }

  handleDeleteDog(name) { . . . }

  render() {
    return (
      <div className="App">
        <Dogs dogs={this.state.dogs}
          onDelete={this.handleDeleteDog.bind(this)} />
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
        <hr />
      </div>
    );
  }
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
    return (
      <div className="App">
        <Dogs dogs={this.state.dogs}
          onDelete={this.handleDeleteDog.bind(this)} />
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
        <hr />
      </div>
    );
  }
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
```

```
    return (
```

```
      <div className="App">
```

```
        <Dogs dogs={this.state.dogs}
```

```
          onDelete={this.handleDeleteDog.bind(this)} />
```

```
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
        <hr />
```

```
      </div>
```

```
    );
```

```
  }
```

```
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
render () {
```

```
  return (
```

```
    <div className="App">
```

```
      <Dogs dogs={this.state.dogs}
```

```
        onDelete={this.handleDeleteDog.bind(this)} />
```

```
      <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
      <hr />
```

```
    </div>
```

```
  );
```

```
}
```

```
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
```

```
    return (
```

```
      <div className="App">
```

```
        <Dogs dogs={this.state.dogs}
```

```
          onDelete={this.handleDeleteDog.bind(this)} />
```

```
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
        <hr />
```

```
      </div>
```

```
    );
```

```
  }
```

```
};
```


Dogs

Good Dogs



Riley: Husky [X](#)



Cooper: Catahoula Leopard [X](#)

App

Add Dog

Name

Image

Breed

Submit

AddDog

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
```

```
    return (
```

```
      <div className="App">
```

```
        <Dogs dogs={this.state.dogs}
```

```
          onDelete={this.handleDeleteDog.bind(this)} />
```

```
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
        <hr />
```

```
      </div>
```

```
    );
```

```
  }
```

```
};
```

```
import React, { Component } from 'react';
import Dogs from './components/Dogs';
import AddDog from './components/AddDog';
import './App.css';
```

```
class App extends Component {
  constructor() { . . . }
```

```
  handleAddDog(dog) { . . . }
```

```
  handleDeleteDog(name) { . . . }
```

```
  render() {
```

```
    return (
```

```
      <div className="App">
```

```
        <Dogs dogs={this.state.dogs}
```

```
          onDelete={this.handleDeleteDog.bind(this)} />
```

```
        <AddDog onAddDog={this.handleAddDog.bind(this)} />
```

```
        <hr />
```

```
      </div>
```

```
    );
```

```
  }
```

```
};
```

Testing React Apps

- **Mocha** – widely used test runner (testing framework) used to run JavaScript tests
- **Chai** – assertion library for Behavior Driven Testing
- **Enzyme** – testing utility for React for manipulating and inspecting React Component state and output



Getting Started - Installation

- To include Enzyme and Chai as dependencies, run the following command:

```
npm install --save-dev enzyme react-test-renderer chai
```

Getting Started - Installation

- To include Enzyme and Chai as dependencies, run the following command:

```
npm install --save-dev enzyme react-test-renderer chai
```

- Note that the default file structure places all JavaScript and CSS code in the 'src' folder.
- We will create an additional folder within 'src' named 'tests' in which we include all testing scripts
- All test files must be in the form of *.test.js, e.g. **Dogs.test.js**

Getting Started

- Node.js should create a default **App.test.js**, or you can write your own

Getting Started

- Node.js should create a default **App.test.js**, or you can write your own
- Include libraries necessary for testing
 - Import React and ReactDOM for component manipulation

```
import React from 'react';  
import ReactDOM from 'react-dom';
```

- Import keywords from Enzyme

```
import { mount, shallow } from 'enzyme';
```

- Import keywords from Chai

```
import { expect } from 'chai';
```


Getting Started

- Node.js should create a default **App.test.js**, or you can write your own
- Include libraries necessary for testing
 - Import React and ReactDOM for component manipulation

```
import React from 'react';  
import ReactDOM from 'react-dom';
```

- Import keywords from Enzyme

```
import { mount, shallow } from 'enzyme';
```

- Import keywords from Chai

```
import { expect } from 'chai';
```

Getting Started

- Node.js should create a default **App.test.js**, or you can write your own
- Include libraries necessary for testing
 - Import React and ReactDOM for component manipulation

```
import React from 'react';  
import ReactDOM from 'react-dom';
```

- Import keywords from Enzyme

```
import { mount, shallow } from 'enzyme';
```

- Import keywords from Chai

```
import { expect } from 'chai';
```

Getting Started

- Node.js should create a default **App.test.js**, or you can write your own
- Include libraries necessary for testing
 - Import React and ReactDOM for component manipulation

```
import React from 'react';  
import ReactDOM from 'react-dom';
```

- Import keywords from Enzyme

```
import { mount, shallow } from 'enzyme';
```

- Import keywords from Chai

```
import { expect } from 'chai';
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```


Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```


Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App").length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Anatomy of a React Test

```
import React from 'react';
import { expect } from 'chai';
import { mount, shallow } from 'enzyme';
import App from '../App';

describe("Test suite for App component", function() {

  it("only one element in App class", function() {
    const wrapper = shallow(<App />);
    expect(wrapper.find(".App")).length(1);
  });

});
```

Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```

Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```

Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```


Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```

Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```

Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```

Testing React Component Relationships

```
it('Dog List contains two dogs', function() {  
  const wrapper = mount(<App/>);  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(2);  
});
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
function() {
  const wrapper = mount(<App/>);
  const adddog = wrapper.find('AddDog');

  adddog.find('#dogName').get(0).value = 'Lola';
  adddog.find('#imageUrl').get(0).value = 'https://
    static.pexels.com/photos/54386/pexels-
    photo-54386.jpeg';
  adddog.find('#dogBreed').get(0).value = 'Beagle';

  const form = adddog.find('form');
  form.simulate('submit');
  expect(wrapper.find('Dogs')
    .find('DogItem')).toHaveLength(3);
  expect(wrapper.state().dogs[2].name == 'Lola');
});
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).length(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).length(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```


Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).length(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).length(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).length(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```


Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).length(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Data Entry and Form Submission

```
it("successfully adds dog to list when form submitted",
  function() {
    const wrapper = mount(<App/>);
    const adddog = wrapper.find('AddDog');

    adddog.find('#dogName').get(0).value = 'Lola';
    adddog.find('#imageUrl').get(0).value = 'https://
      static.pexels.com/photos/54386/pexels-
      photo-54386.jpeg';
    adddog.find('#dogBreed').get(0).value = 'Beagle';

    const form = adddog.find('form');
    form.simulate('submit');
    expect(wrapper.find('Dogs')
      .find('DogItem')).toHaveLength(3);
    expect(wrapper.state().dogs[2].name == 'Lola');
  });
```

Testing Links

```
it('removes dog from list when deleted', function() {  
  const wrapper = mount(<App/>);  
  const deleteLink = wrapper.find('a').first();  
  
  deleteLink.simulate('click');  
  
  expect(wrapper.find('Dogs')  
            .find('DogItem')).length(1);  
});
```


Testing Links

```
it('removes dog from list when deleted', function() {  
  const wrapper = mount(<App/>);  
  const deleteLink = wrapper.find('a').first();  
  
  deleteLink.simulate('click');  
  
  expect(wrapper.find('Dogs')  
            .find('DogItem')).length(1);  
});
```

Testing Links

```
it('removes dog from list when deleted', function() {  
  const wrapper = mount(<App/>);  
  const deleteLink = wrapper.find('a').first();  
  
  deleteLink.simulate('click');  
  
  expect(wrapper.find('Dogs')  
            .find('DogItem')).length(1);  
});
```

Testing Links

```
it('removes dog from list when deleted', function() {  
  const wrapper = mount(<App/>);  
  const deleteLink = wrapper.find('a').first();  
  
  deleteLink.simulate('click');  
  
  expect(wrapper.find('Dogs')  
            .find('DogItem')).length(1);  
});
```

Testing Links

```
it('removes dog from list when deleted', function() {  
  const wrapper = mount(<App/>);  
  const deleteLink = wrapper.find('a').first();  
  
  deleteLink.simulate('click');  
  
  expect(wrapper.find('Dogs')  
    .find('DogItem')).length(1);  
});
```

Testing Links

```
it('removes dog from list when deleted', function() {  
  const wrapper = mount(<App/>);  
  const deleteLink = wrapper.find('a').first();  
  
  deleteLink.simulate('click');  
  
  expect(wrapper.find('Dogs')  
            .find('DogItem')).length(1);  
});
```

Running Tests

- To run tests, navigate to the project within the terminal and run the following command:

```
npm run test
```

FAIL src/test/AddDog.test.js

● **<AddDog/> > successfully adds dog to list when form submitted**

AssertionError: expected { Object (component, root, ...) } to have a length of 4 but got 3
at Object.<anonymous> (src/test/AddDog.test.js:36:85)

FAIL src/test/App.test.js

● **<App/> - loads > Dog List contains 3 dogs**

AssertionError: expected { Object (component, root, ...) } to have a length of 3 but got 2
at Object.<anonymous> (src/test/App.test.js:36:85)

FAIL src/test/Dogs.test.js

● **<Dogs/> > removes dog from list when deleted**

AssertionError: expected { Object (component, root, ...) } to have a length of 2 but got 1
at Object.<anonymous> (src/test/Dogs.test.js:25:72)

Test Suites: **3 failed**, 3 total
Tests: **3 failed**, **5 passed**, 8 total
Snapshots: 0 total
Time: 2.3s, estimated 3s
Ran all test suites.

Watch Usage

- > Press **p** to filter by a filename regex pattern.
- > Press **q** to quit watch mode.
- > Press **Enter** to trigger a test run.

Running Tests: Success!

```
PASS src/test/App.test.js
PASS src/test/Dogs.test.js
PASS src/test/AddDog.test.js
```

```
Test Suites: 3 passed, 3 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        2.398s
Ran all test suites.
```

Watch Usage

- > Press `o` to only run tests related to changed files.
- > Press `p` to filter by a filename regex pattern.
- > Press `q` to quit watch mode.
- > Press `Enter` to trigger a test run.



Summary

- We can use **Node.js** to create React applications
- This allows us to put component code into separate .js files and then include them into our App as necessary
- **Mocha**, **Chai**, and **Enzyme** can be used for testing our React apps



Video 3.8

ES6

Chris Murphy

What is ES6?

- **ES6** stands for ECMAScript 6
- ECMAScript is the "proper" name for JavaScript
- ES6 is the newest JavaScript Specification, released in 2015

What can you do with ES6?

- In ES6, you can...
 - Define constants
 - Use simpler notations for function declarations
 - Build classes
 - Refactor code into modules
 - Store data in Sets, Maps, and Typed Arrays
 - Copy objects in one line of code
 - ... and much more!

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];
var square = function (n) {
    return n*n;
};
arr.forEach( function(v, i) {
    arr[i] = square(v);
});
```

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];  
var square = function (n) {  
    return n*n;  
};  
arr.forEach( function(v, i) {  
    arr[i] = square(v);  
});
```

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];  
var square = function (n) {  
    return n*n;  
};  
arr.forEach( function(v, i) {  
    arr[i] = square(v);  
});
```

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];
var square = function (n) {
    return n*n;
};
arr.forEach( function(v, i) {
    arr[i] = square(v);
});
```

- ES6 Syntax:

```
let arr = [1,2,3,4,5];
let square = n => {
    return n*n;
};
arr.forEach( (v, i) => {
    arr[i] = square(v);
});
```


ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];
var square = function (n) {
    return n*n;
};
arr.forEach( function(v, i) {
    arr[i] = square(v);
});
```

- ES6 Syntax:

```
let arr = [1,2,3,4,5];
let square = n => {
    return n*n;
};
arr.forEach( (v, i) => {
    arr[i] = square(v);
});
```

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];
var square = function (n) {
    return n*n;
};
arr.forEach( function(v, i) {
    arr[i] = square(v);
});
```

- ES6 Syntax:

```
let arr = [1,2,3,4,5];
let square = n => {
    return n*n;
};
arr.forEach( (v, i) => {
    arr[i] = square(v);
});
```

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];
var square = function (n) {
    return n*n;
};
arr.forEach( function(v, i) {
    arr[i] = square(v);
});
```

- ES6 Syntax:

```
let arr = [1,2,3,4,5];
let square = n => {
    return n*n;
};
arr.forEach( (v, i) => {
    arr[i] = square(v);
});
```

ES6 – Arrow Functions

- New syntax for defining functions using arrows

- ES5 Syntax:

```
var arr = [1,2,3,4,5];
var square = function (n) {
    return n*n;
};
arr.forEach( function(v, i) {
    arr[i] = square(v);
});
```

- ES6 Syntax:

```
let arr = [1,2,3,4,5];
let square = n => {
    return n*n;
};
arr.forEach( (v, i) => {
    arr[i] = square(v);
});
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));
```


ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));           // 9
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));           // 9  
  
console.log(pow(3, 3));
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));           // 9  
  
console.log(pow(3, 3));
```

ES6 – Default Parameter Values

- The “=” symbol can be used to assign default values to function parameters

```
function pow (base, power = 2) {  
    return Math.pow(base, power);  
};  
  
console.log(pow(3));           // 9  
  
console.log(pow(3, 3));       // 27
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };
```

```
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```


ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };
```

```
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ", \n" + "How are you? ";
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

- ES6 Syntax:

```
var person = { name: "Lydia" };  
var msg = `Dear ${person.name},  
          How are you?`
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

- ES6 Syntax:

```
var person = { name: "Lydia" };  
var msg = `Dear ${person.name},  
How are you?`
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

- ES6 Syntax:

```
var person = { name: "Lydia" };  
var msg = `Dear ${person.name},  
How are you?`
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

- ES6 Syntax:

```
var person = { name: "Lydia" };  
var msg = `Dear ${person.name},  
           How are you?`
```

ES6 – Template Literals

- Can define a template for rendering strings
- ES5 Syntax:

```
var person = { name: "Lydia" };  
var msg = "Dear " + person.name + ",\n" + "How are you? ";
```

- ES6 Syntax:

```
var person = { name: "Lydia" };  
var msg = `Dear ${person.name},  
How are you?`
```


ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}  
  
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}
```

```
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}
```

```
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {
    this.height = height;
    this.width = width;
}

Rectangle.prototype.area = function () {
    return this.height * this.width;
}
```

- ES6 Syntax:

```
class Rectangle {
    constructor (height, width) {
        this.height = height;
        this.width = width;
    }
    area () {
        return this.height * this.width;
    }
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}
```

```
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

- ES6 Syntax:

```
class Rectangle {  
    constructor (height, width) {  
        this.height = height;  
        this.width = width;  
    }  
    area () {  
        return this.height * this.width;  
    }  
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}
```

```
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

- ES6 Syntax:

```
class Rectangle {  
    constructor (height, width) {  
        this.height = height;  
        this.width = width;  
    }  
    area () {  
        return this.height * this.width;  
    }  
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {
    this.height = height;
    this.width = width;
}

Rectangle.prototype.area = function () {
    return this.height * this.width;
}
```

- ES6 Syntax:

```
class Rectangle {
    constructor (height, width) {
        this.height = height;
        this.width = width;
    }
    area () {
        return this.height * this.width;
    }
}
```

ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}  
  
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

- ES6 Syntax:

```
class Rectangle {  
    constructor (height, width) {  
        this.height = height;  
        this.width = width;  
    }  
    area () {  
        return this.height * this.width;  
    }  
}
```


ES6 – Classes

- Instead of building prototypes, ES6 allows classes to be directly defined in more traditional OOP style

- ES5 Syntax:

```
var Rectangle = function (height, width) {  
    this.height = height;  
    this.width = width;  
}  
  
Rectangle.prototype.area = function () {  
    return this.height * this.width;  
}
```

- ES6 Syntax:

```
class Rectangle {  
    constructor (height, width) {  
        this.height = height;  
        this.width = width;  
    }  
    area () {  
        return this.height * this.width;  
    }  
}
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");      // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```


ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v); // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
  console.log(v);      // prints each value in order
```

ES6 Data Structures: Sets

- ES6 introduces a **Set** class
- Elements are distinct and maintain order

```
let s = new Set();

s.add("alligator"); // s = {"alligator"}
s.add("dolphin");   // s = {"alligator", "dolphin"}
s.add("fox");       // s = {"alligator", "dolphin", "fox"}
s.add("alligator"); // s = {"alligator", "dolphin", "fox"}

s.has("alligator"); // true

s.delete("alligator"); // s = {"dolphin", "fox"}

for (let v of s.values())
    console.log(v);    // prints each value in order
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```


ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

ES6 Data Structures: Maps

- ES6 also introduces a **Map** class
- A **Set** of keys is mapped to corresponding values

```
let m = new Map();

m.set("dog", "rover"); // {"dog" => "rover"}
m.set("cat", "felix"); // {"dog" => "rover", "cat" => "felix"}

m.get("cat"); // "felix"
m.get("mouse"); // undefined

for (let [key, val] of m.entries())
  console.log(key + ": " + val); // prints keys and values
```

Summary

- ES6 provides simplified syntax and new libraries and functionality
- We will use ES6 notation in the remaining lessons in the course



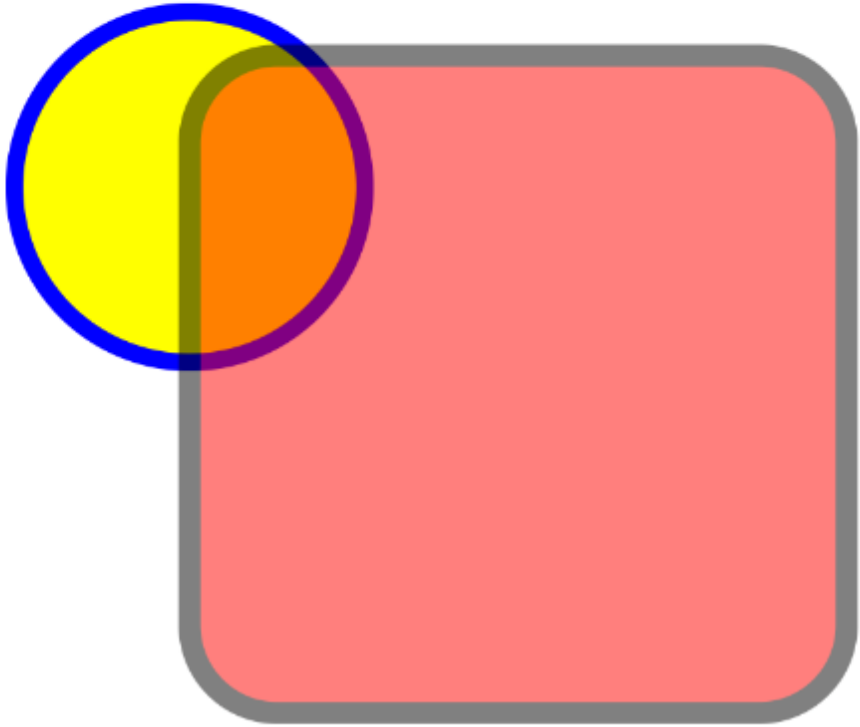
Video 3.9
D3 Intro
Chris Murphy

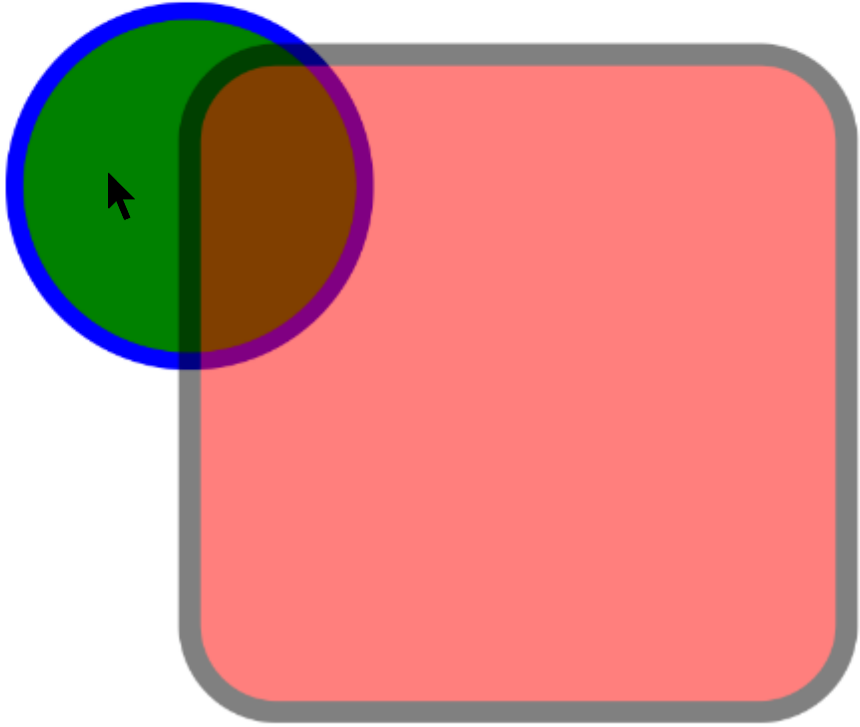
Review

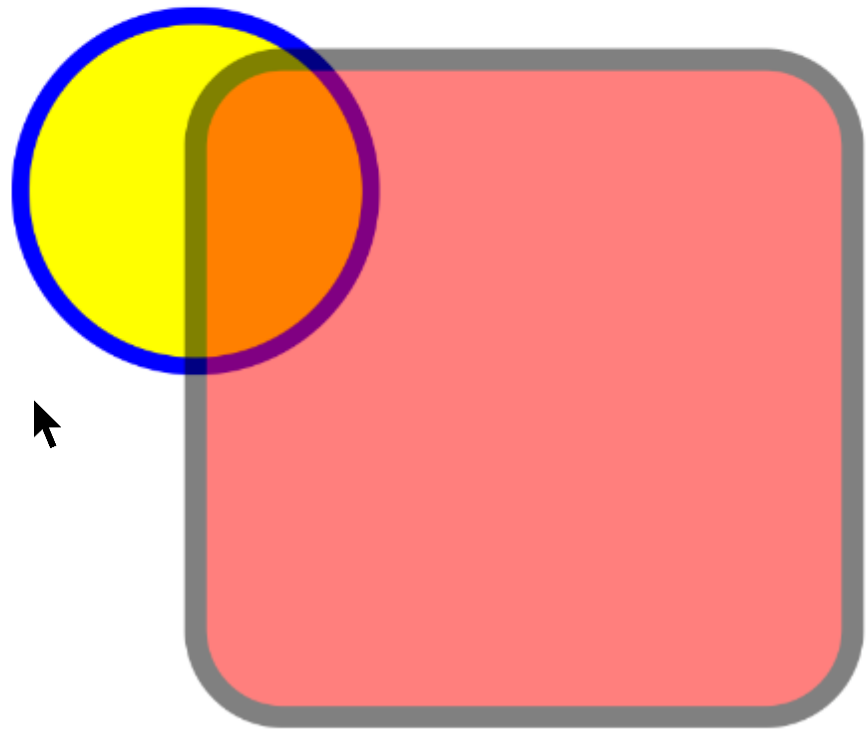
- We can use **JavaScript** to dynamically modify/create HTML elements in Web pages
- **jQuery** provides a simpler syntax and other additional features
- **React** allows us to create reusable, modular components

