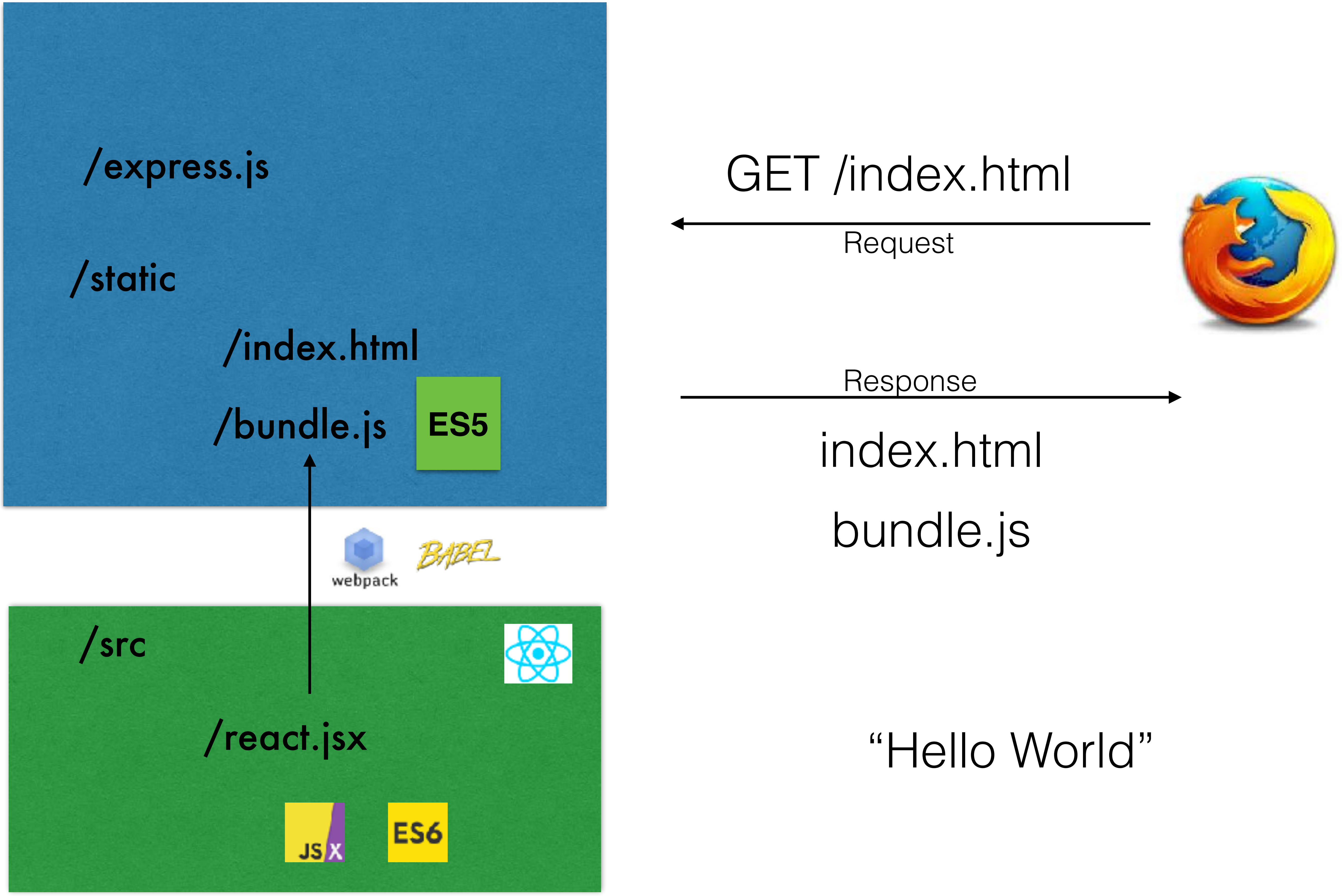
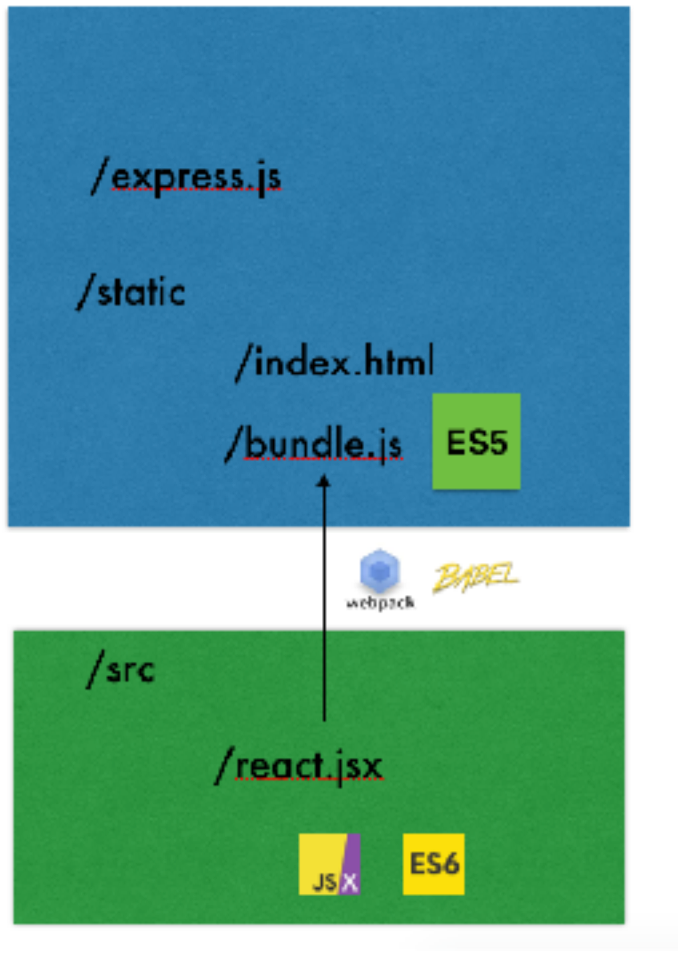

/express.js
/static
/index.html
/bundle.js
ES5
webpack
BABEL
/src
/react.jsx
JSX
ES6
GET /index.html
Request
Response
index.html
bundle.js
"Hello World"

Webpack Watchmode



webpack --watch

OR

watch:true in webpack.config.js

ES6 import, export variations

- export
- export default

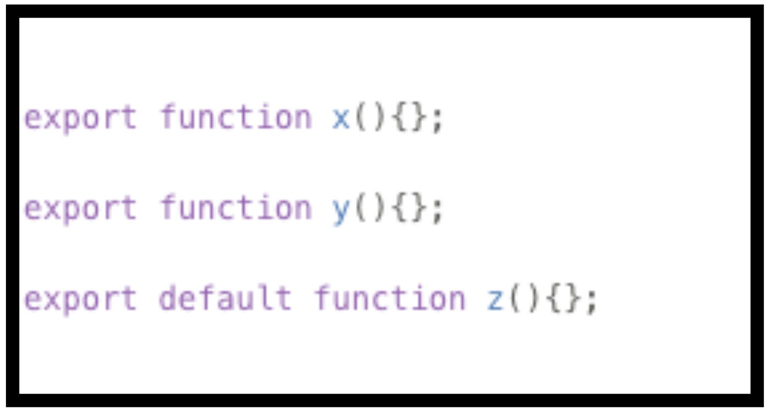
- Import Everything
- Import Selectively
- Import Default Exported Item

ES6 - Exports

- **Named Exports**
  - **export** keyword in front of functions, classes, expressions
  - Multiple named exports
    - eg. **export function() {….}**
- **Default Export**
  - Only one per script
  - It’s the "main" exported value since it will be the simplest to import.

    -> https://developer.mozilla.org/en/docs/web/javascript/reference/statements/export

es6_module.js



```
export function x(){};

export function y(){};

export default function z(){};
```

ES6 import

- Import Everything
  - `import * as SomeName from ‘SomeModule’`
- Import Default
  - `import defaultExportItem from ‘SomeModule’`
- Import Specific
  - `import {item1, item2} from ‘SomeModule’`

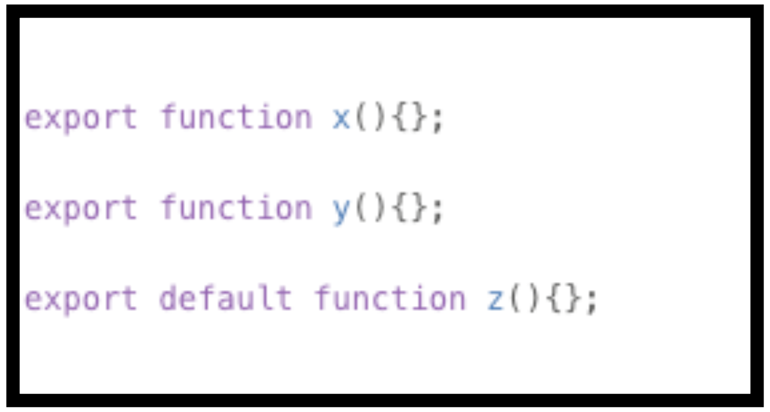
-> https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import

Import, Export Combinations

- ~~Node style export, Node style import~~
- **ES6 style export, ES6 style import**
- ES6 style export, Node style import
- Node style export, ES6 style import

ES6 export, ES6 import

es6_module.js



```
export function x(){};

export function y(){};

export default function z(){};
```

import * as es6mod from 'es6_module'

//es6mod = {x:x, y:y, default:z}

---

import esdef from 'es6_module'

//esdef = z

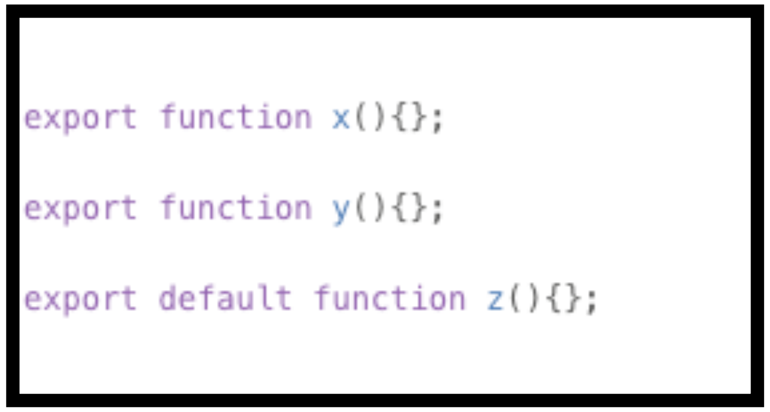
---

import {x,y} from 'es6_module'