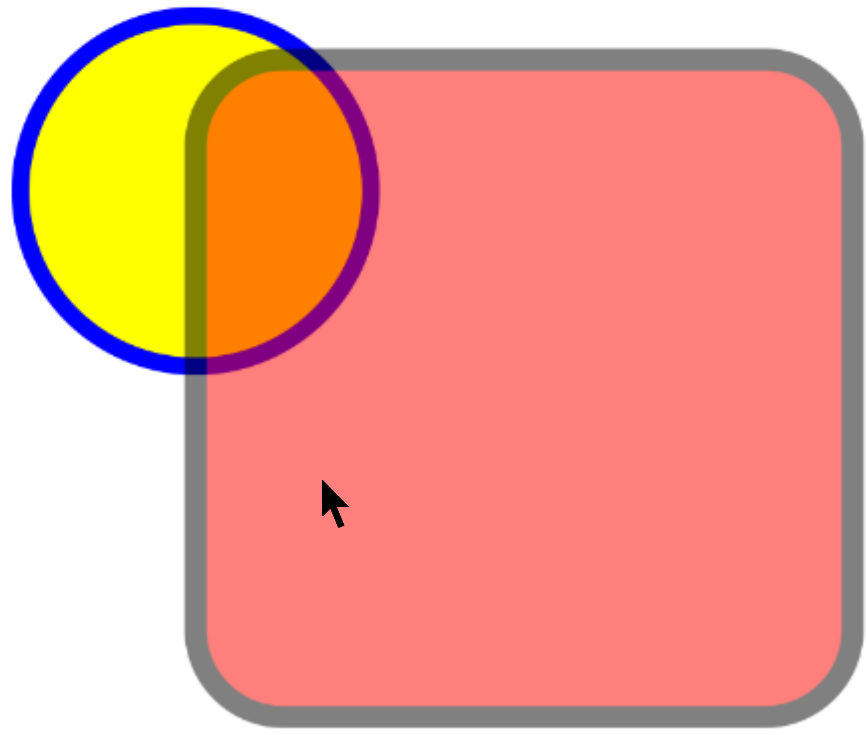
Standard SVG

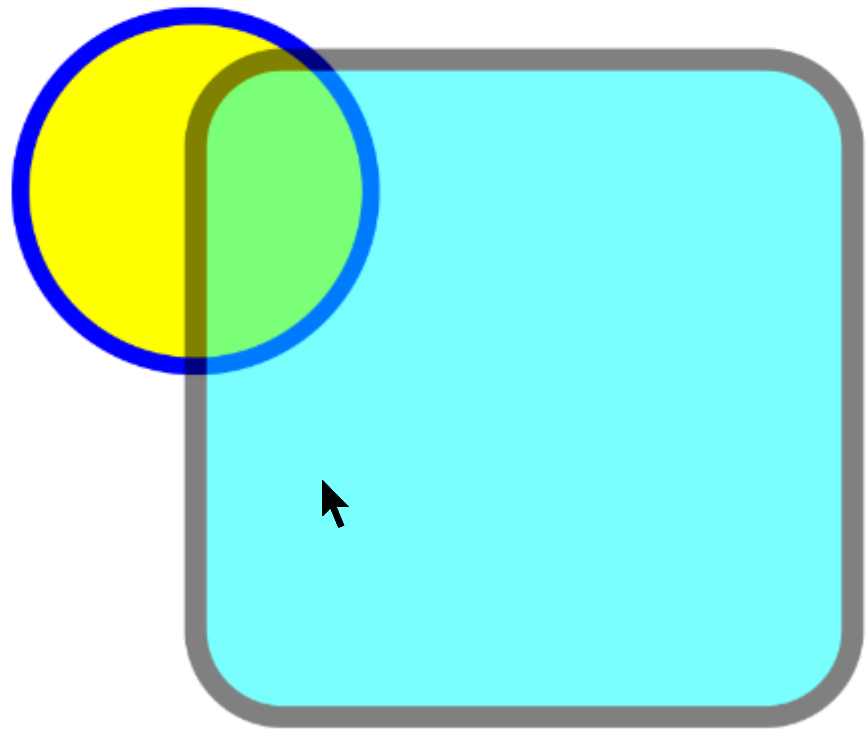
- We can render basic visual elements in HTML using **Scalable Vector Graphics (SVG)**
- These are HTML elements that are included in the Web page
- They do not lose quality when the page is resized
- SVG elements are part of the DOM, so their attributes can be modified by CSS and JavaScript











```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```



```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

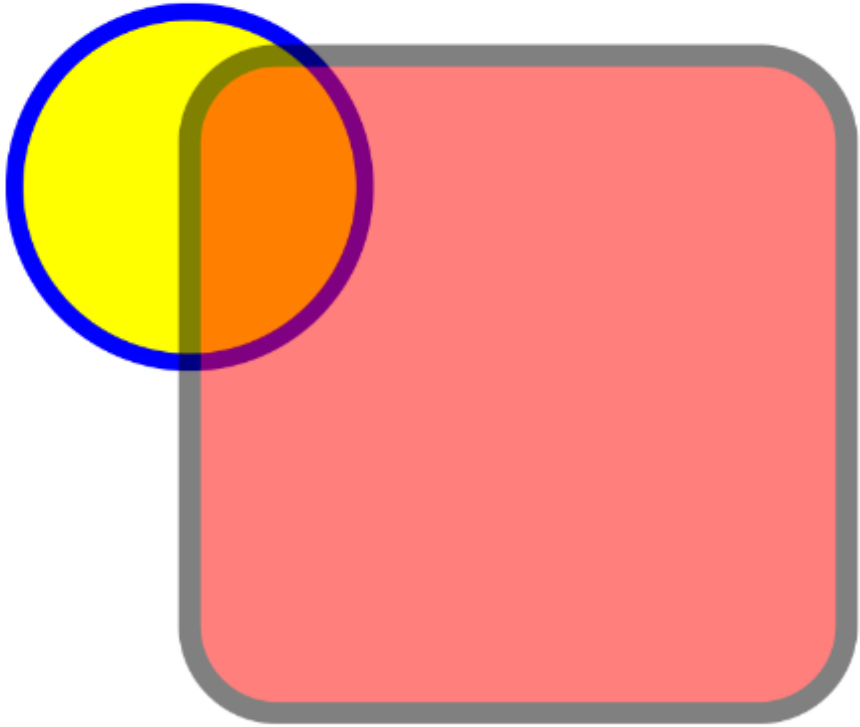
</svg>
</body>
</html>
```

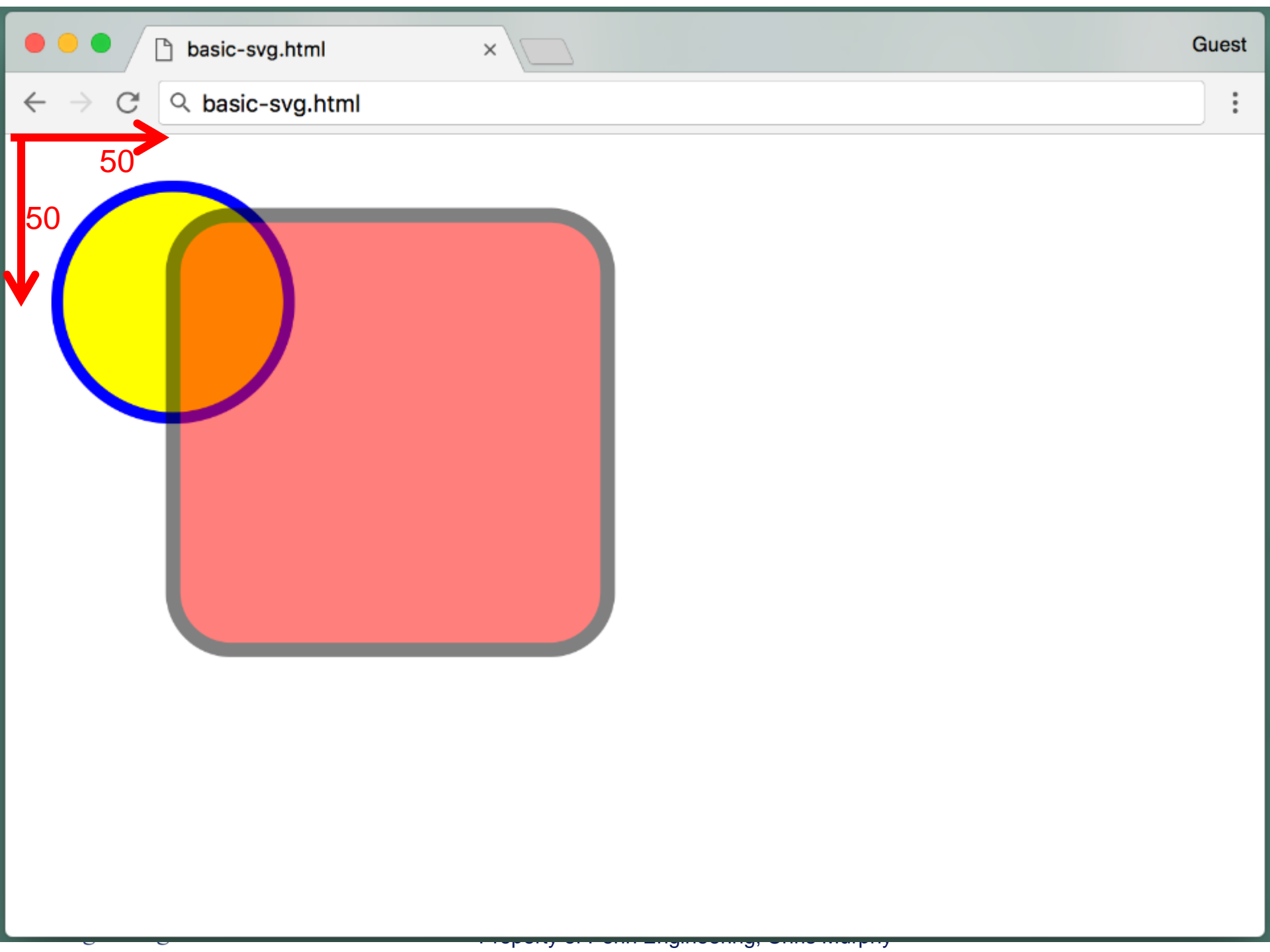


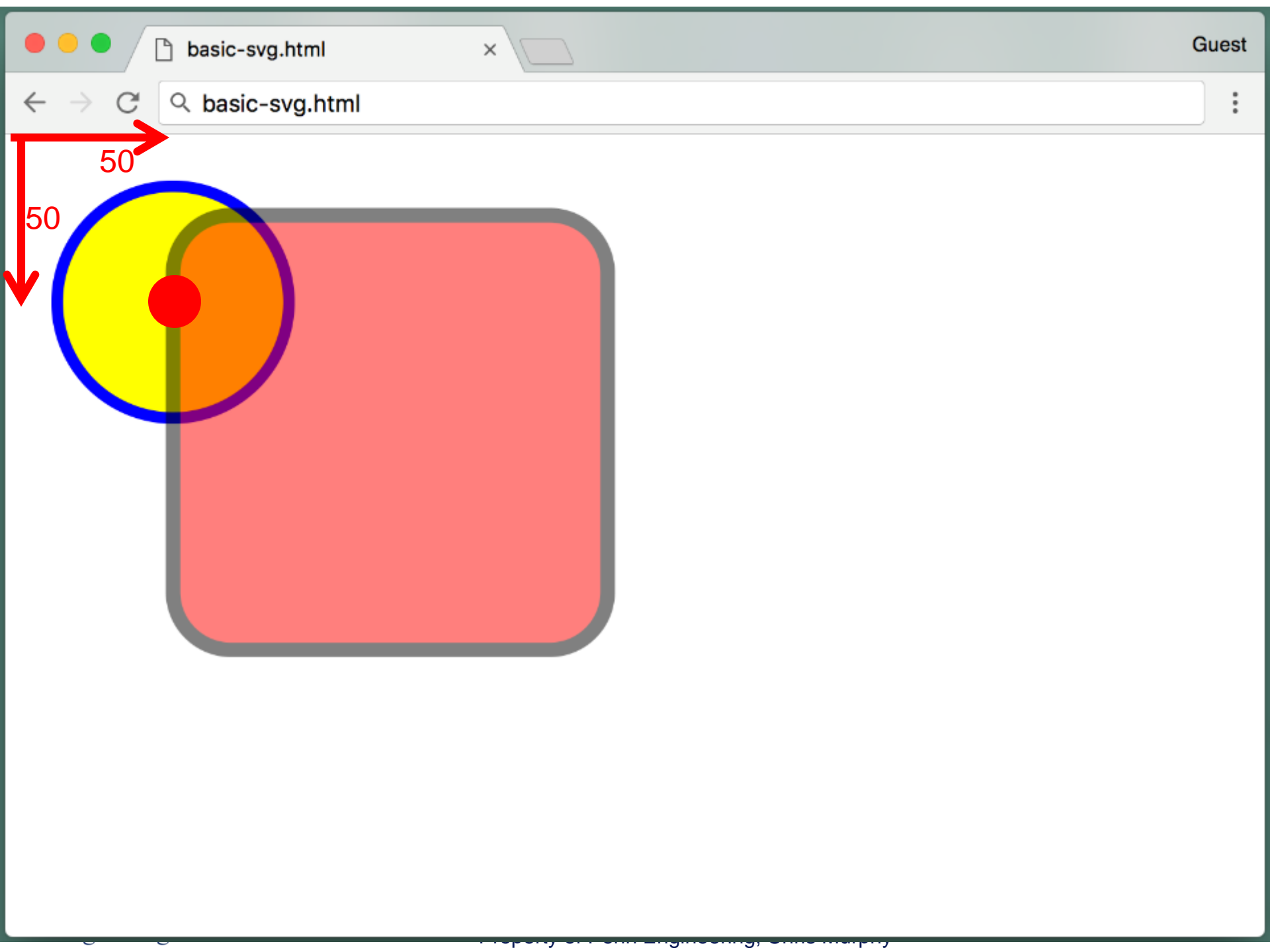
basic-svg.html



50







```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```



```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

```
<html>
<head>
<style>
circle:hover {
    fill:green;
};
</style>
</head>

<body>
<svg width="400" height="400">

    <circle cx="50" cy="50" r="40"
        stroke="blue" stroke-width="4" fill="yellow" />

    <rect x="50" y="20" rx="20" ry="20"
        width="150" height="150"
        style="fill:red;stroke:black;stroke-width:5;opacity:0.5"
        onclick="style.fill='cyan'" />

</svg>
</body>
</html>
```

Dynamic SVG

- Since SVG elements are part of the DOM, we can use JavaScript to manipulate their attributes

- But we may also want to dynamically generate SVG elements based on user actions, data received from an API, etc.

D3: Data-Driven Documents

- **D3.js** is a JavaScript library for manipulating HTML documents based on data
- Data can be bound to DOM elements (HTML, SVG) and then we can programmatically apply data-driven transformations
- This can be used for generating HTML tables, SVG charts and graphs, etc.

```
<html>
<head>
</head>
<body>
<svg width="400" height="400">
  <circle cx="50" cy="50" r="40"
    stroke="blue" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

```
<html>
<head>
</head>
<body>
<svg width="400" height="400">
  <circle cx="50" cy="50" r="40"
    stroke="blue" stroke-width="4" fill="yellow" />
</svg>
</body>
</html>
```



```
<html>
<head>
</head>
<body>
<svg width="400" height="400">
  <circle cx="50" cy="50" r="40"
    stroke="blue" stroke-width="4" fill="yellow" />
</svg>

</body>
</html>
```

```
<html>

<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>

<svg width="400" height="400">
  <circle />

</svg>

<script>
  // next slide -->
</script>

</body>
</html>
```

```
<html>

<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>

<svg width="400" height="400">
  <circle />

</svg>

<script>
  // next slide -->
</script>

</body>
</html>
```

```
<html>

<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>

<svg width="400" height="400">
  <circle />

</svg>

<script>
  // next slide -->
</script>

</body>
</html>
```

```
<html>

<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>

<svg width="400" height="400">
  <circle />

</svg>

<script>
  // next slide -->
</script>

</body>
</html>
```

```
<html>

<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>

<svg width="400" height="400">
  <circle />

</svg>

<script>
  // next slide -->
</script>

</body>
</html>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

<script>

```
var circle = {  
  x: 50,  
  y: 50,  
  r: 40,  
  stroke: 'blue',  
  width: 4,  
  fill: 'yellow'  
};  
  
var svg = d3.select("svg");  
svg.select("circle")  
  .attr("cx", circle.x)  
  .attr("cy", circle.y)  
  .attr("r", circle.r)  
  .style("stroke", circle.stroke)  
  .style("stroke-width", circle.width);  
  .style("fill", circle.fill);
```

</script>


```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```



```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```



```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4,
    fill: 'yellow'
  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", circle.fill);

</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", () => {
      if (circle.r < 50) return 'yellow';
      else return 'cyan'; })
</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", () => {
      if (circle.r < 50) return 'yellow';
      else return 'cyan'; })
</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", () => {
      if (circle.r < 50) return 'yellow';
      else return 'cyan'; })
</script>
```



```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", () => {
      if (circle.r < 50) return 'yellow';
      else return 'cyan'; })
</script>
```

```
<script>
  var circle = {
    x: 50,
    y: 50,
    r: 40,
    stroke: 'blue',
    width: 4

  };

  var svg = d3.select("svg");
  svg.select("circle")
    .attr("cx", circle.x)
    .attr("cy", circle.y)
    .attr("r", circle.r)
    .style("stroke", circle.stroke)
    .style("stroke-width", circle.width);
    .style("fill", () => {
      if (circle.r < 50) return 'yellow';
      else return 'cyan'; })
</script>
```

Summary

- We can render basic visual elements in HTML using Scalable Vector Graphics (**SVG**)
- We can use **D3.js** to programmatically generate HTML elements (including SVG) based on data



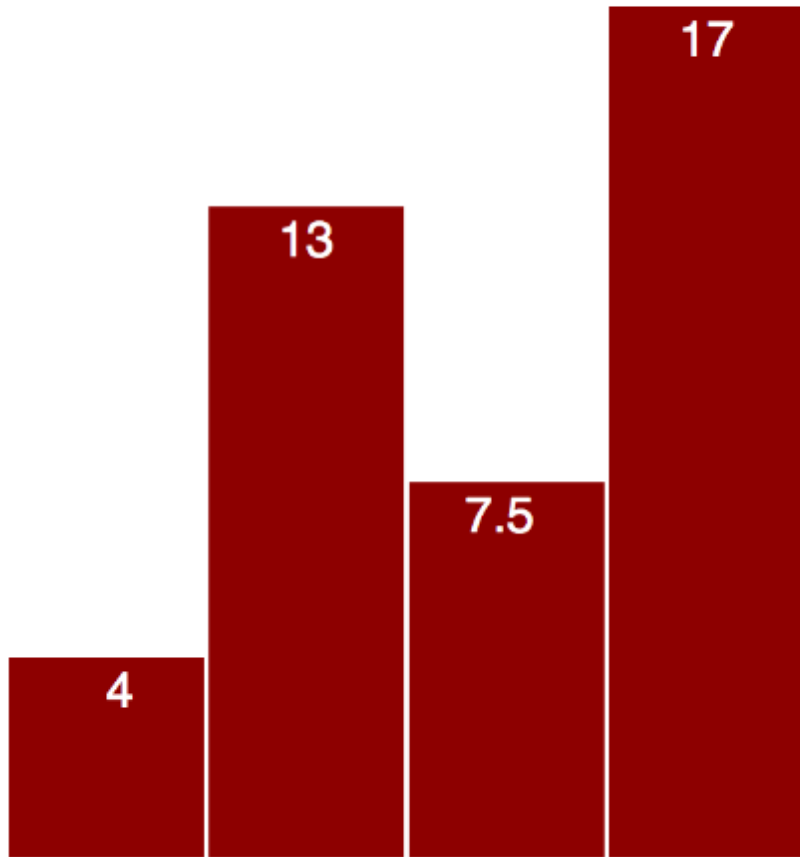
Video 3.10

D3 Charts

Chris Murphy

Review

- D3.js allows us to programmatically generate HTML elements (including SVG) based on data
- The most common visual representation of data is in the form of a **chart**

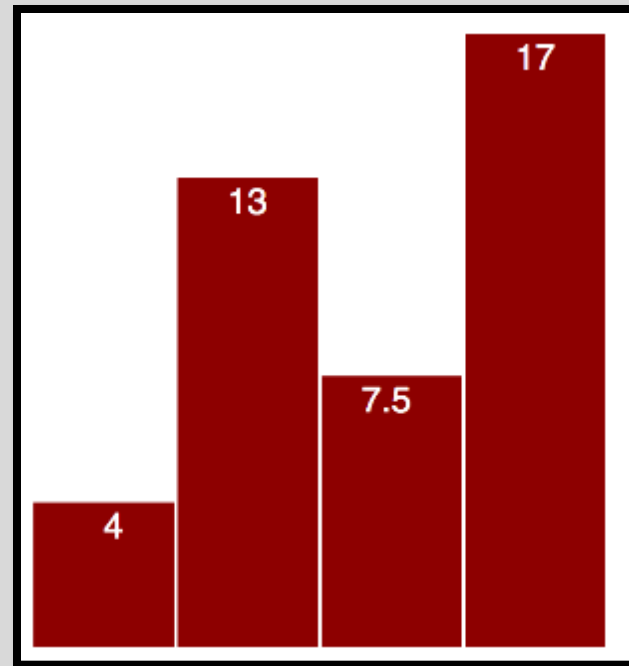



```
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>
```



```
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
```

```
<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>
```



```

<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```
<svg class="chart" height="200">
```

```

  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>

```

```

  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>

```

```

  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>

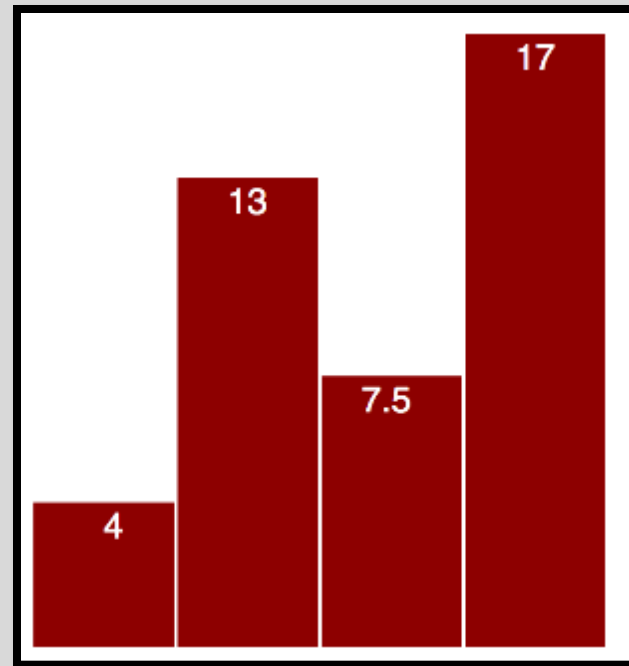
```

```

  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>

```

```
</svg>
```



```

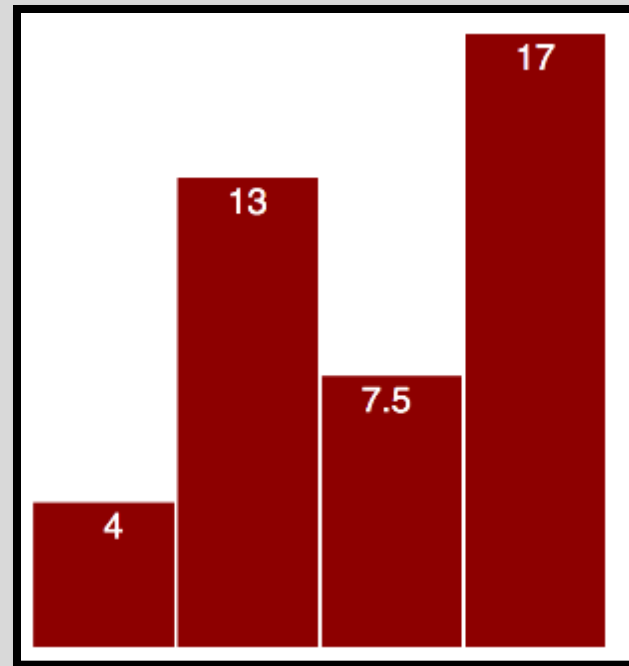
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

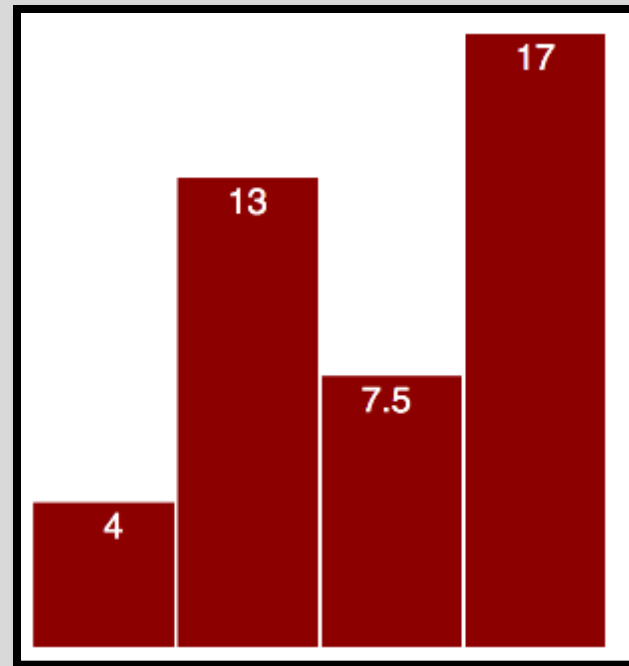
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

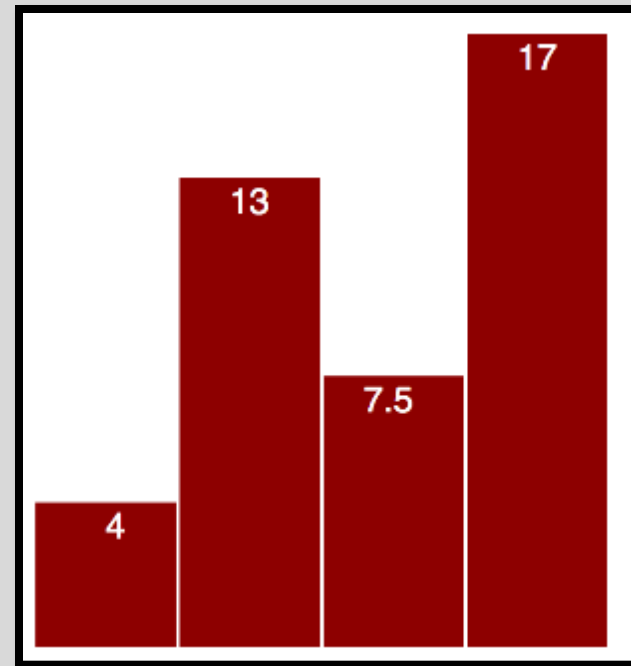
<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
```

```
<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>
```



```

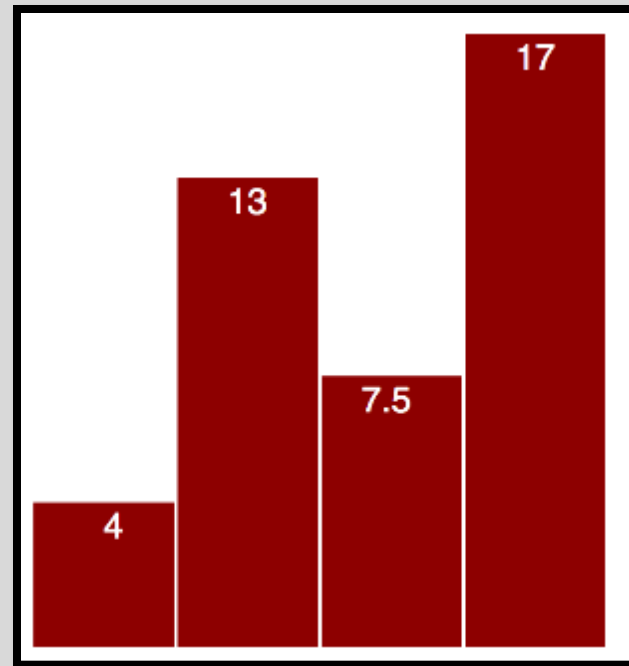
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

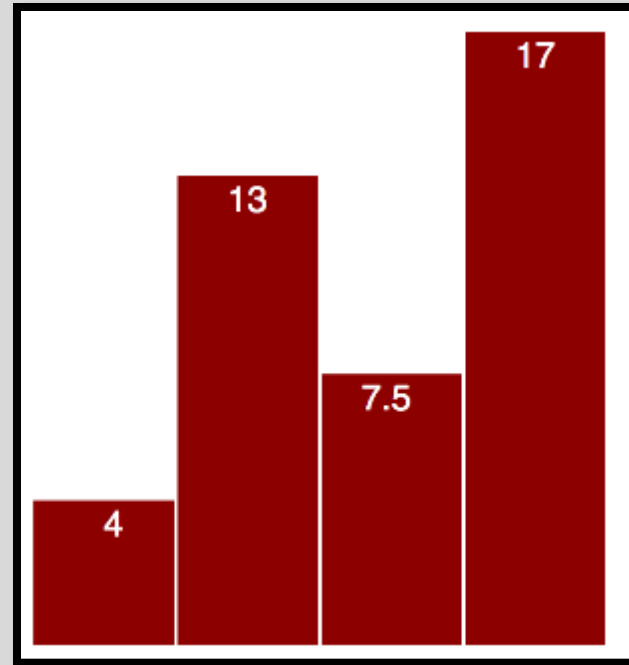
```

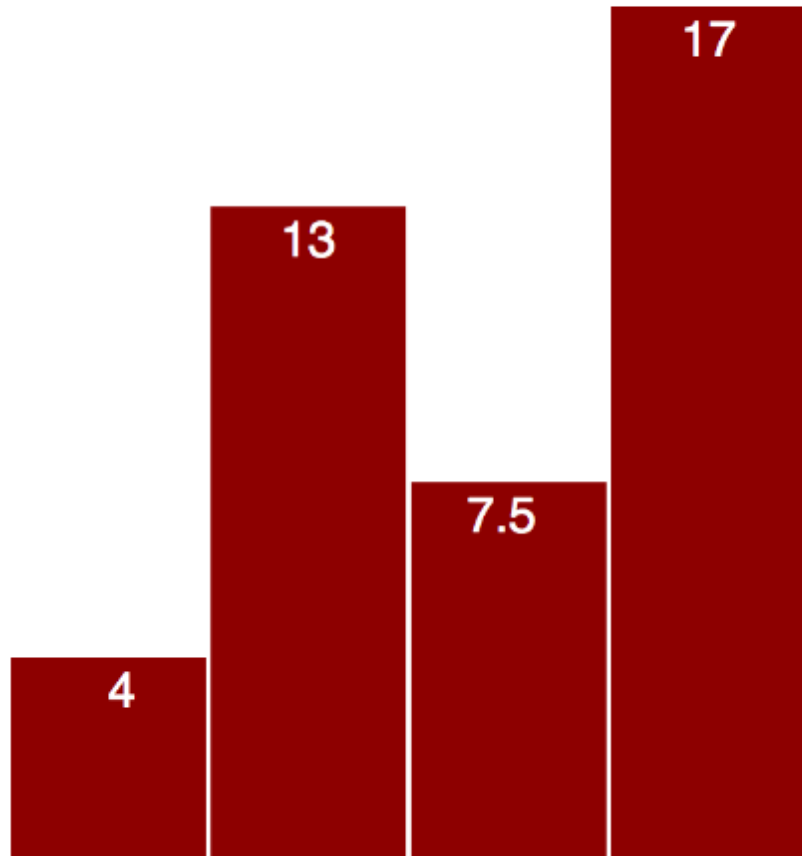


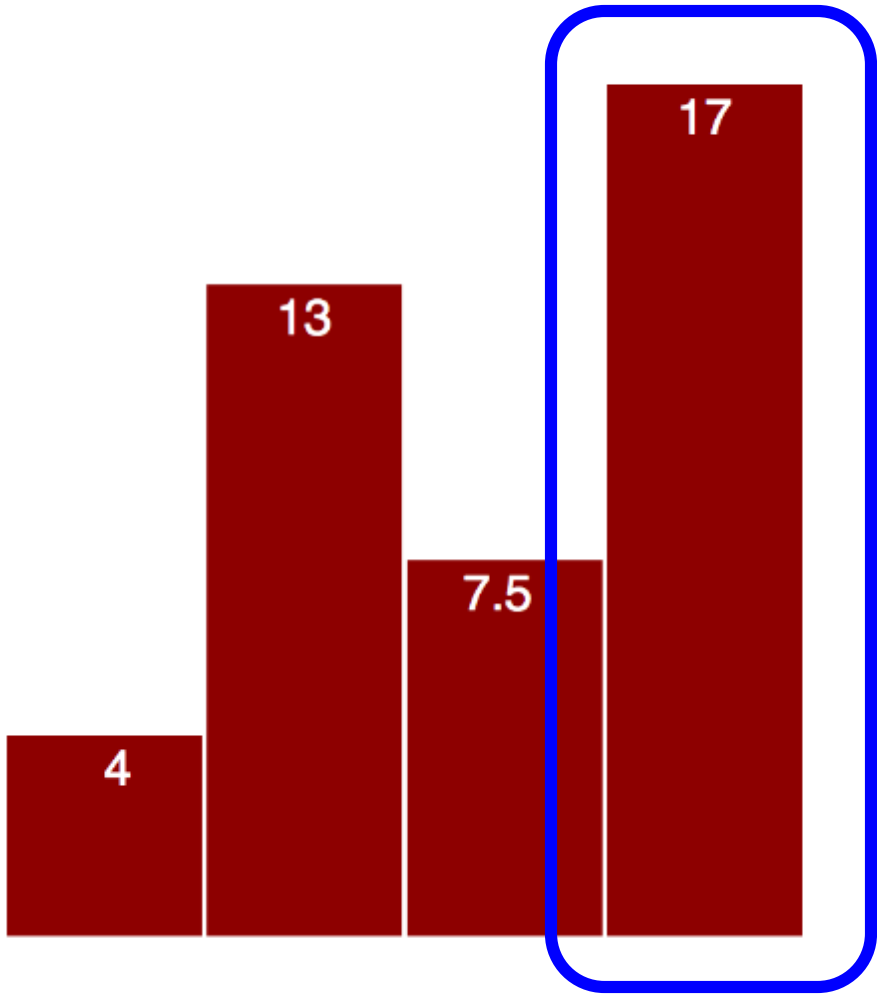
```

<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```







```

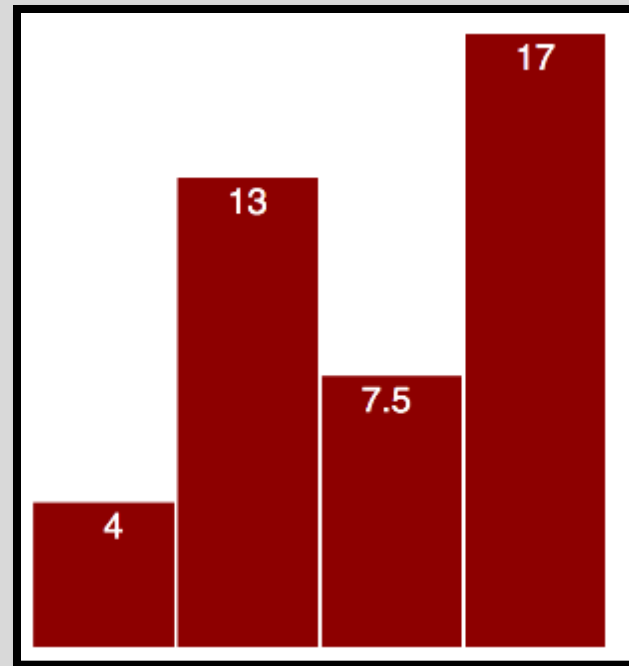
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

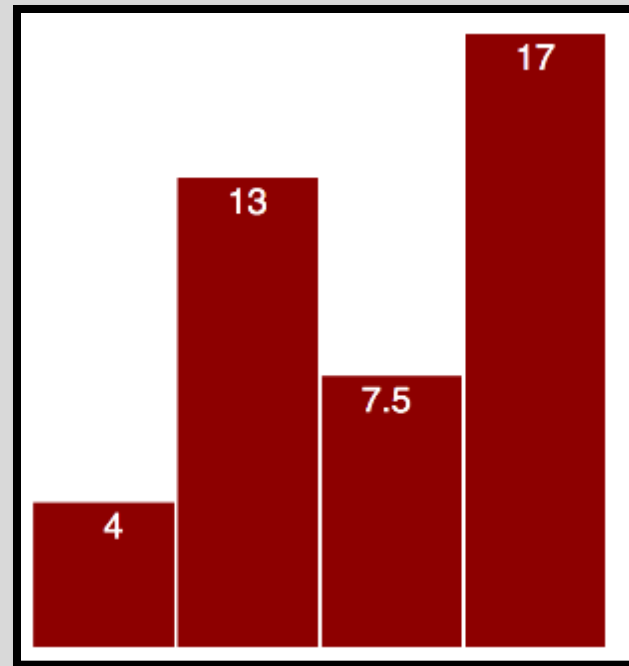
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

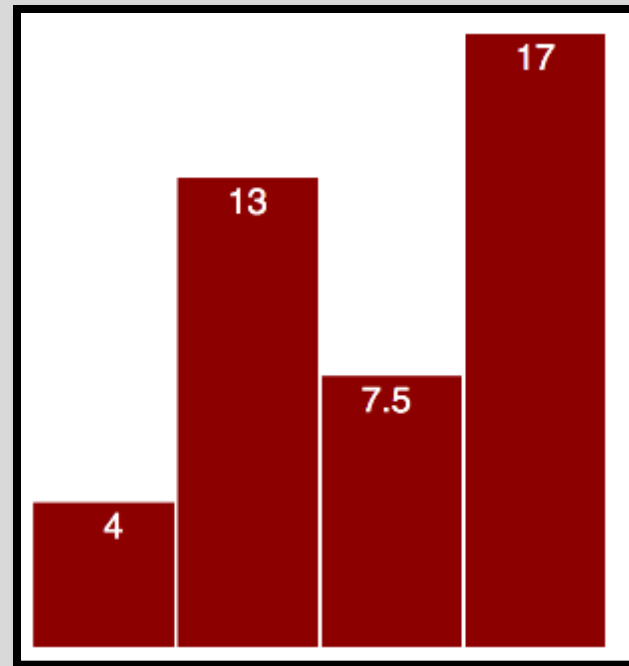
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

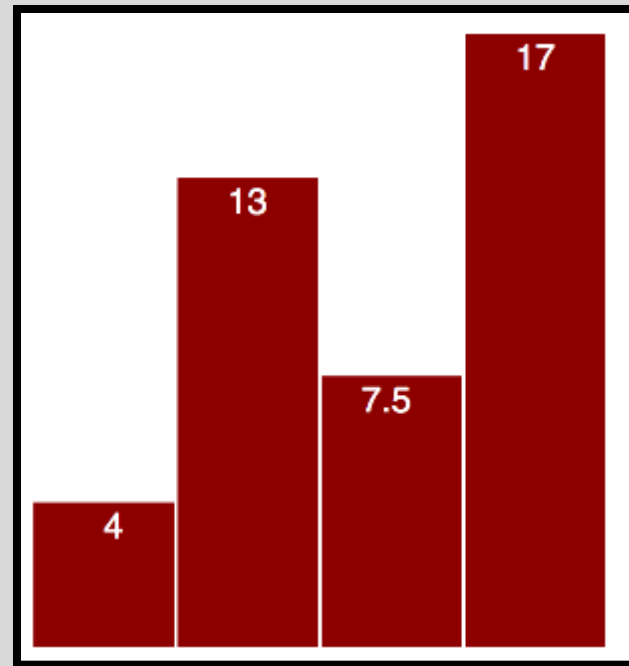
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

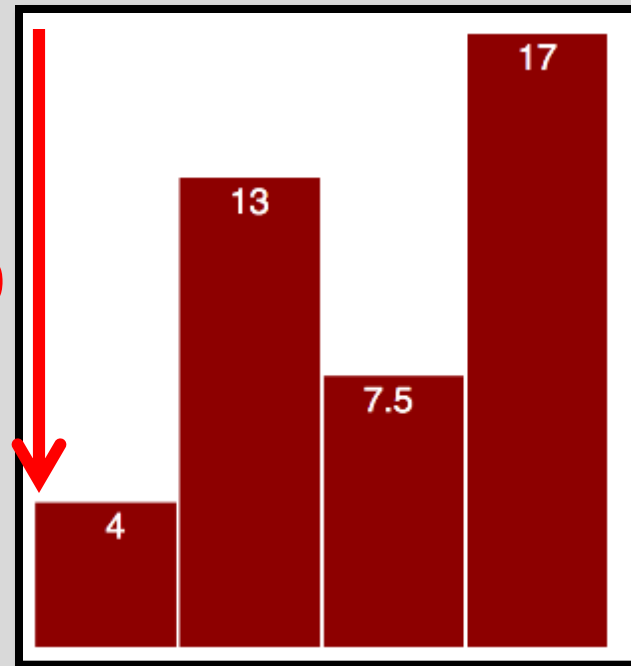
```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

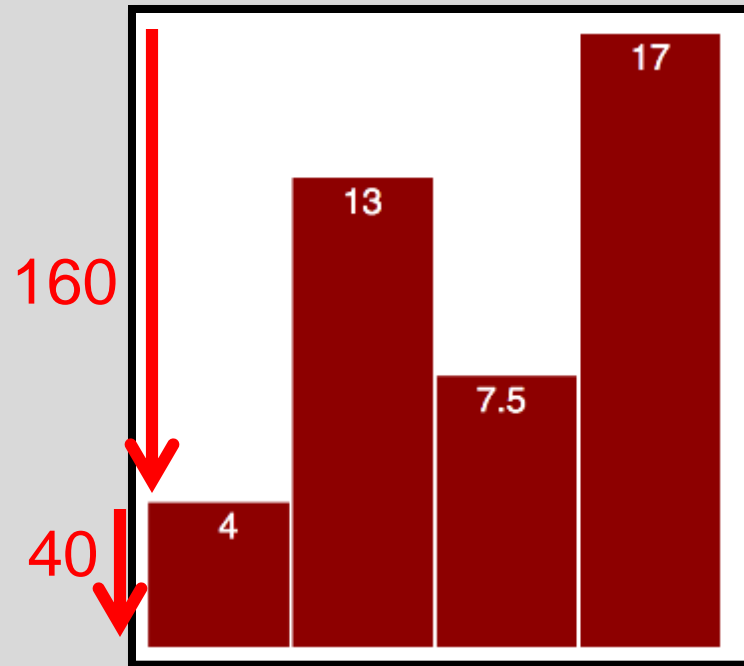
```

160



```
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
```

```
<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>
```



```

<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

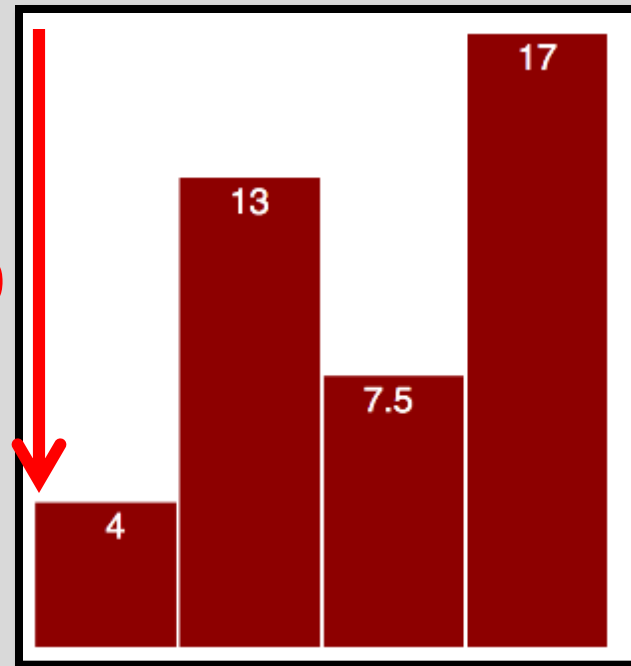
```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```

160