//x =x, y=y

ES6 export, Node import

es6_module.js



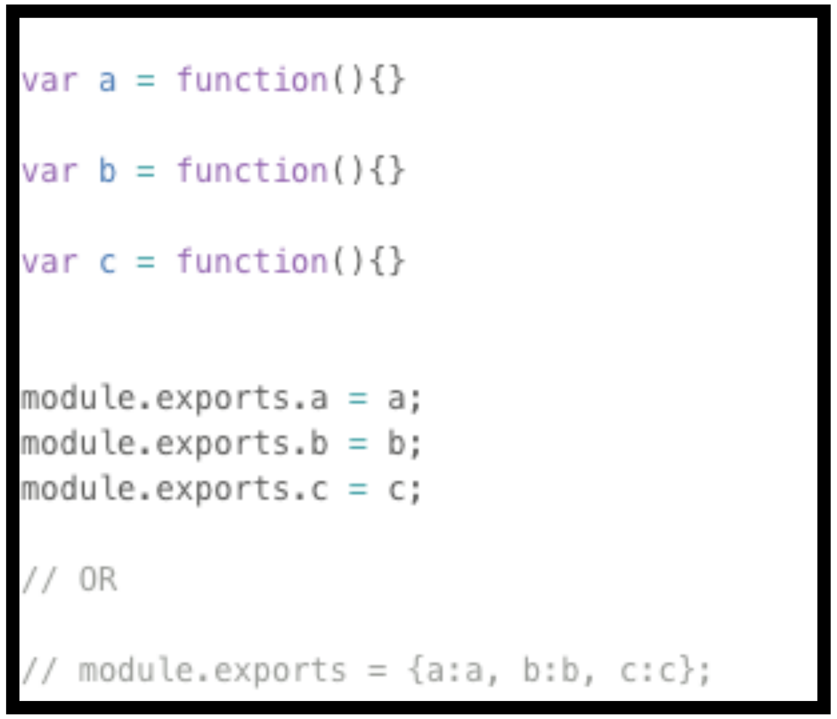
```
export function x(){};
export function y(){};
export default function z(){};
```

~ module.exports = {x:x, y:y, default:z}

```
var es6_node = require(“./es6_module.js);

//es6_node == module.exports = {x:x, y:y, default:z}
```

Importing a Node module in ES6

node_module.js



```
var a = function(){}

var b = function(){}

var c = function(){}


module.exports.a = a;
module.exports.b = b;
module.exports.c = c;

// OR

// module.exports = {a:a, b:b, c:c};
```

Importing a Node module in ES6

```
import * as nm from 'node_module'
```

Import Everything

```
//nm == module.exports == {a:a,b:b,c:c}

import {a,b} from 'node_module'
```

Import Selectively

```
//a=a, b=b
```

Import Default Item?

Importing a Node module in ES6

```
import esdef from ‘node_module’

//esdef == default item in Node module?
//default == module.exports for Node module!
//esdef == module.exports == {a:a, b:b, c:c}
```

So importing from Node modules - “ES6 import everything” and “ES6 import default” are equivalent.

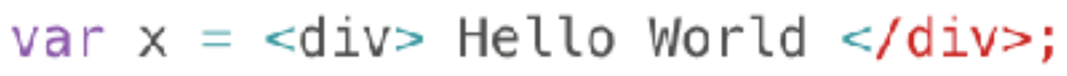
‘react’ is a Node module

```
import * as React from ‘react’;
import React from ‘react’;
```

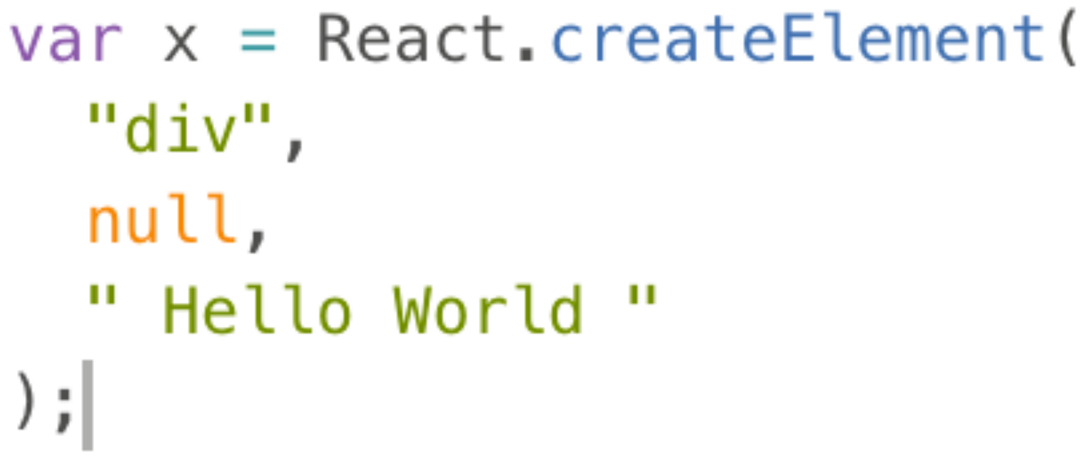
Both will capture module.exports of react.js

More JSX

- Syntax extension - JavaScript
- To express HTML elements in React code - what the UI should look like.