```

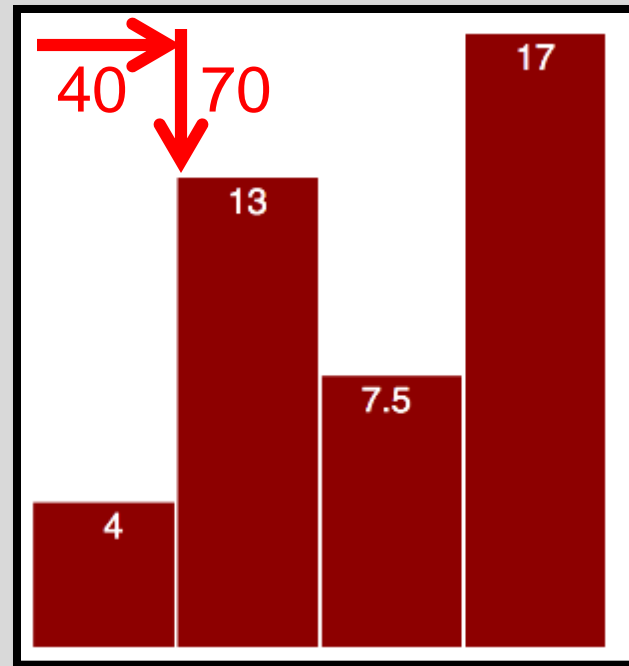
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

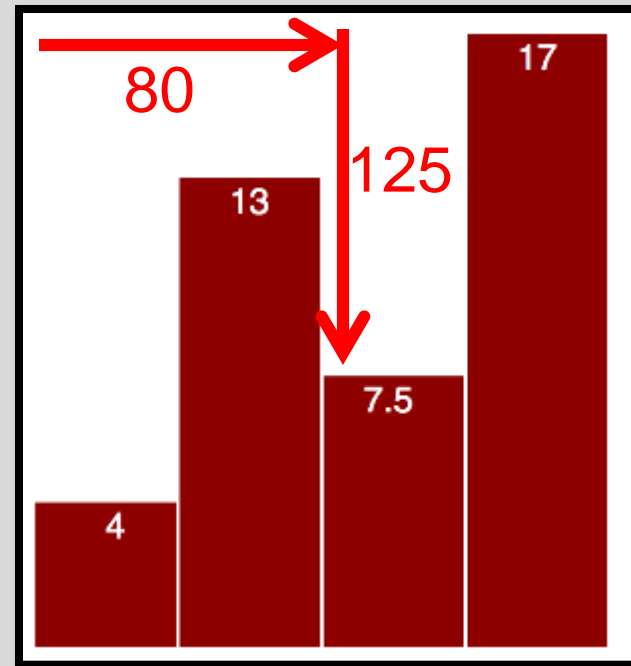
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

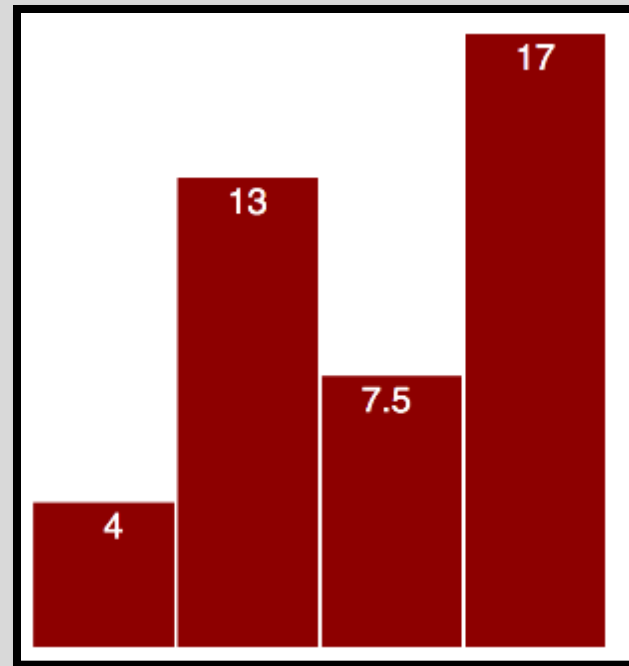
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



```

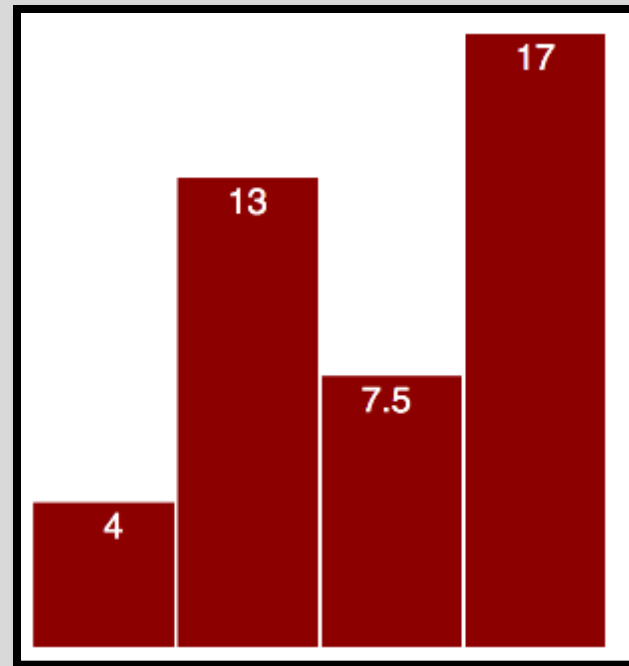
<style>
rect {
  fill: darkred;
}
rect:hover {
  fill: darkblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>

```

```

<svg class="chart" height="200">
  <g transform="translate(0,160)">
    <rect width="39" height="40"></rect><text x="25" y="10">4</text>
  </g>
  <g transform="translate(40,70)">
    <rect width="39" height="130"></rect><text x="25" y="10">13</text>
  </g>
  <g transform="translate(80,125)">
    <rect width="39" height="75"></rect><text x="25" y="10">7.5</text>
  </g>
  <g transform="translate(120,30)">
    <rect width="39" height="170"></rect><text x="25" y="10">17</text>
  </g>
</svg>

```



Dynamic SVG with D3.js

- D3.js is specifically designed to allow us to manipulate HTML/SVG elements based on data
- We can dynamically render SVG elements by applying functionality to a set of data

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```



```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter() .append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```



```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

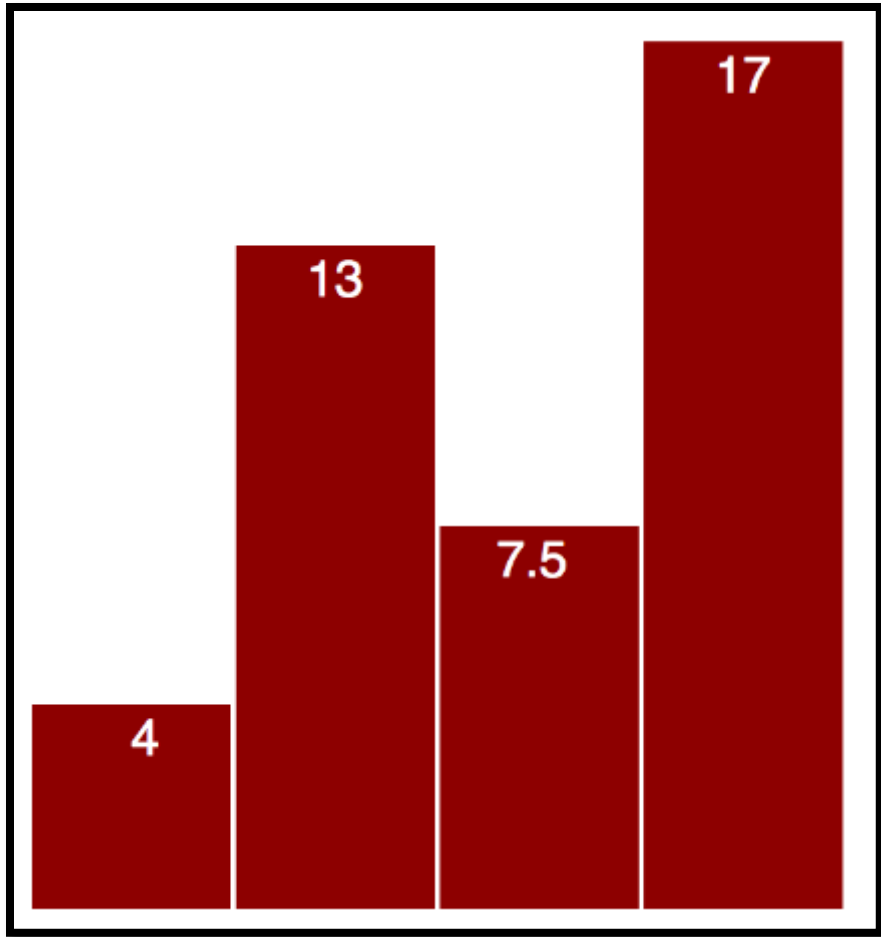
<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

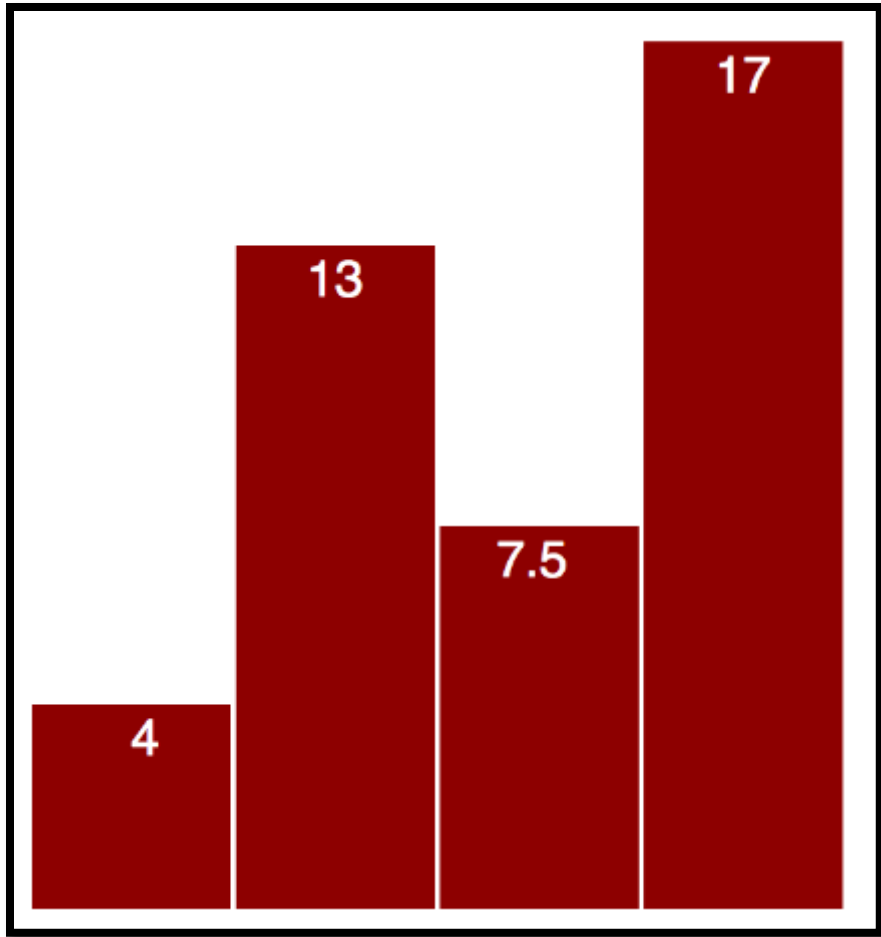
  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

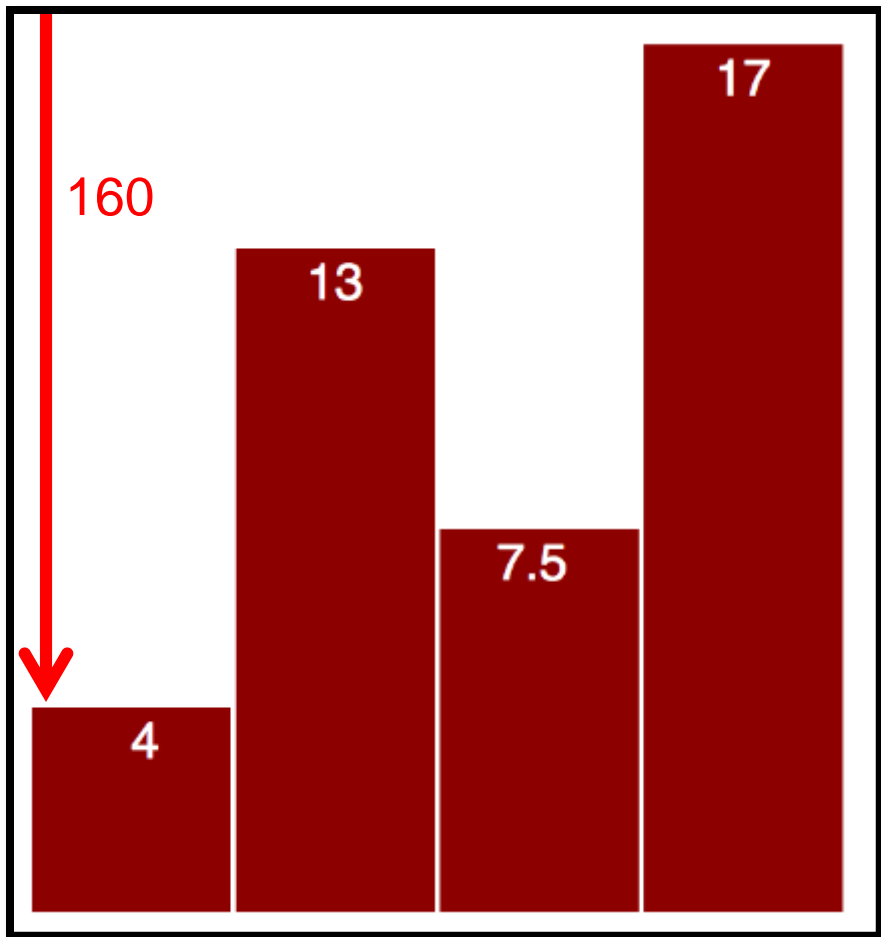
  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```



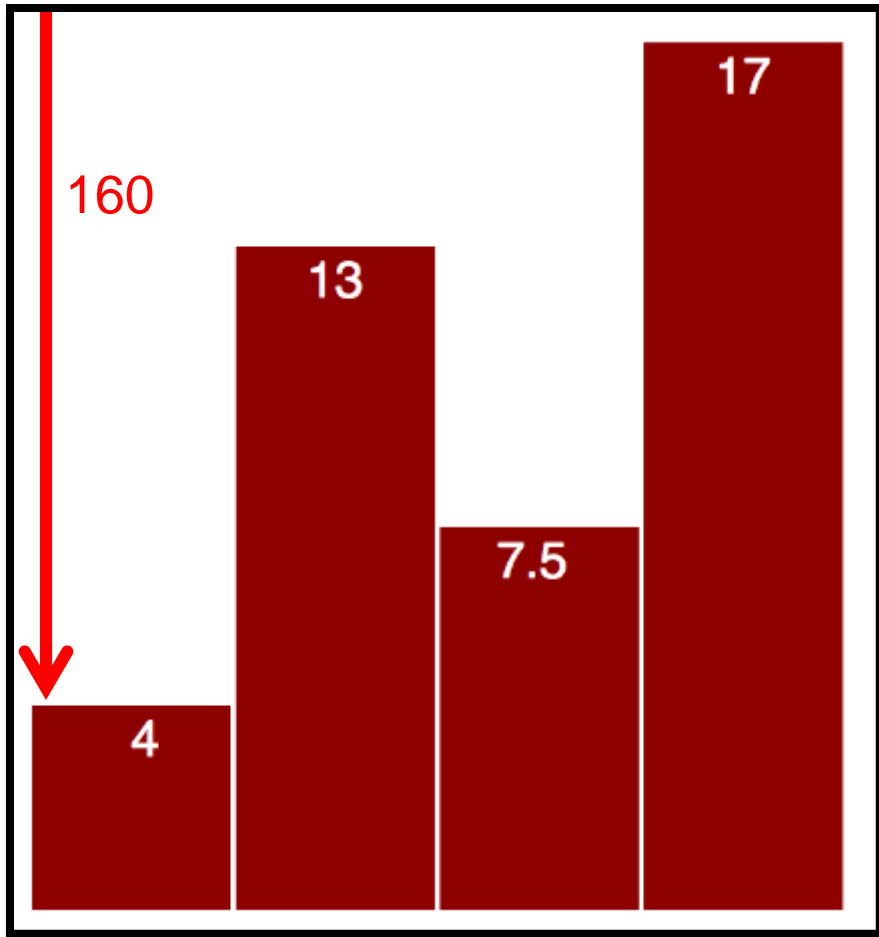
```
var numbers = [40, 130, 75, 170];
```



```
var numbers = [40, 130, 75, 170];
```

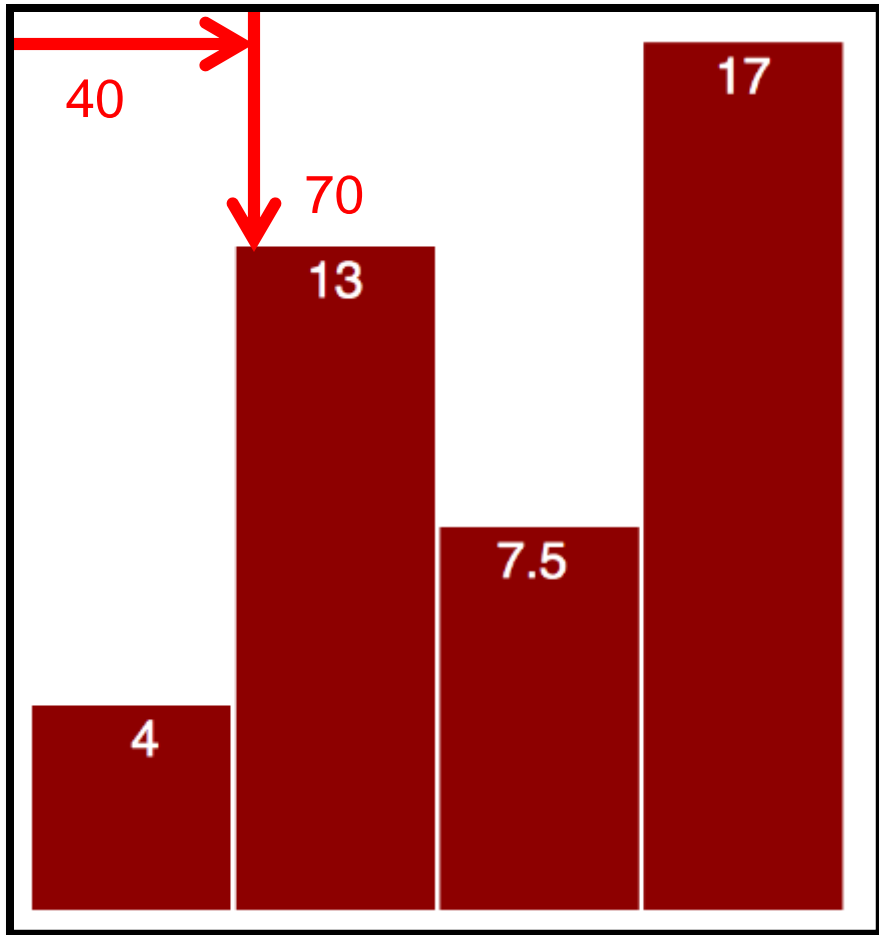


```
var numbers = [40, 130, 75, 170];
```



```
var numbers = [40, 130, 75, 170];
```

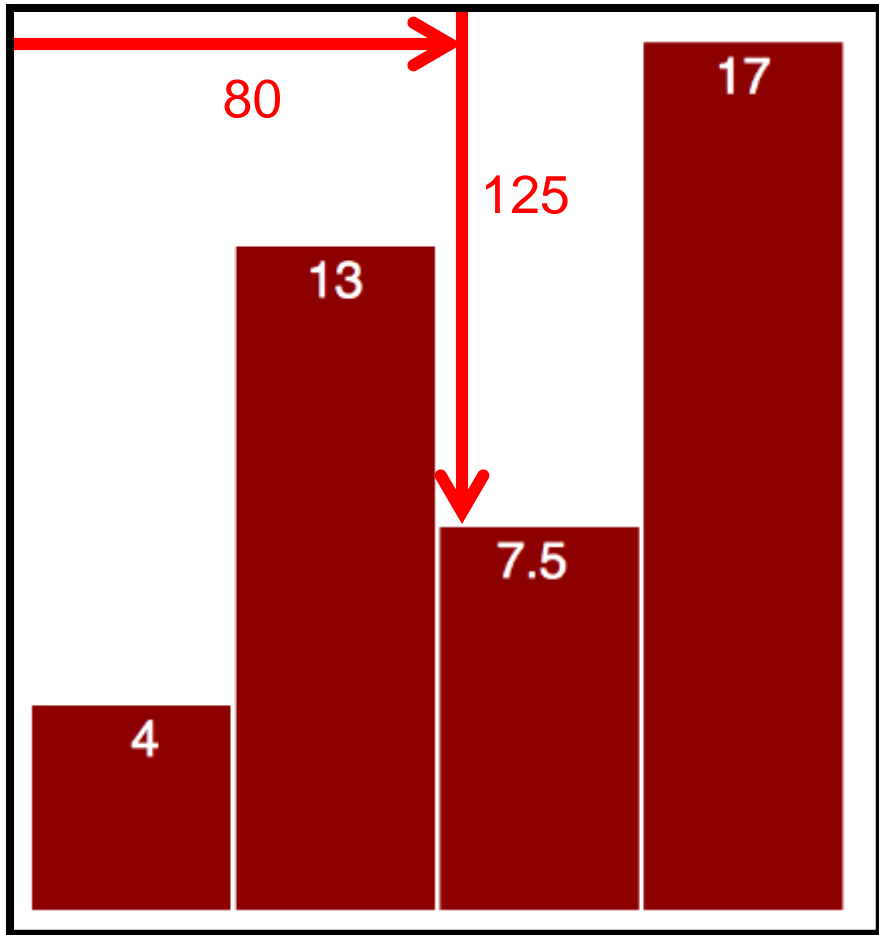
$$d = 40, i = 0 \rightarrow x = 0, y = 160$$



```
var numbers = [40, 130, 75, 170];
```

$d = 40, i = 0 \rightarrow x = 0, y = 160$

$d = 130, i = 1 \rightarrow x = 40, y = 70$

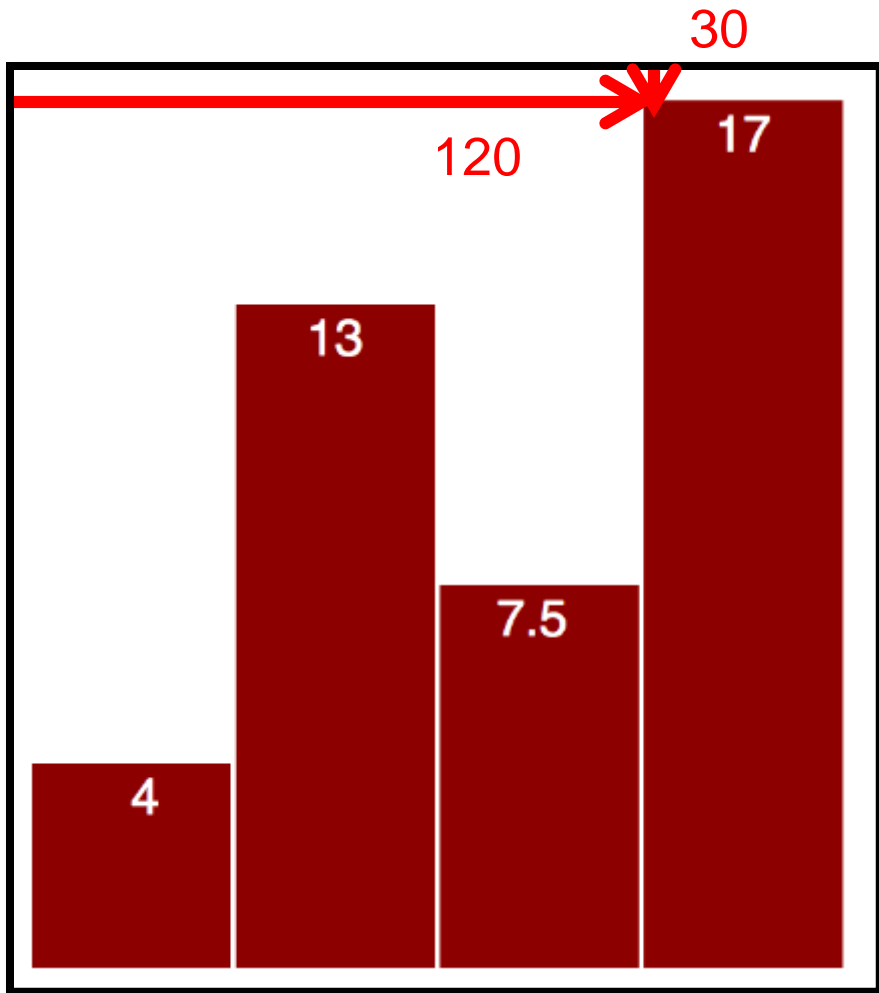


```
var numbers = [40, 130, 75, 170];
```

$d = 40, i = 0 \rightarrow x = 0, y = 160$

$d = 130, i = 1 \rightarrow x = 40, y = 70$

$d = 75, i = 2 \rightarrow x = 80, y = 125$



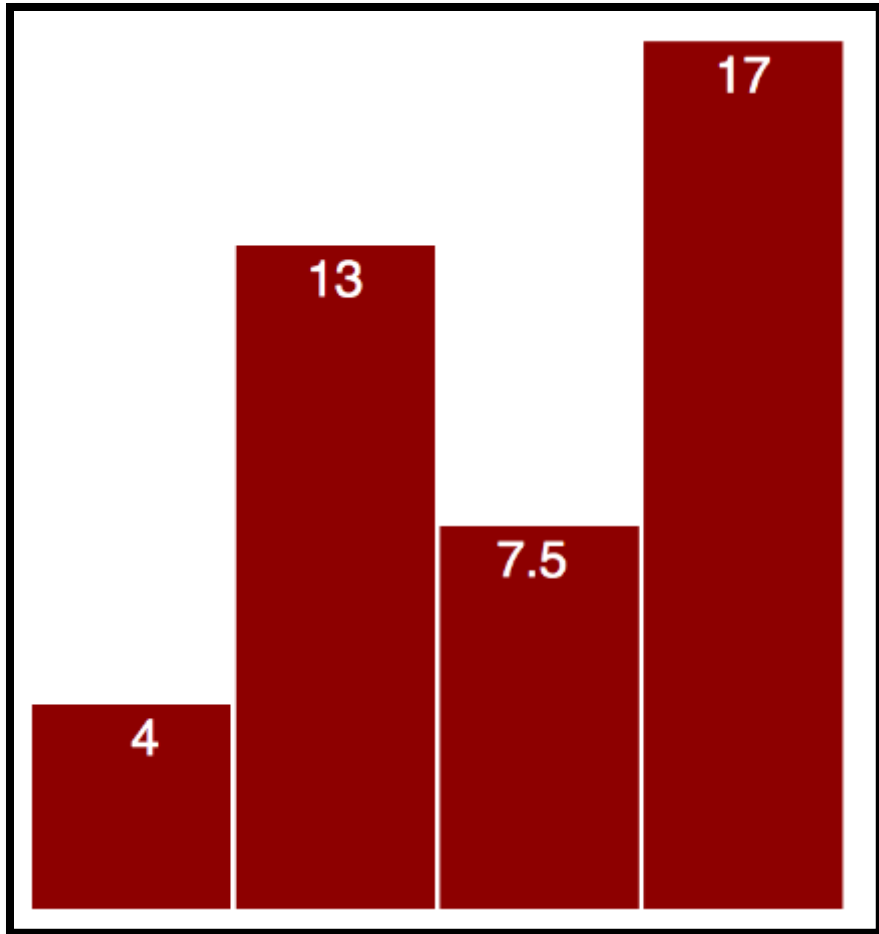
```
var numbers = [40, 130, 75, 170];
```

$d = 40, i = 0 \rightarrow x = 0, y = 160$

$d = 130, i = 1 \rightarrow x = 40, y = 70$

$d = 75, i = 2 \rightarrow x = 80, y = 125$

$d = 170, i = 3 \rightarrow x = 120, y = 30$



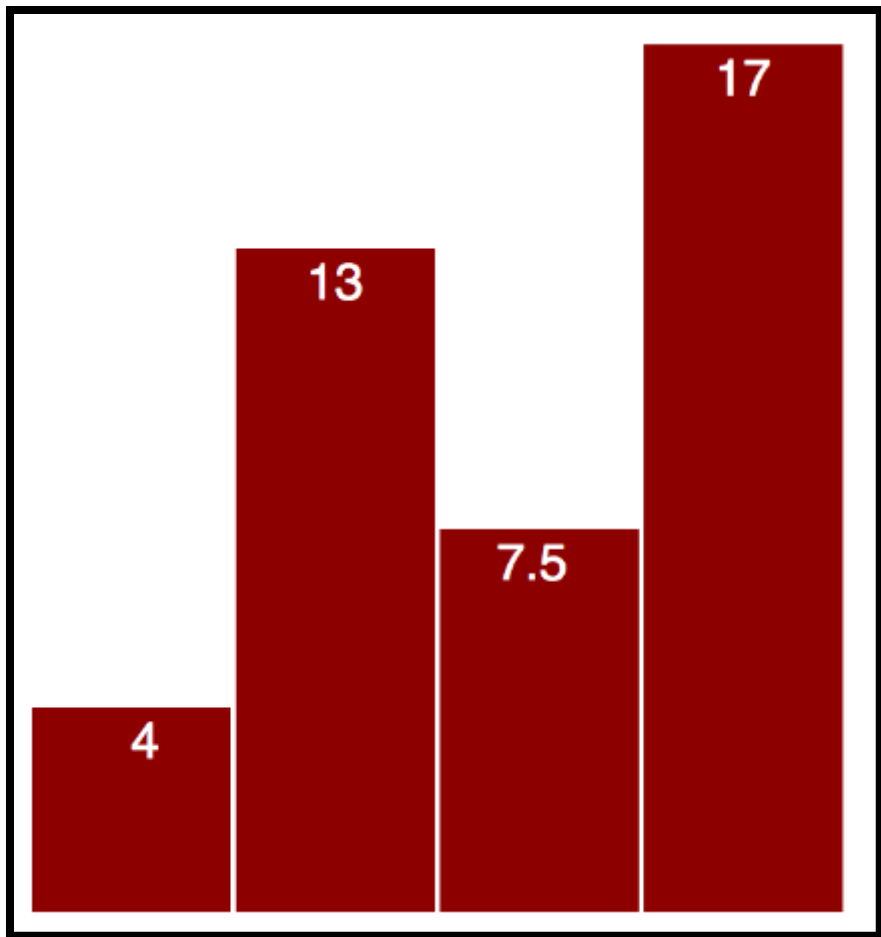
```
var numbers = [40, 130, 75, 170];
```

$$d = 40, i = 0 \rightarrow x = 0, y = 160$$

$$d = 130, i = 1 \rightarrow x = 40, y = 70$$

$$d = 75, i = 2 \rightarrow x = 80, y = 125$$

$$d = 170, i = 3 \rightarrow x = 120, y = 30$$



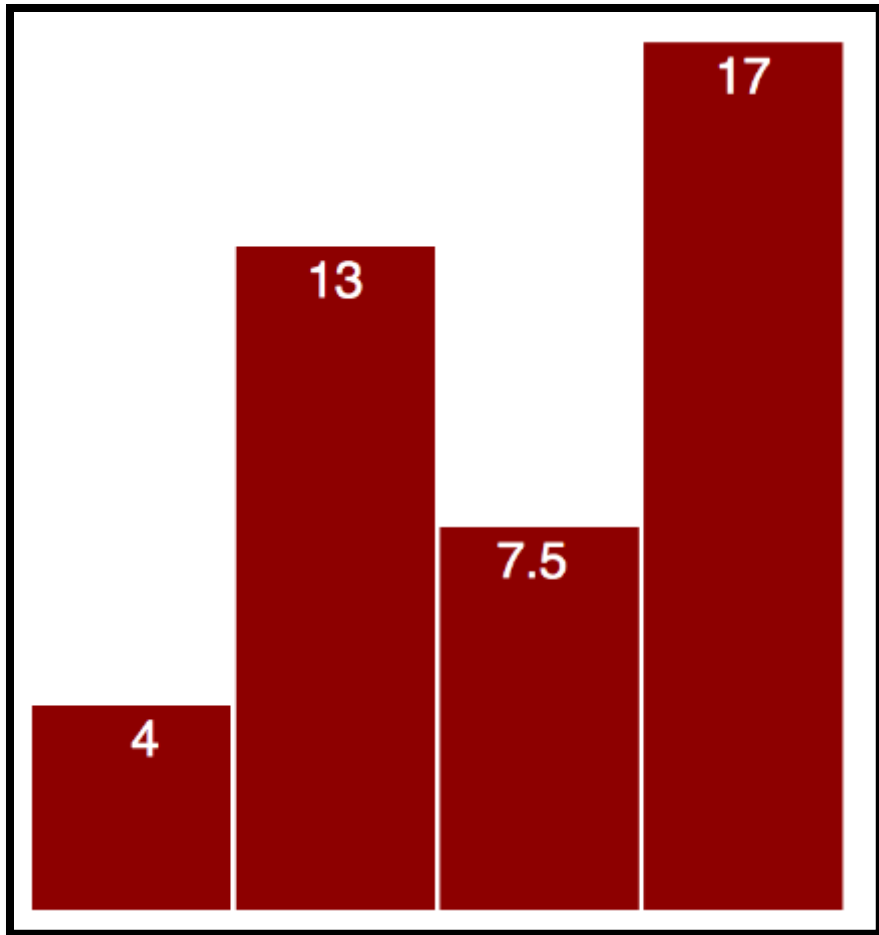
```
var numbers = [40, 130, 75, 170];
```

$$d = 40, i = 0 \rightarrow x = 0, y = 160$$

$$d = 130, i = 1 \rightarrow x = 40, y = 70$$

$$d = 75, i = 2 \rightarrow x = 80, y = 125$$

$$d = 170, i = 3 \rightarrow x = 120, y = 30$$



```
var numbers = [40, 130, 75, 170];
```

$$d = 40, i = 0 \rightarrow x = 0, y = 160$$

$$d = 130, i = 1 \rightarrow x = 40, y = 70$$

$$d = 75, i = 2 \rightarrow x = 80, y = 125$$

$$d = 170, i = 3 \rightarrow x = 120, y = 30$$

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```



```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

```
<style>
<!-- same CSS as before -->
</style>

<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>

<script>
  var numbers = [40, 130, 75, 170];
  var svg = d3.select("svg");

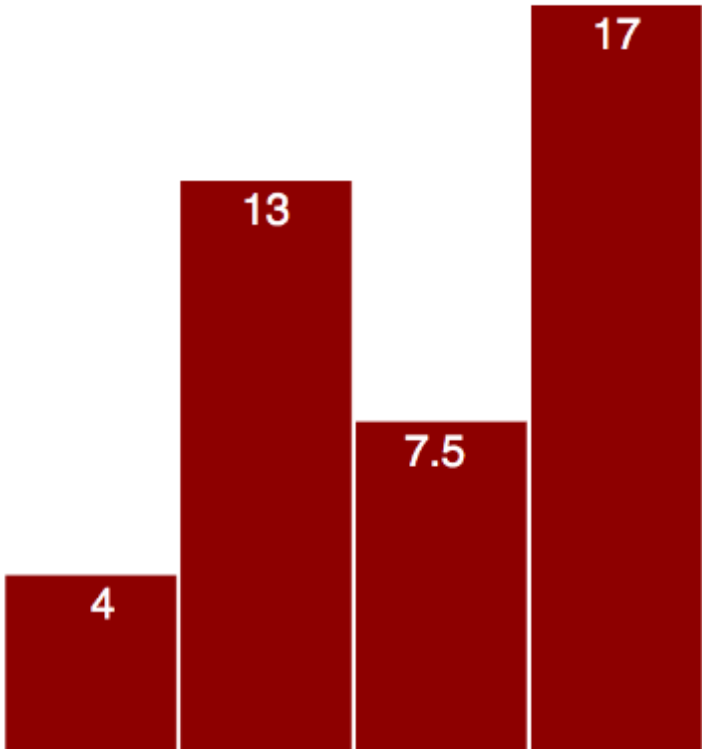
  var selection = svg.selectAll("g")
    .data(numbers)
    .enter().append("g")
    .attr("transform", (d,i) => {
      return "translate(" + 40*i + ", " + (200-d) + ")"; });

  selection.append("rect")
    .attr("width", 39)
    .attr("height", (d,i) => { return d; });

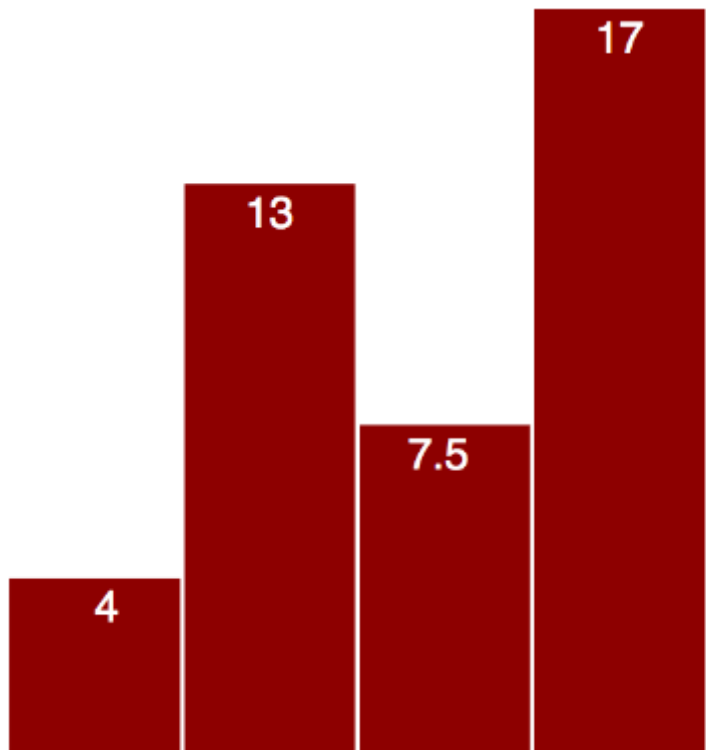
  selection.append("text")
    .attr("x", 25)
    .attr("y", 25)
    .text((d) => { return d/10; });
</script>
```

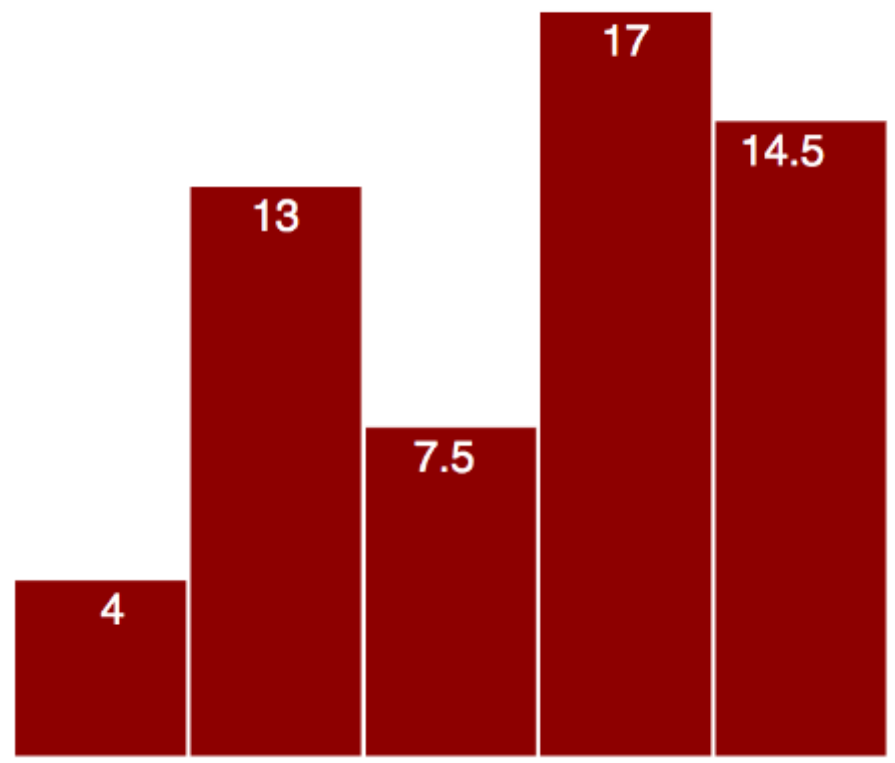
Dynamic Graphics with D3.js

- In addition to rendering HTML/SVG elements using JavaScript, D3.js also allows us to add elements based on any changes to the data
- This allows us to dynamically modify our chart and other graphics

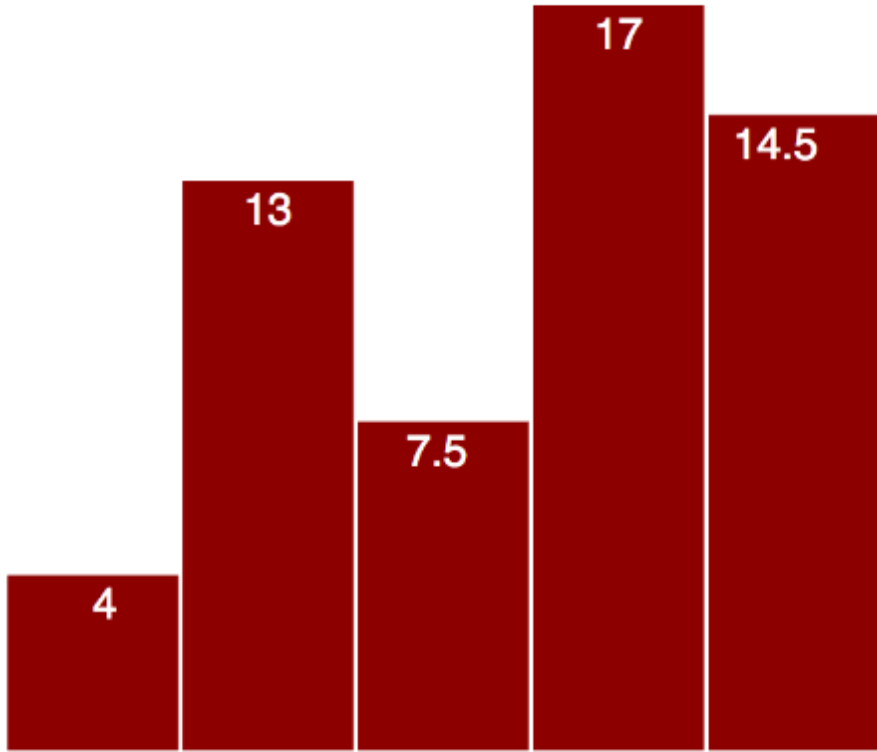


Insert

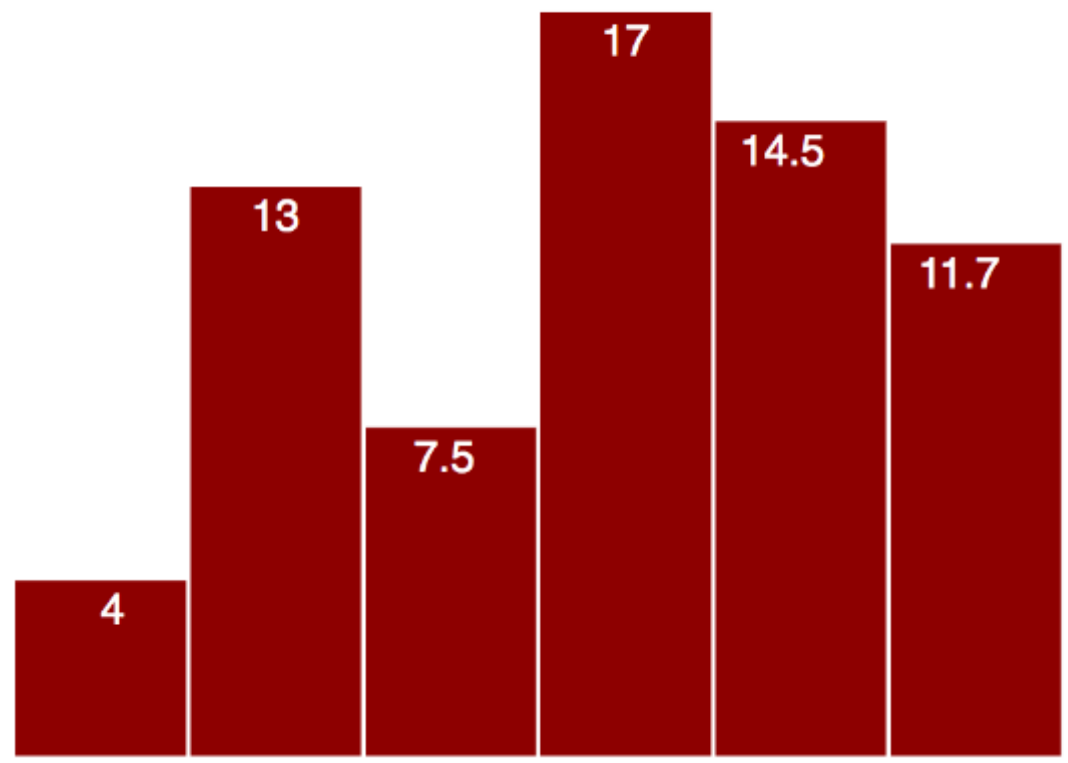




Insert



Insert



Insert

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>
```

```
<svg class="chart" height="200"></svg>
```

```
<p>
```

```
<input id="inputField"></input>
```

```
<button onclick="insert();">Insert</button>
```

```
<script>
```

```
  var numbers = [40, 130, 75, 170];
```

```
  function insert() {
```

```
    var value = document.getElementById('inputField').value;
```

```
    numbers.push(value);
```

```
    drawChart();
```

```
    document.getElementById('inputField').value = '';
```

```
  }
```

```
  function drawChart() {
```

```
    // same D3 code as before!
```

```
  }
```

```
  drawChart();
```

```
</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>
```

```
<svg class="chart" height="200"></svg>
```

```
<p>
```

```
<input id="inputField"></input>
```

```
<button onclick="insert();">Insert</button>
```

```
<script>
```

```
  var numbers = [40, 130, 75, 170];
```

```
  function insert() {
```

```
    var value = document.getElementById('inputField').value;
```

```
    numbers.push(value);
```

```
    drawChart();
```

```
    document.getElementById('inputField').value = '';
```

```
  }
```

```
  function drawChart() {
```

```
    // same D3 code as before!
```

```
  }
```

```
  drawChart();
```

```
</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```