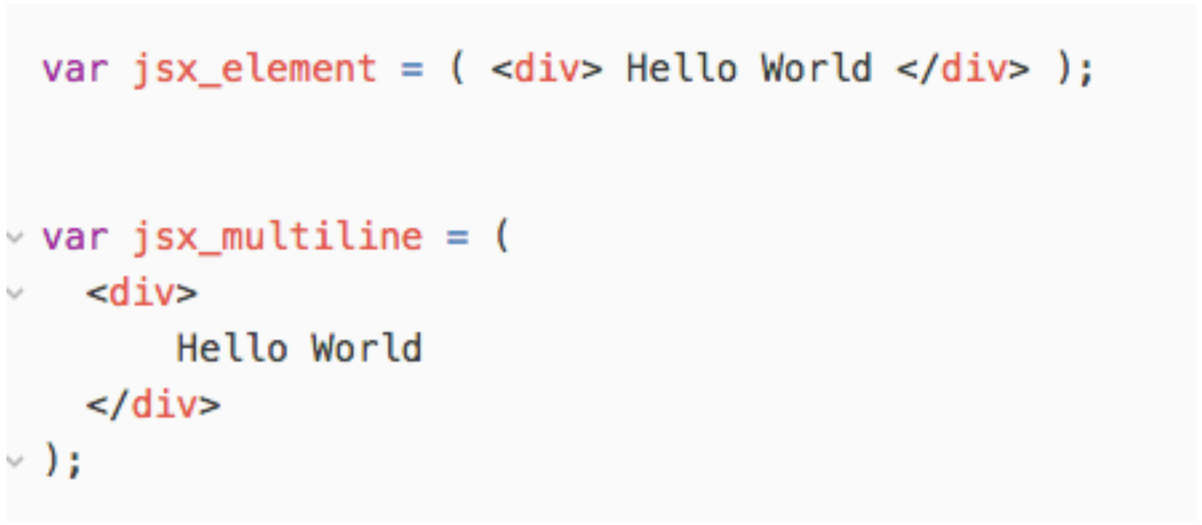
```
var x = <div> Hello World </div>;
```



```
var x = React.createElement(
  "div",
  null,
  " Hello World "
);
```

Wrap JSX in ()

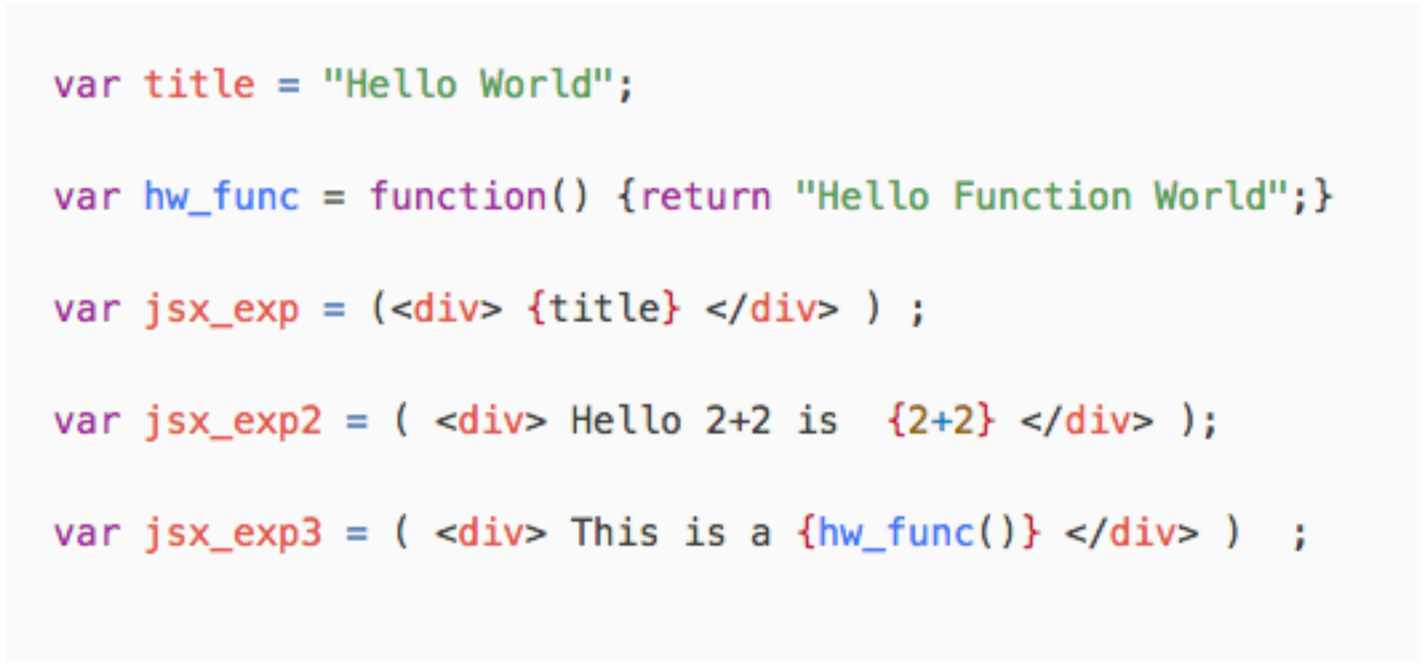
- To Prevent Automatic Semicolon Insertion
- http://stackoverflow.com/q/2846283



```
var jsx_element = ( <div> Hello World </div> );


var jsx_multiline = (
  <div>
      Hello World
  </div>
);
```

JavaScript Expressions using {}

Use {curly braces} to add JS expressions to JSX

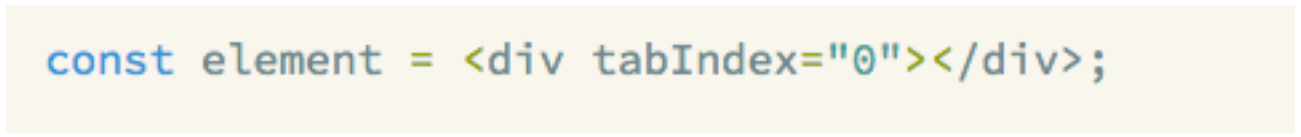


```
var title = "Hello World";

var hw_func = function() {return "Hello Function World";}

var jsx_exp = (<div> {title} </div> ) ;

var jsx_exp2 = ( <div> Hello 2+2 is  {2+2} </div> );

var jsx_exp3 = ( <div> This is a {hw_func()} </div> )  ;
```

JS > JSX > JS again : We will use this a lot!

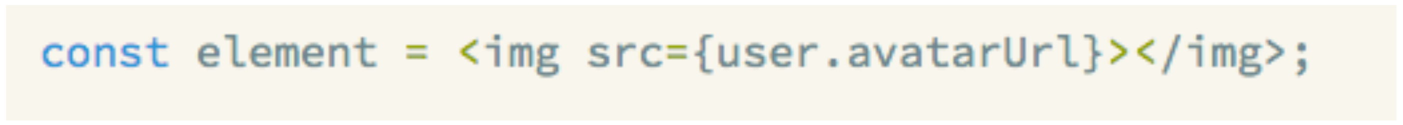
Specify HTML style Attributes

- As string literals (Use Double Quotes)



```
const element = <div tabIndex="0"></div>;
```

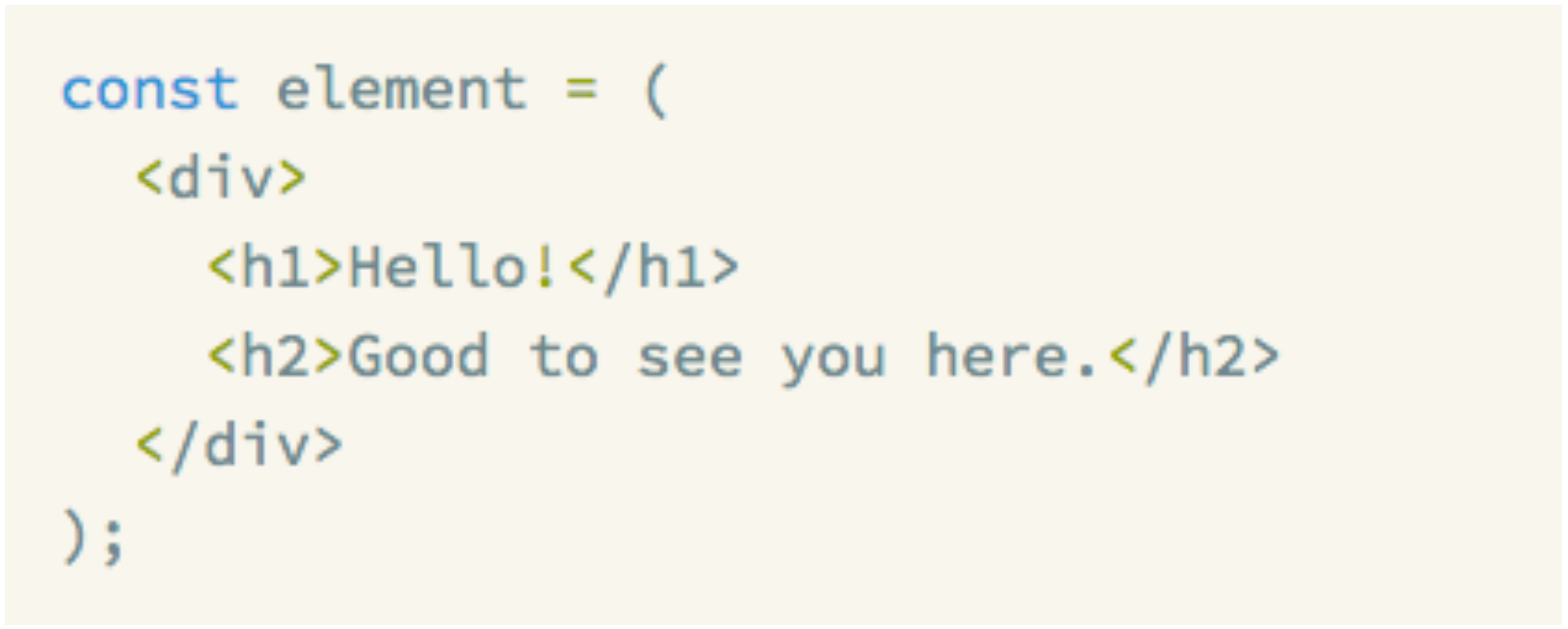
- Use JavaScript Expressions (in {})



```
const element = <img src={user.avatarUrl}></img>;
```