```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>
```

```
<svg class="chart" height="200"></svg>
```

```
<p>
```

```
<input id="inputField"></input>
```

```
<button onclick="insert();">Insert</button>
```

```
<script>
```

```
var numbers = [40, 130, 75, 170];
```

```
function insert() {
```

```
    var value = document.getElementById('inputField').value;
```

```
    numbers.push(value);
```

```
    drawChart();
```

```
    document.getElementById('inputField').value = '';
```

```
}
```

```
function drawChart() {
```

```
    // same D3 code as before!
```

```
}
```

```
drawChart();
```

```
</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```



```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

```
<script src="http://d3js.org/d3.v4.min.js"></script>

<svg class="chart" height="200"></svg>
<p>
<input id="inputField"></input>
<button onclick="insert();">Insert</button>

<script>
  var numbers = [40, 130, 75, 170];

  function insert() {
    var value = document.getElementById('inputField').value;
    numbers.push(value);
    drawChart();
    document.getElementById('inputField').value = '';
  }

  function drawChart() {
    // same D3 code as before!
  }

  drawChart();

</script>
```

Summary

- D3.js is a powerful library for generating HTML and SVG elements based on data
- We can apply functions to data sets to generate graphical elements, e.g. charts
- And use D3.js to modify the elements when new data is added



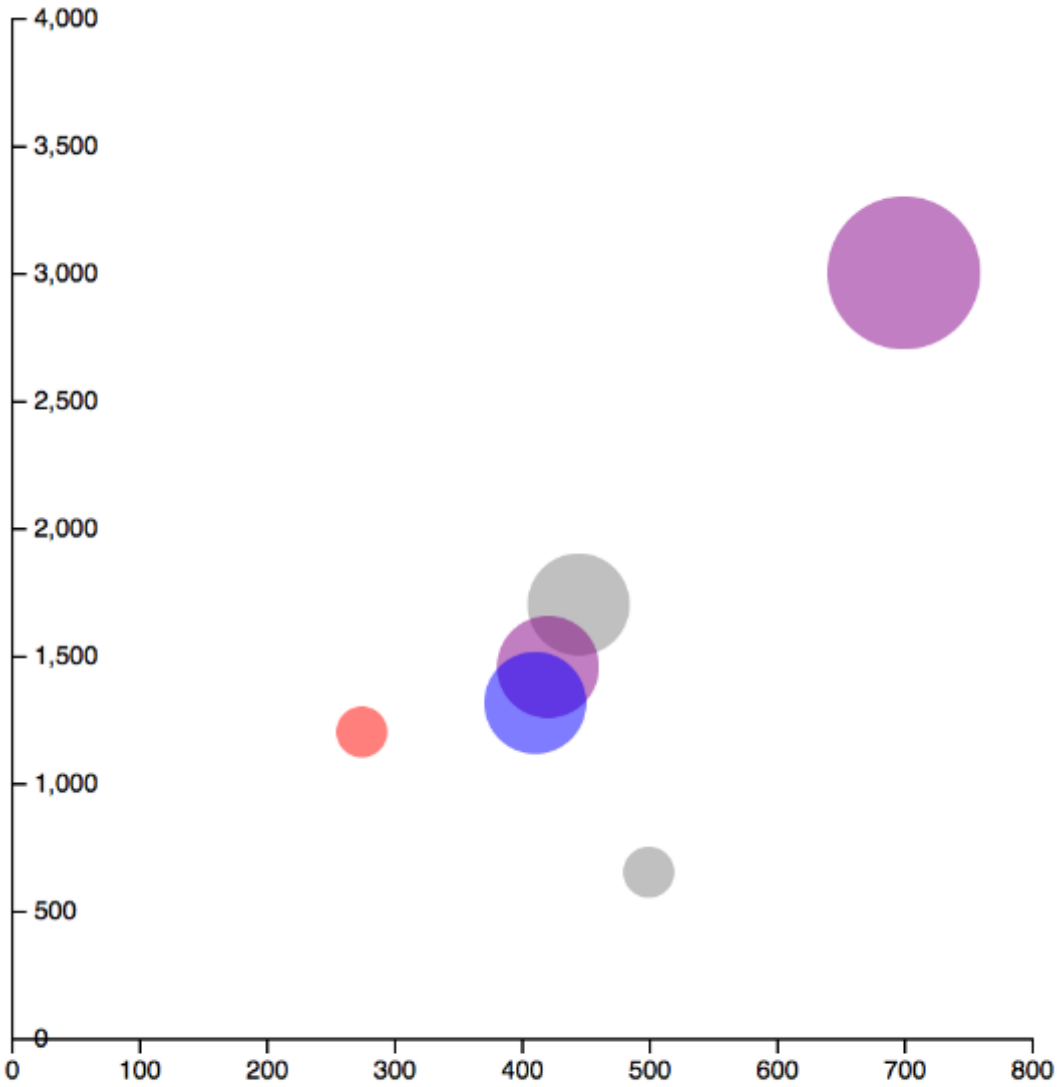
Video 3.11

More D3

Chris Murphy

Review

- D3.js allows us to generate HTML and SVG elements based on data
- We can apply functions to data sets to generate graphical elements, e.g. charts



D3 and Data

- The data used in D3 can be objects, not just numbers
- We can then use the properties of these objects when deciding how to render the visualization (chart, SVG, etc.)


```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>
```

<script>

```
var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [ ] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [ ] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```



```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```

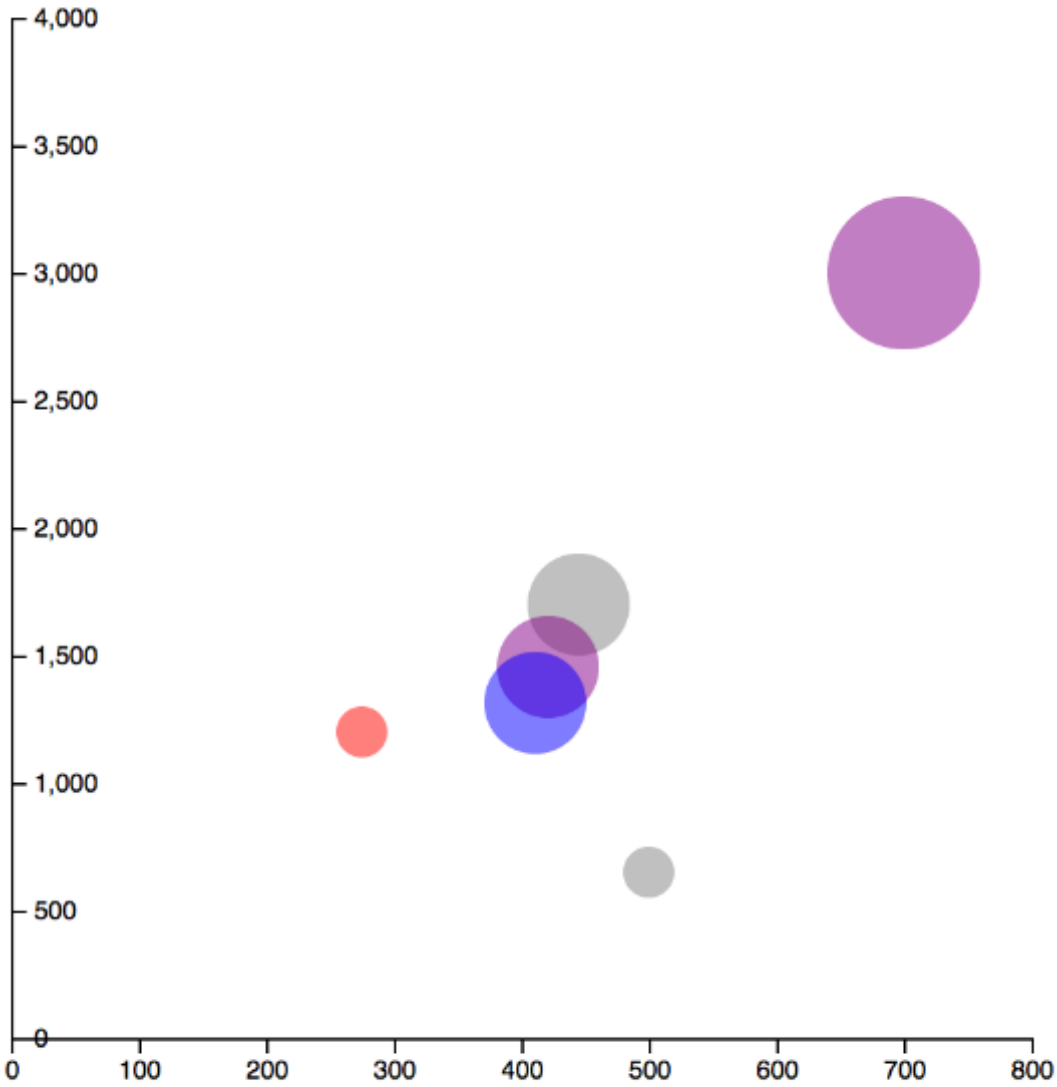
```
<html>
<head>
<script src="http://d3js.org/d3.v4.min.js"></script>
</head>

<body>
<svg class="chart" height="900" width="900">
</svg>

<script>

var values = [
  {price: 700, sqft: 3000, br: 3, pets: [ 'cats', 'dogs' ] },
  {price: 445, sqft: 1700, br: 2, pets: [] },
  {price: 421, sqft: 1455, br: 2, pets: [ 'cats', 'dogs' ] },
  {price: 411, sqft: 1314, br: 2, pets: [ 'dogs' ] },
  {price: 275, sqft: 1200, br: 1, pets: [ 'cats' ]},
  {price: 500, sqft: 650, br: 1, pets: [] },
];

. . .
```



```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
    .data(values)
    .enter()
    .append("g")
    .attr("transform", "translate(10,10)");

selection.append("circle")
    .attr("cx", (d,i) => { return d.price / 2; })
    .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
    .attr("r", (d,i) => { return d.br * 10 ; })
    .style("fill", (d,i) => { return color(d.pets); })
    .style("opacity", "0.5")
    .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
    var dogs = pets.indexOf('dogs') !== -1;
    var cats = pets.indexOf('cats') !== -1;
    if (dogs) return cats ? 'purple' : 'blue' ;
    else return cats ? 'red' : 'gray';
}

function print(home) {
    return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```



```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

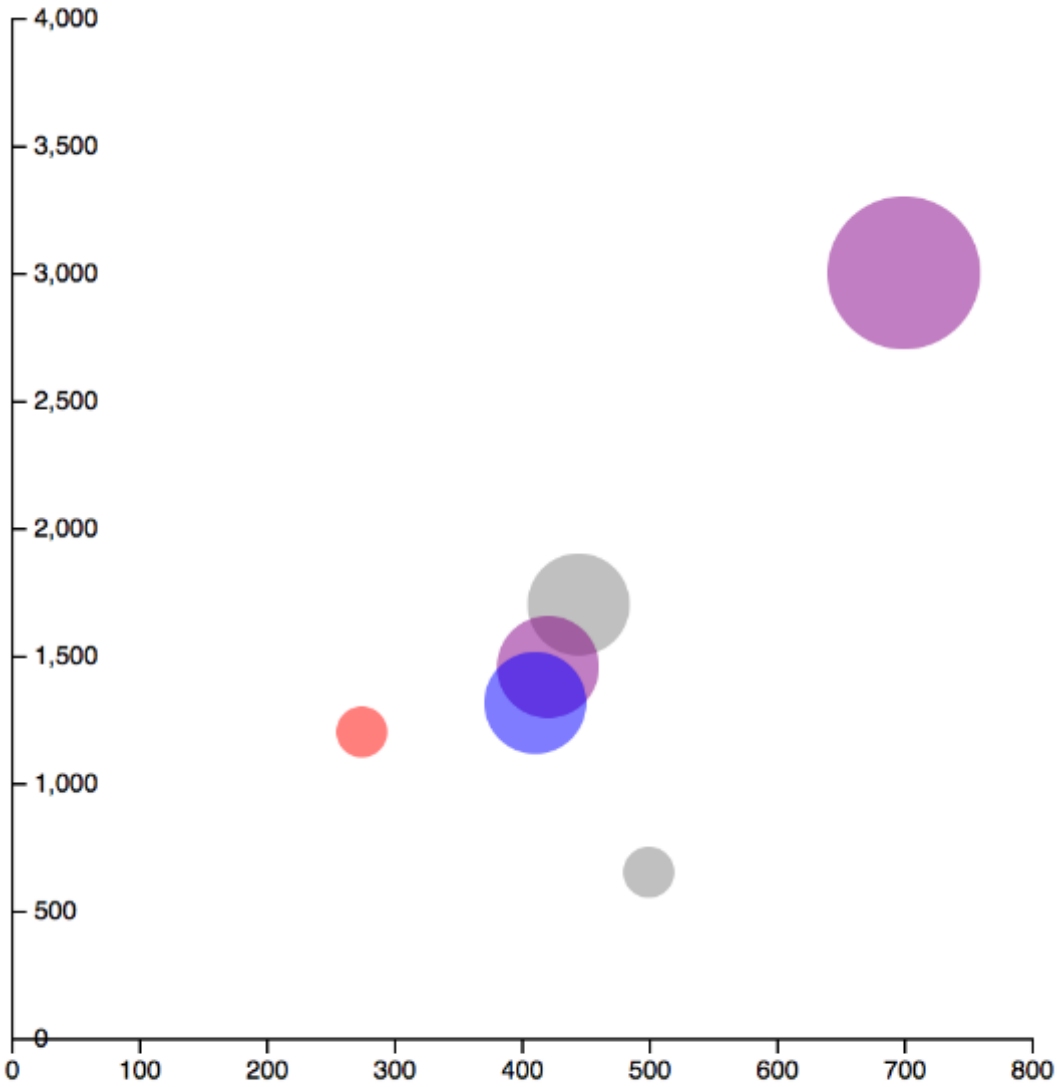
function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```



```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");
```

selection.append("circle")

```
.attr("cx", (d,i) => { return d.price / 2; })
.attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
.attr("r", (d,i) => { return d.br * 10 ; })
.style("fill", (d,i) => { return color(d.pets); })
.style("opacity", "0.5")
.append("svg:title").text( (d,i) => { return print(d); });
```

```
function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}
```

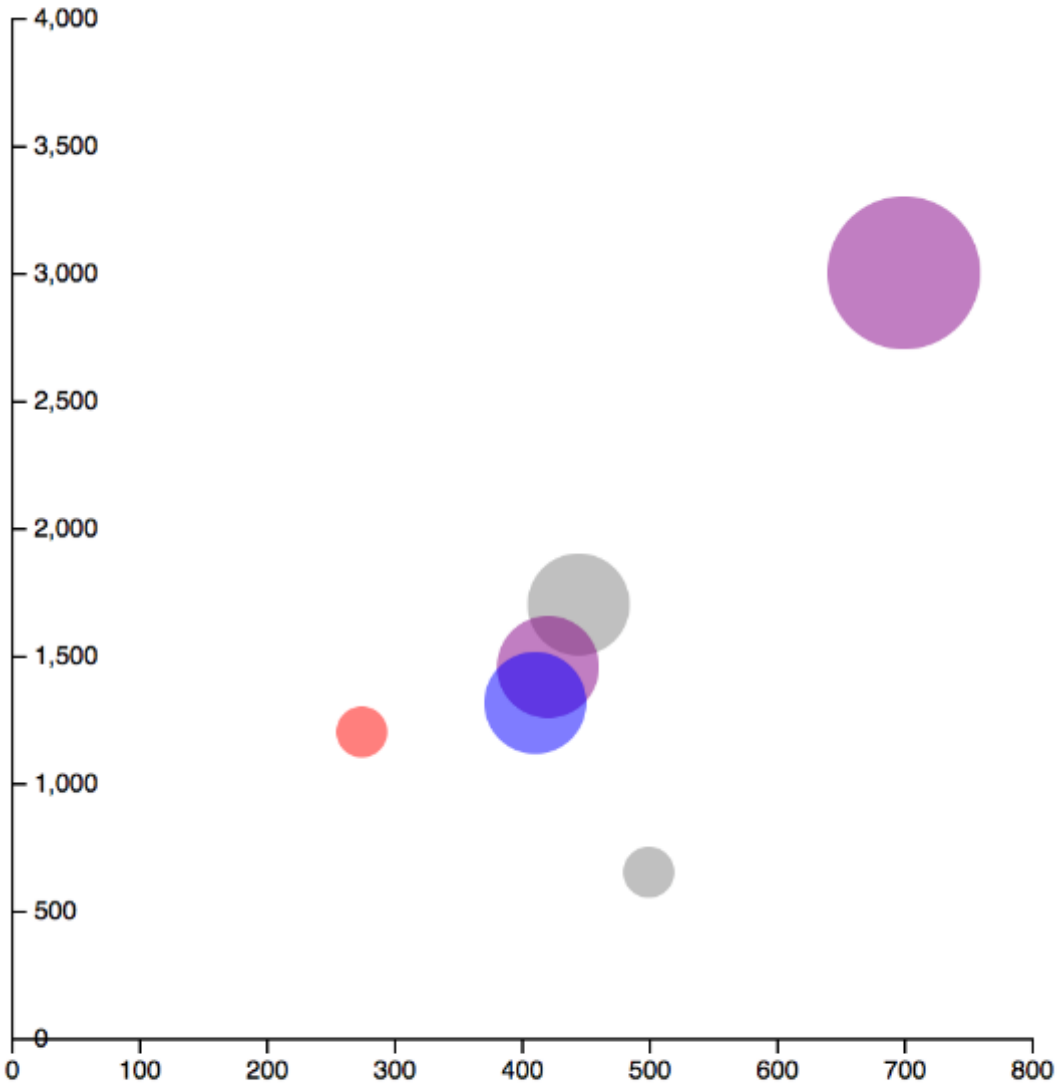
```
function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

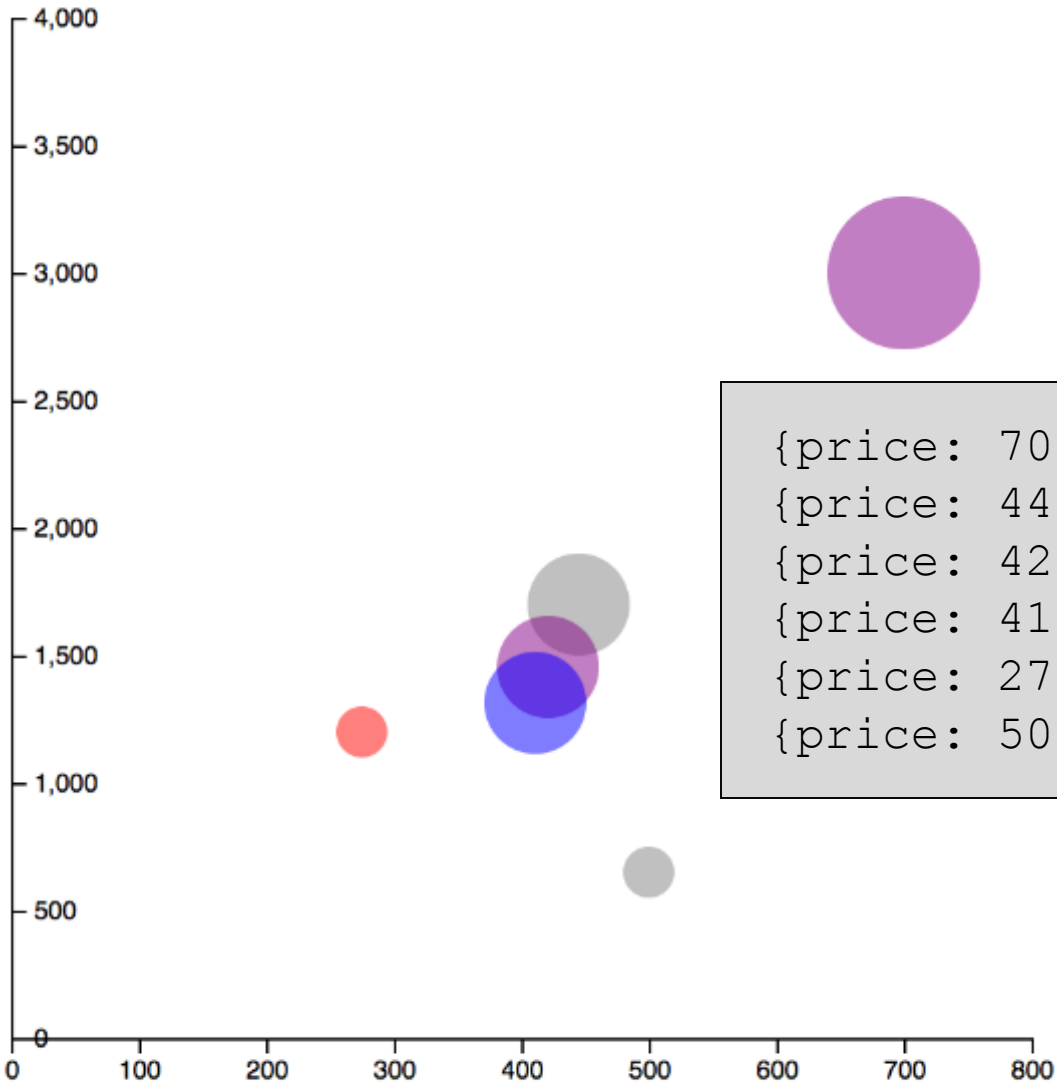
```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

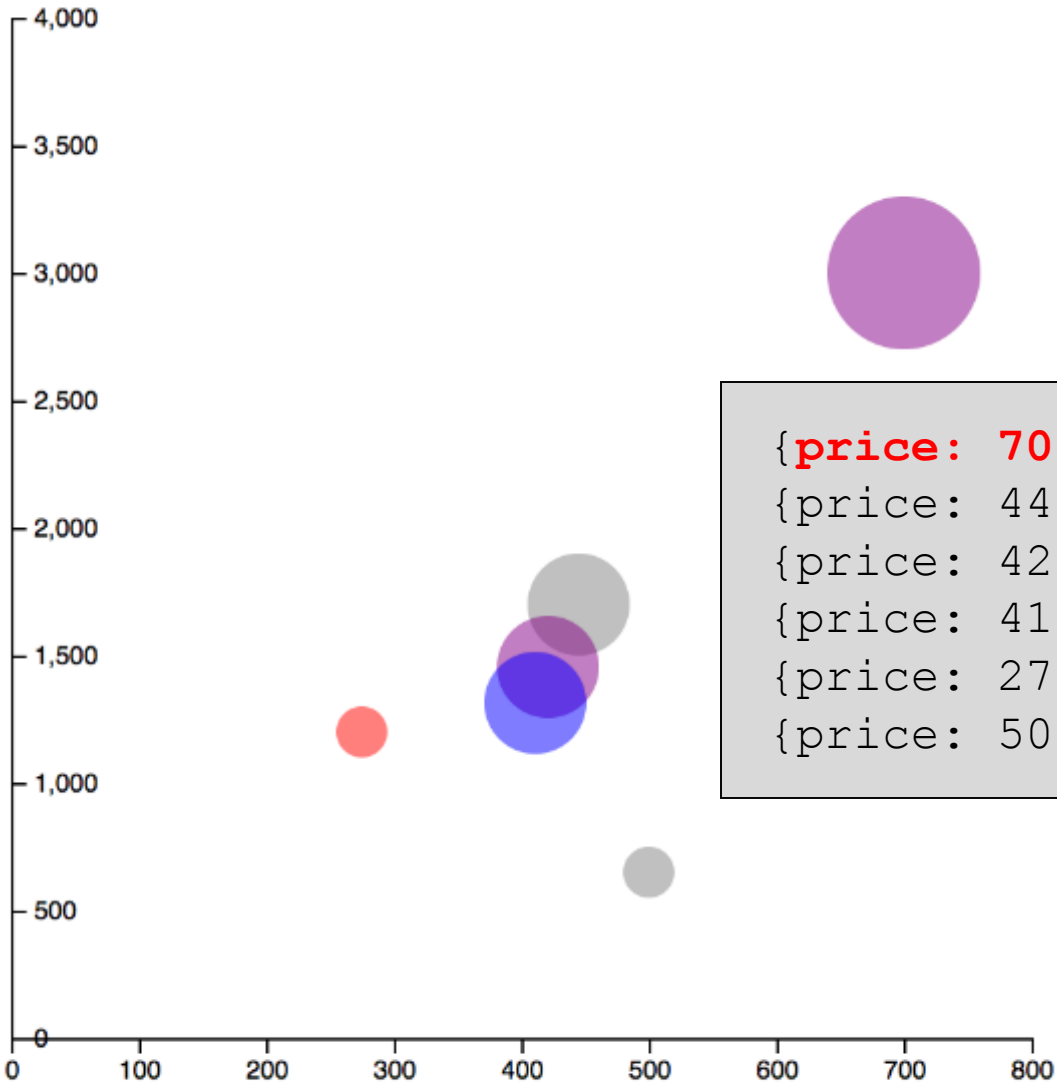
function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

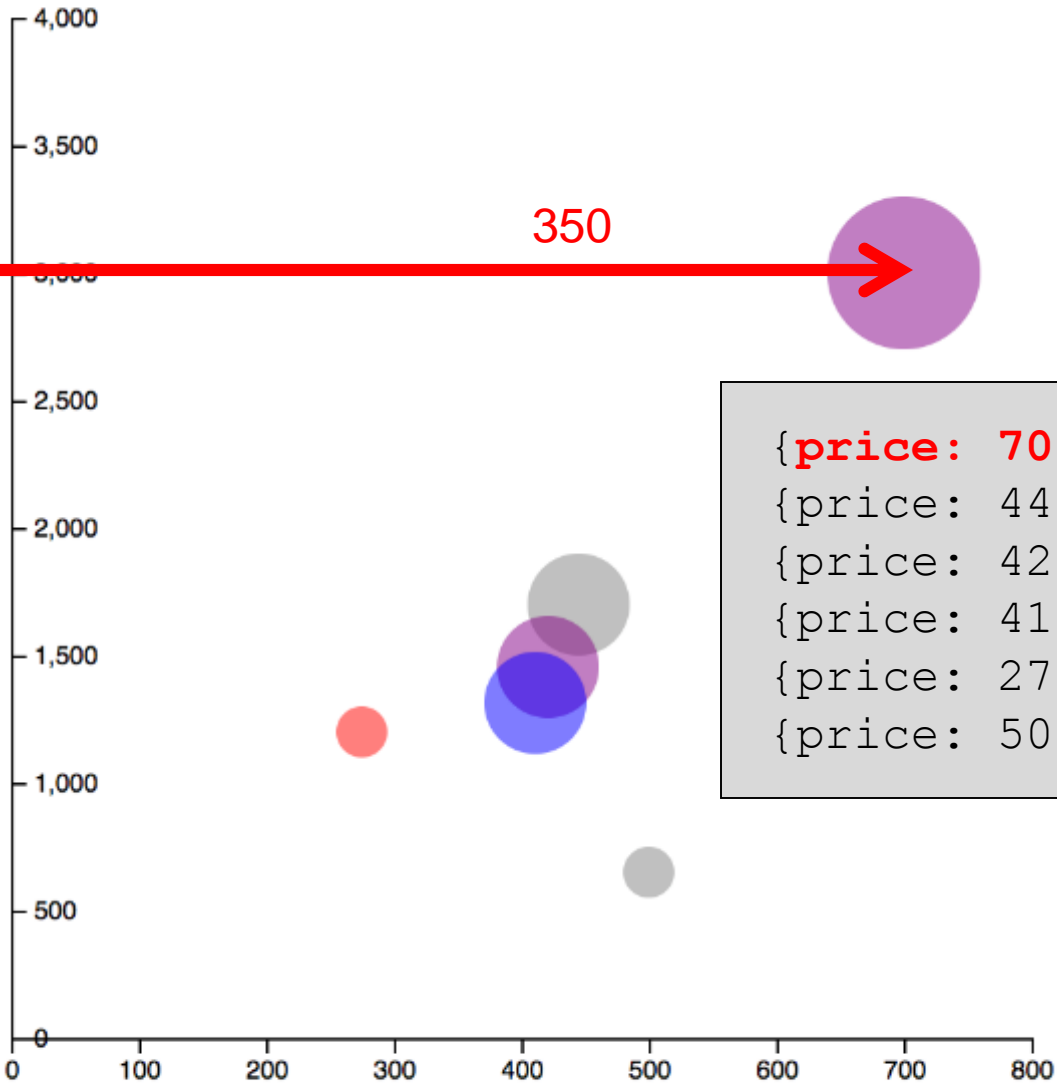




```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```



```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```



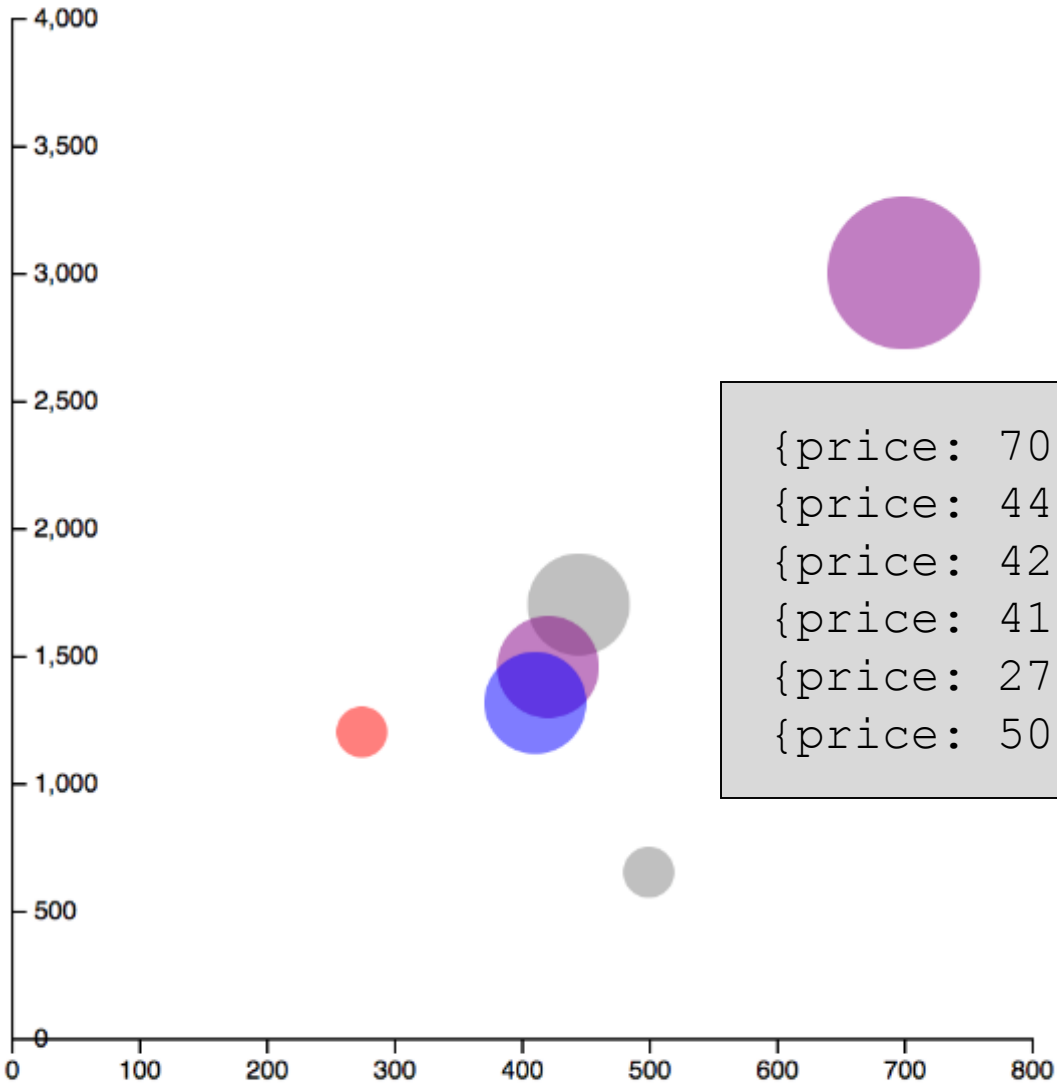
```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

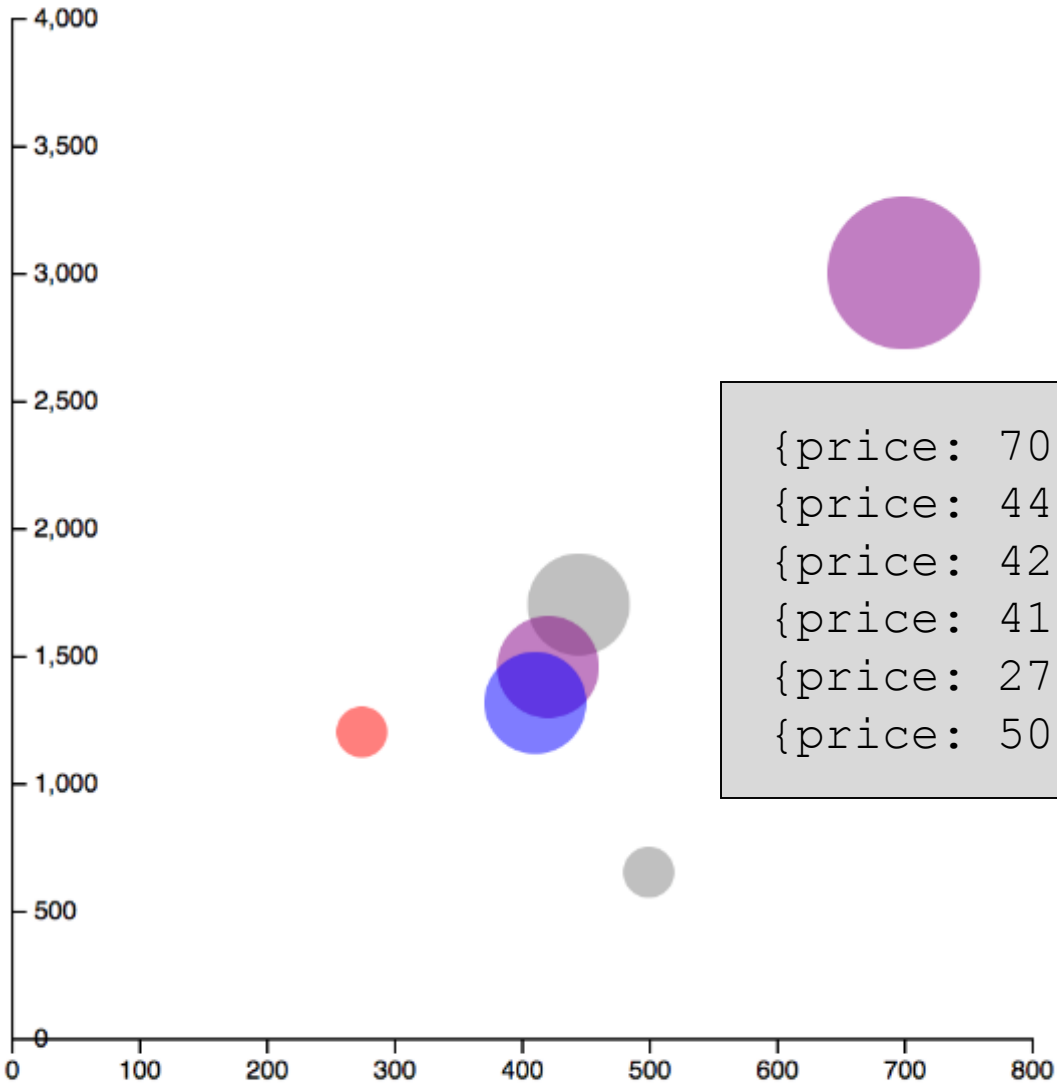
selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft) / (4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

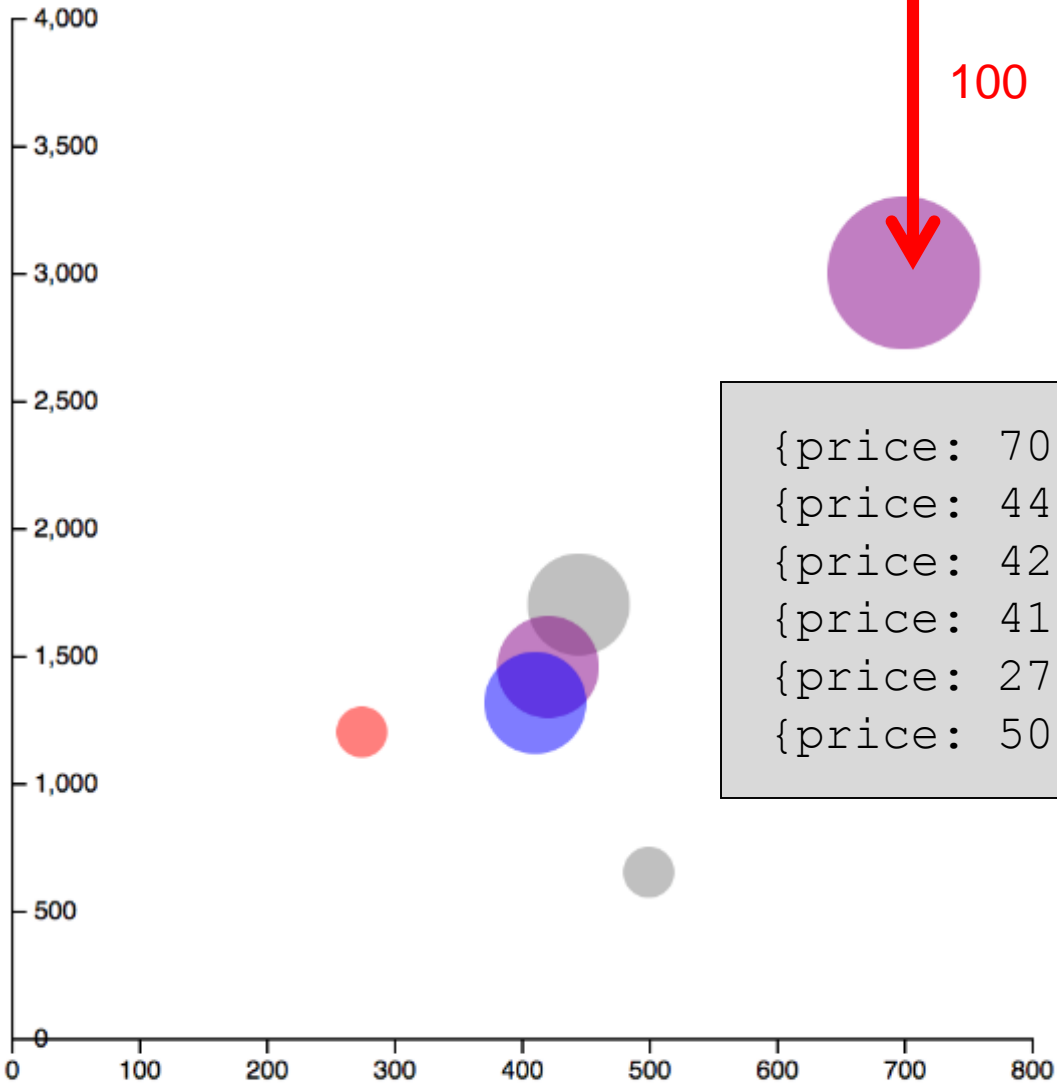
function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```



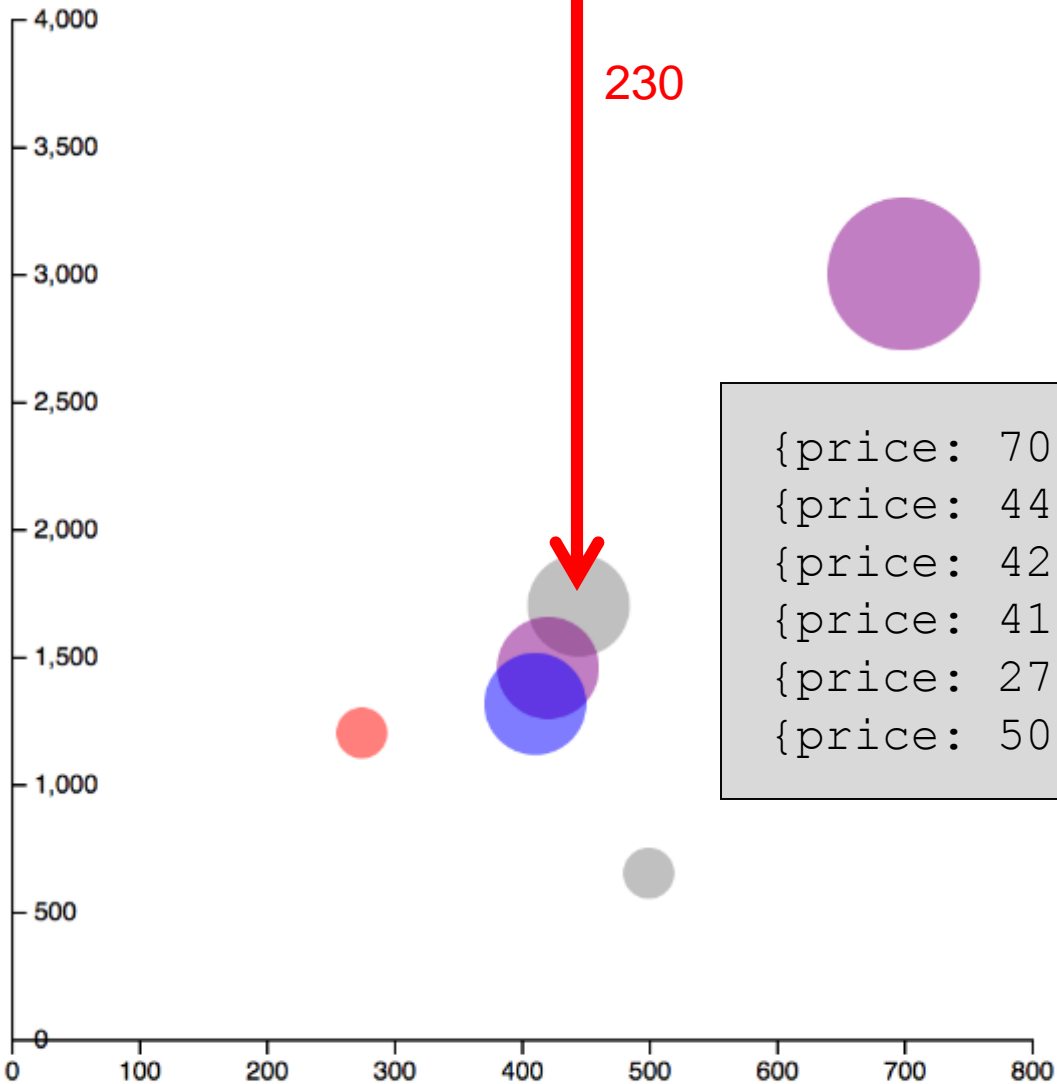
```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```