- *Don’t use both simultaneously!*

JSX tags can have children

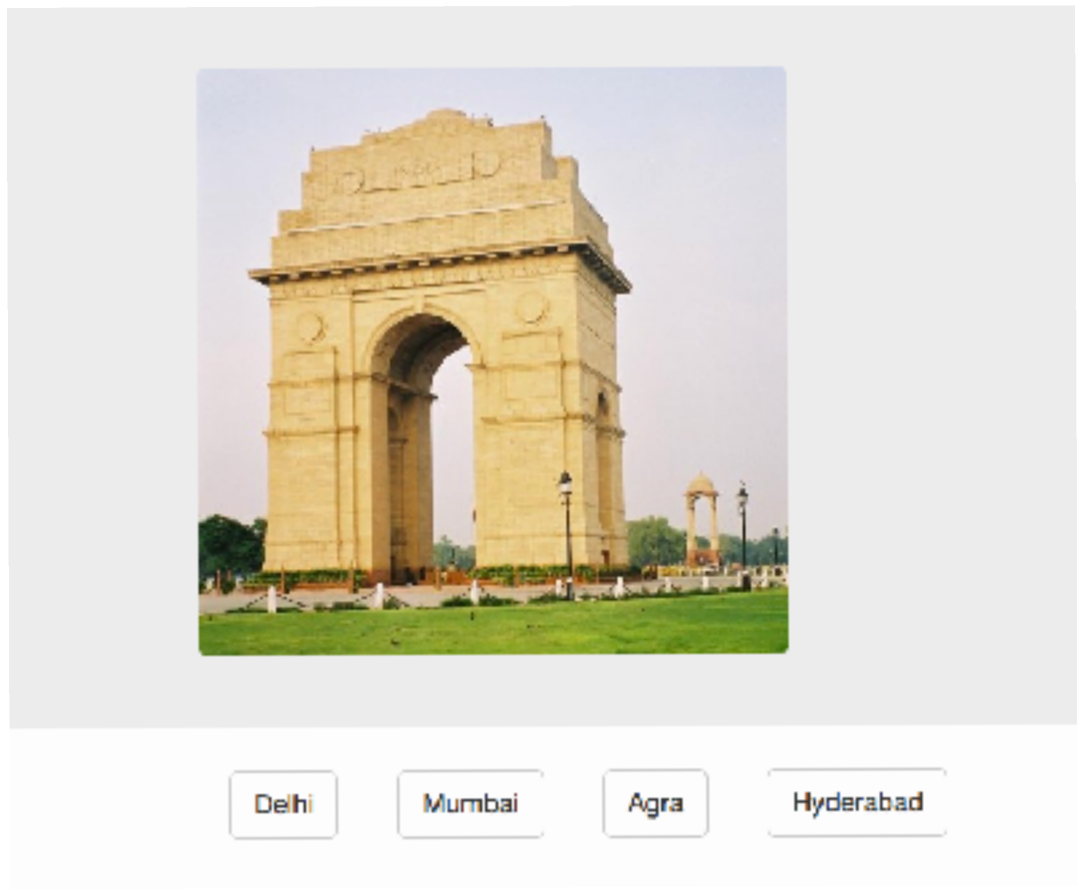


```
const element = (
  <div>
    <h1>Hello!</h1>
    <h2>Good to see you here.</h2>
  </div>
);
```

Day 2 - React - Topics

- **React Elements**
- **React Components**
- **Props**
- **State**
- **Events**
- Controlled Components, Forms
- API Handling
- React Router
- Related libraries - Redux …

Hands-On



- React Elements
- React Components
- Props
- State
- Events

React Elements

- Fundamental building blocks of React app
- Creating React Elements (we use JSX)
  - React Elements from HTML tags
    - <div/>
  - Instantiation of user defined React Components
    - <HelloWorld/>

- We inject React Elements into the DOM using ReactDOM.render

React Components

- React Components describe the UI with independent re-usable pieces

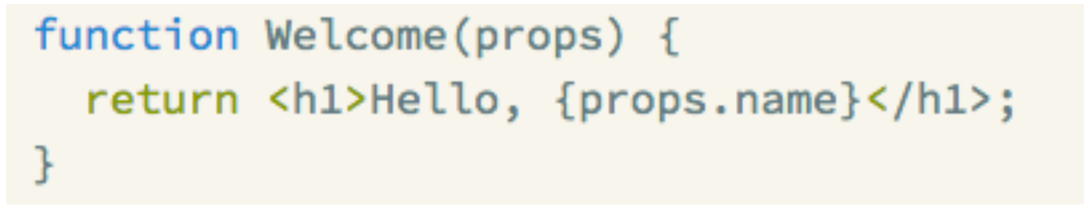
- React Elements are instances of React Components
- Analogy
  - FunctionConstructor/Class —-> Object
  - React Component —> React Element

React Components - Analogy

- Functional Constructors/Classes
  - Takes in data members as input
  - Return an object - instance of that ‘class’.
- **React Components**
  - Take in “**props**” as input
  - Return a React Element - an instance of that component.

Two types of Components

- **Functional Components**
  - Dumb/Pure/Stateless Components
  - Takes in **props** as input, returns a React element

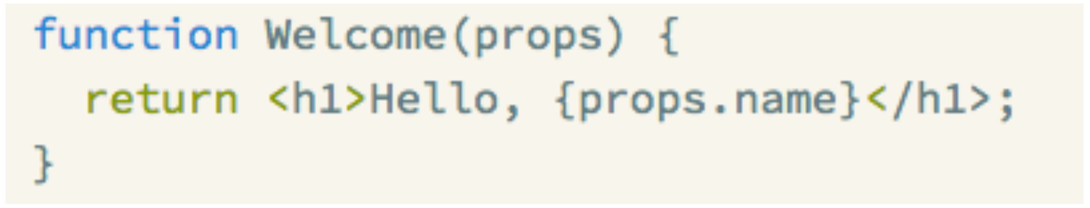


```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

- **Class Components**
  - Extends React.Component
  - Takes in **this.props** as input, returns a React element in render()
  - **Class Components let you store a state - how? - later.**

```
// import React and ReactDOM
import * as React from "react";
import * as ReactDOM from "react-dom";

// Create a React Component for our hello world
class HelloComponent extends React.Component{
    render() {
        return <div> Hello World from React! </div>
    }
}



// Create an object of this class
var my_hello_world_object = <HelloComponent/>;
var node = document.getElementById("app");

ReactDOM.render(my_hello_world_object, node);
```

Props



```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

- Data inputs for React Components
- React Components (Props) —> React Elements
- Eg. of Props : City Names
- How to access props within a component definition?
  - Functional components
  - Class components
- How to define props during instantiation
  - JSX Attributes

Hands - On

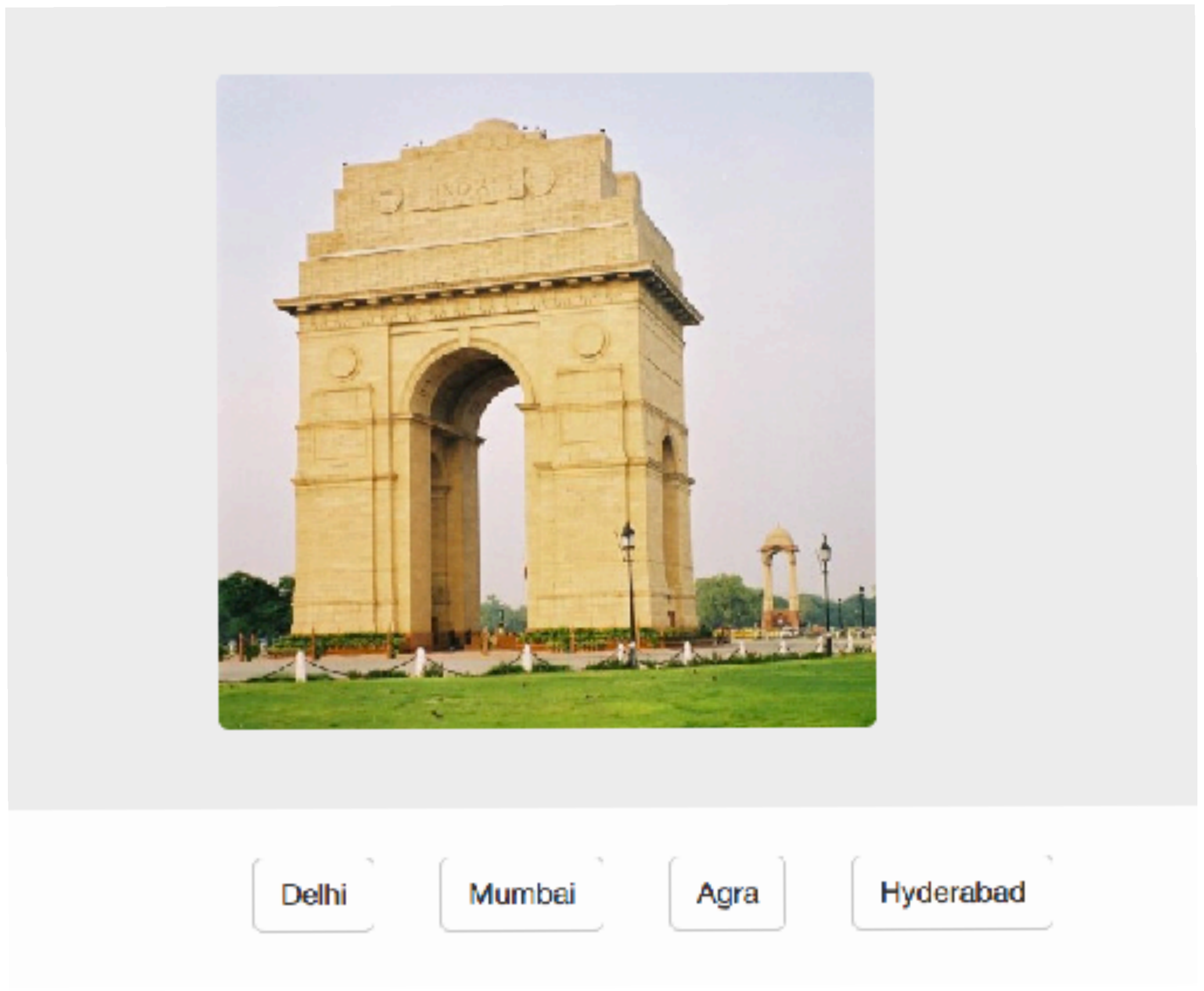
- Components
- Props - Setting Props, Accessing Props
- Instantiation of Components