```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```



```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```



```
{price: 700, sqft: 3000, . . . },  
{price: 445, sqft: 1700, . . . },  
{price: 421, sqft: 1455, . . . },  
{price: 411, sqft: 1314, . . . },  
{price: 275, sqft: 1200, . . . },  
{price: 500, sqft: 650, . . . },
```



```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft) / (4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') != -1;
  var cats = pets.indexOf('cats') != -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') != -1;
  var cats = pets.indexOf('cats') != -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```



```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

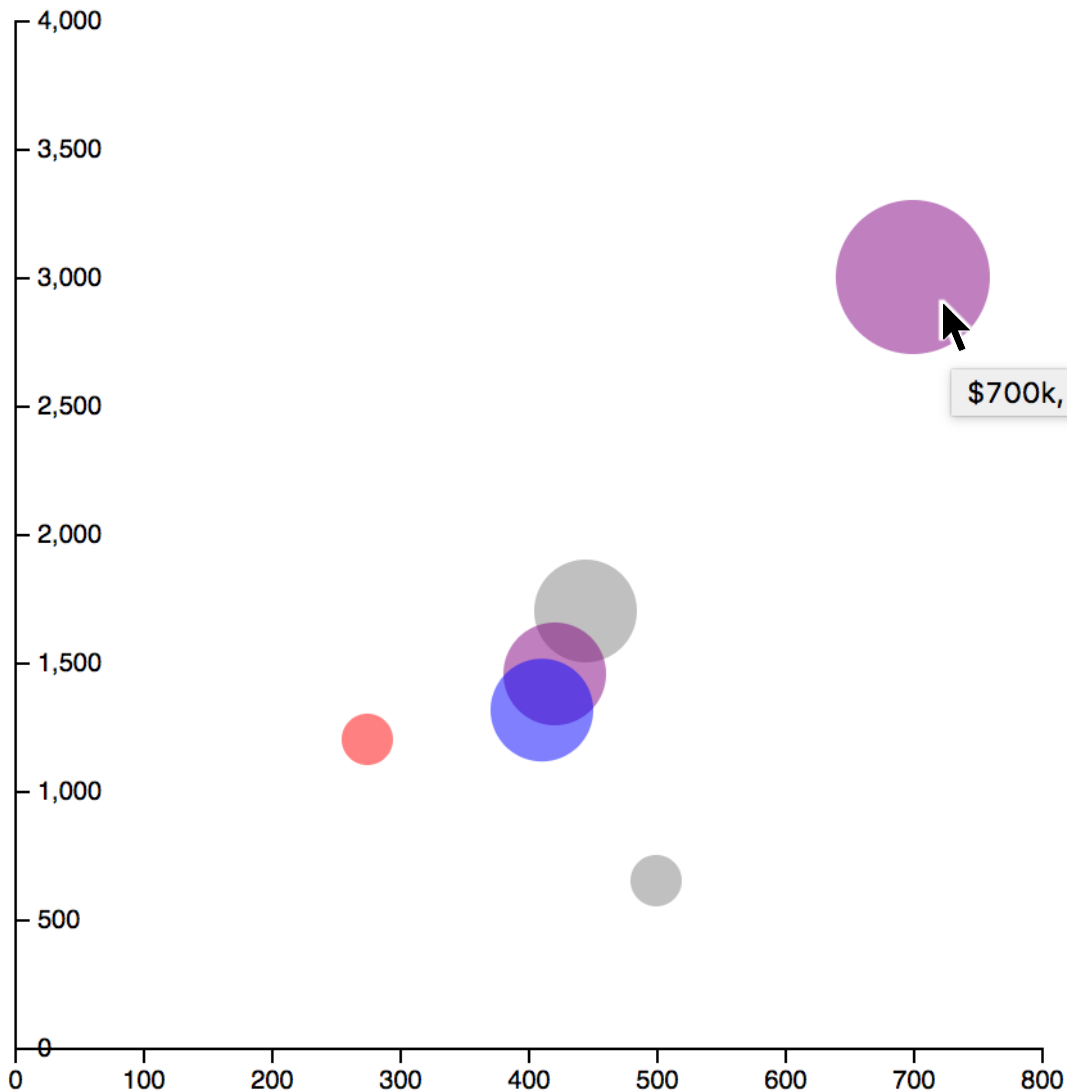
function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```



```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

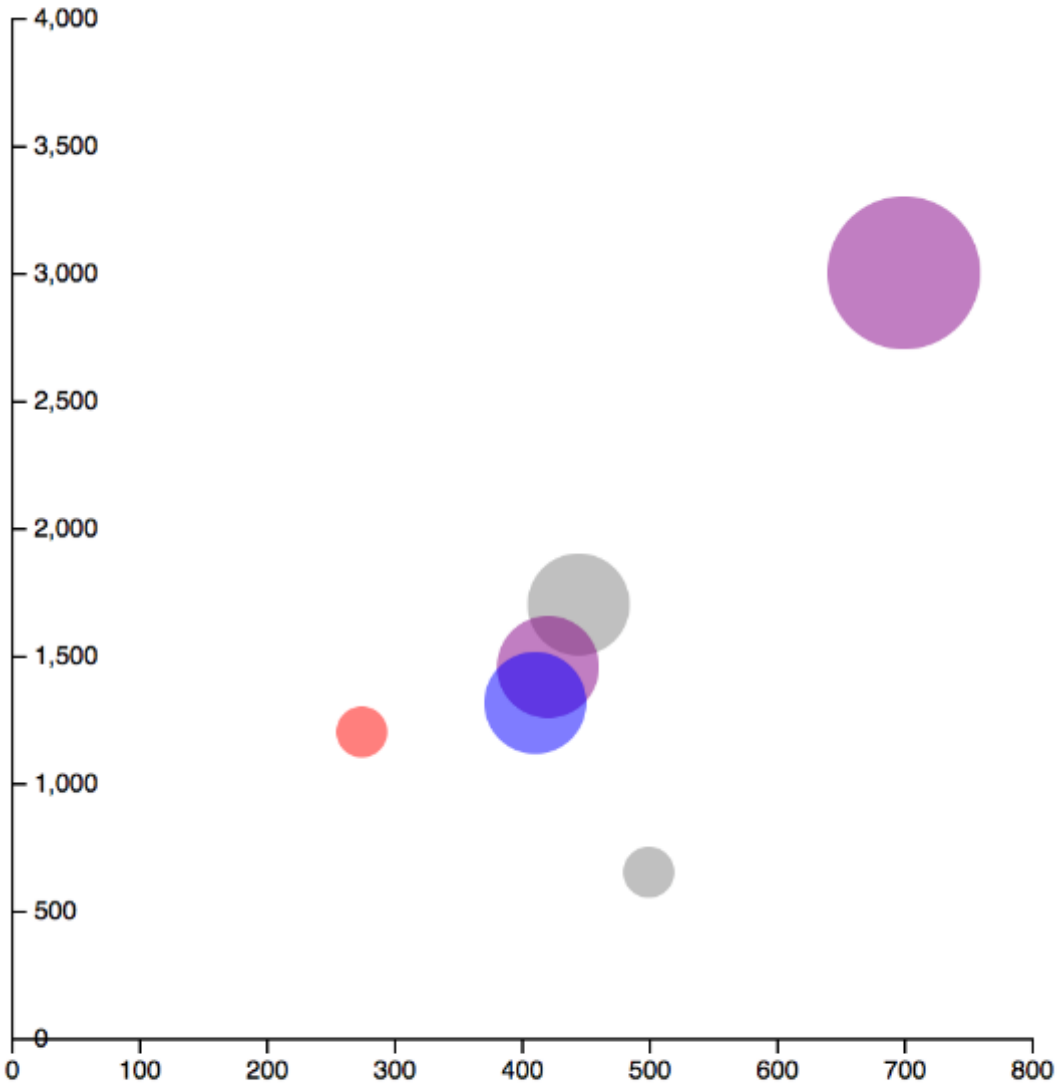
function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```

```
var svg = d3.select("svg");
var selection = svg.selectAll("g")
  .data(values)
  .enter()
  .append("g")
  .attr("transform", "translate(10,10)");

selection.append("circle")
  .attr("cx", (d,i) => { return d.price / 2; })
  .attr("cy", (d,i) => { return (4000 - d.sqft)/(4000/400) ; })
  .attr("r", (d,i) => { return d.br * 10 ; })
  .style("fill", (d,i) => { return color(d.pets); })
  .style("opacity", "0.5")
  .append("svg:title").text( (d,i) => { return print(d); });

function color(pets) {
  var dogs = pets.indexOf('dogs') !== -1;
  var cats = pets.indexOf('cats') !== -1;
  if (dogs) return cats ? 'purple' : 'blue' ;
  else return cats ? 'red' : 'gray';
}

function print(home) {
  return `$$${home.price}k, ${home.sqft}sqft, ${home.br} BRs`;
}
```




```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```



```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale) ;

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

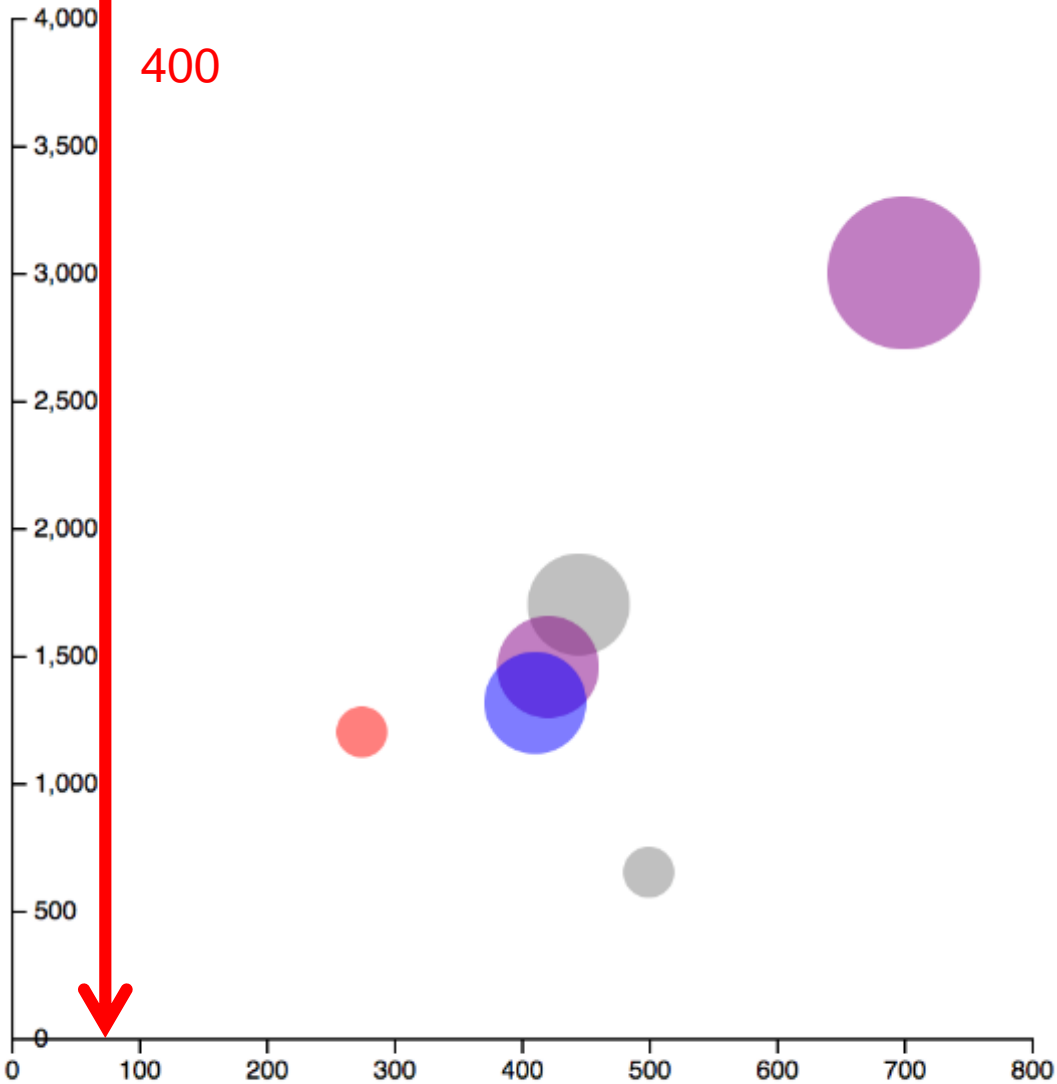
```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```



```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```

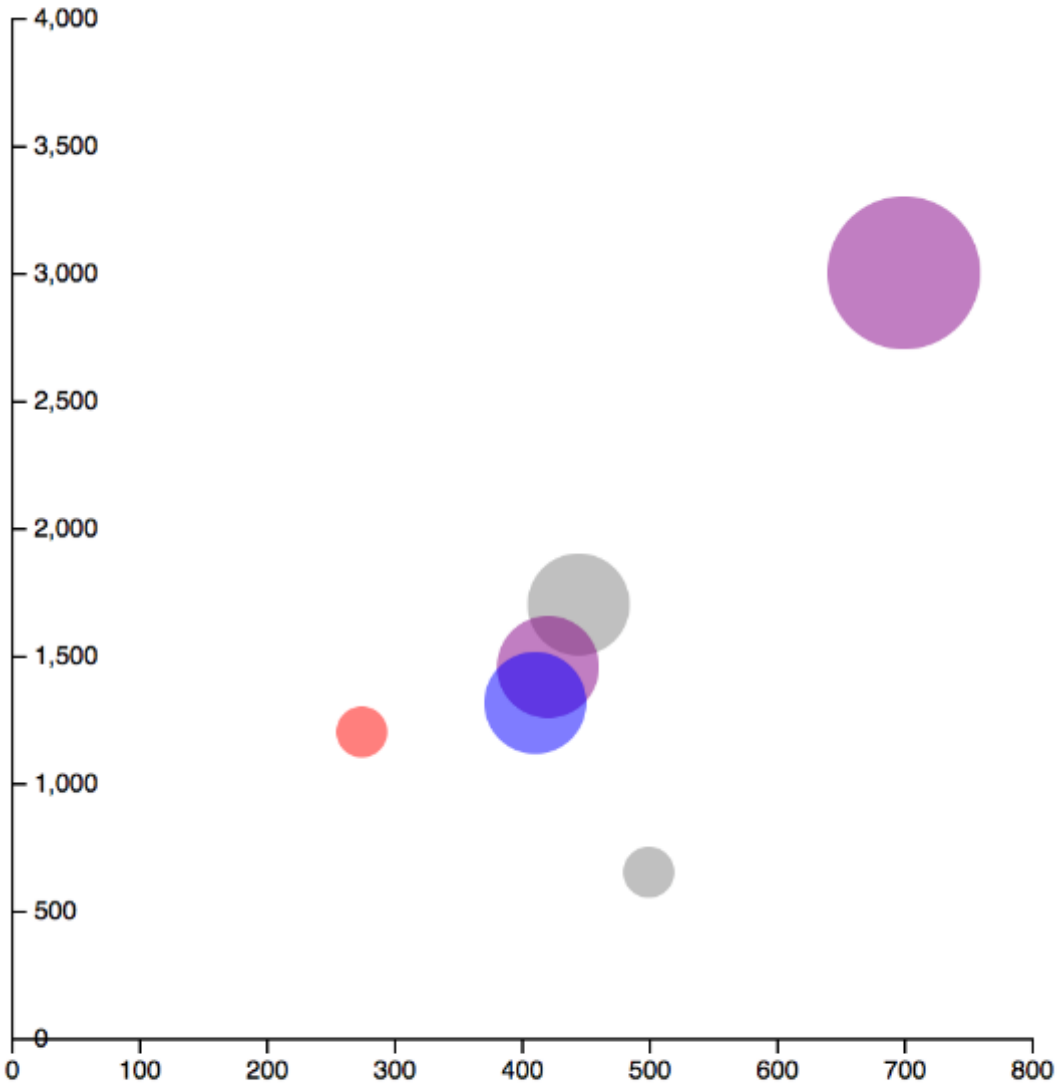
```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height, 0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```




```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
  .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```

```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
  .domain([0, width*2])
  .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
  .attr("transform", "translate(10,410)")
  .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
  .range([height,0]);
  .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g") .attr("transform", "translate(10, 10)")
  .call(yAxis);
```

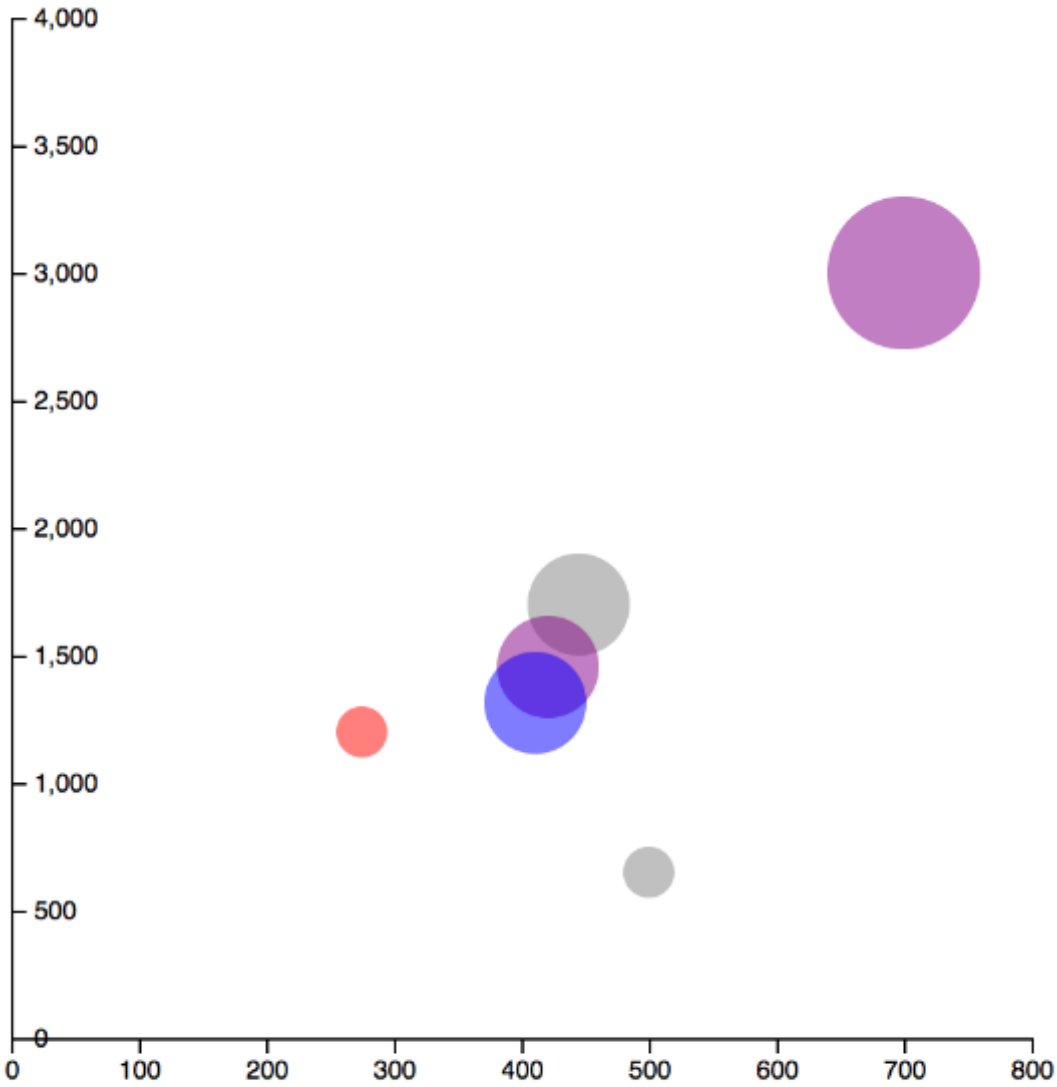
```
var width = 400;
var height = 400;

// draw the x-axis
var xScale = d3.scaleLinear()
    .domain([0, width*2])
    .range([0, width]);
var xAxis = d3.axisBottom(xScale);

svg.append("g")
    .attr("transform", "translate(10,410)")
    .call(xAxis);

// draw the y-axis
var yScale = d3.scaleLinear()
    .range([height,0]);
    .domain([0, 4000]);
var yAxis = d3.axisRight(yScale);

svg.append("g").attr("transform", "translate(10, 10)")
    .call(yAxis);
```



Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```


Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Accessing Data with D3.js

- D3.js can easily access data on the Web that is available through RESTful APIs

```
var values = [];  
  
var URL = . . .  
  
d3.json(URL, (response) => {  
    // populate the values from the data in  
    // response that comes back from request  
    . . .  
});  
  
// now use values with D3 functions  
  
var svg = d3.select("svg");  
var selection = svg.selectAll("g")  
    .data(values)  
    . . .
```

Summary

- D3.js allows us to generate HTML and SVG elements based on data
- We can apply functions to data sets to generate graphical elements, e.g. charts
- The data used by D3.js can include objects
- You can easily access data online using D3.js functions

Review: Week 3

- **React**
 - library and framework for creating reusable, modular components
 - can render themselves based on their state
 - can be combined and work together
- **D3.js**
 - library for generating HTML and SVG based on data
- **ES6**: more recent version of JavaScript