Styling

	HTML	**JSX**
Class (Reserved keyword in JS)	<div **class**="container">	<div **className**="container"> <div className={var x = "container"}/> <div className={x}/>
Inline CSS styles	<div style="margin-left:20px;margin-right:10px">	<div style={ {marginLeft : "20px",marginRight: "10px" }}>
CSS Properties	margin-left margin-right	marginLeft marginRight

Props - Strict Rule

- React - flexible
- One strict rule though
- Props are ‘immutable’ with respect to their components
- Pure functions in JavaScript
- Components are “Pure functions” with respect to their props



- State
- Events

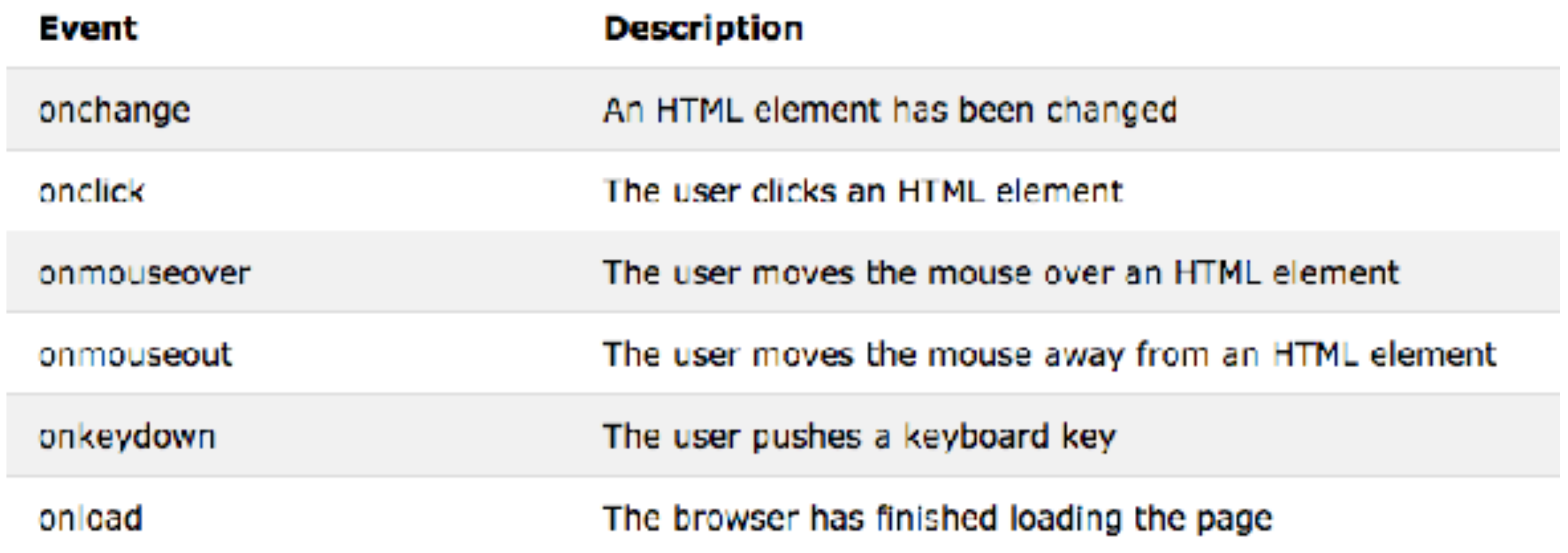
Composition of Components

State

- Class components can have state!
- Similar to props : but private and fully controllable by component.
- Useful for reacting to events
- Syntax
  - Creating a state
  - Modifying a state
  - Accessing a state

HTML Events

- HTML events are "things" that happen to HTML elements.
- JavaScript can react to these events



Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

JS/JSX syntax vs HTML syntax

HTML	**JS/JSX**
class	className
margin-left	marginLeft
onclick	onClick
onchange	onChange

Setting up Events - Hands On

- Identify Event
- Define Event Handler
- Attach Event Handler
- React : Combine State Changes with Event Handling

this.setState

- this.setState()
  - downward data flow
  - calls render() of component and its children
- React is declarative
  - “It should look like this” vs
  - “Do this” (imperative)
  - Butler Example , setState
- VirtualDOM - Homework and next class

HomeWork

Under The Hood

- Why is React so fast?
- DOM
    - API and DOM tree
    - not made for interactive UIs
    - expensive updates
- VirtualDOM
    - in memory representation of UI
    - cheap updates