February 24th, 2018 –

React\_JS [codecademy]

Javascript library developed at facebook

[open source projects](https://medium.mybridge.co/22-amazing-open-source-react-projects-cb8230ec719f)

react is fast – complex updates quickly

react is modular – many smaller, reusable files

react is scalable – best used displaying changing data

react is flexible – potential still unknown

react is popular – helps to become employable

# what is jsx

## A syntax extension for JavaScript. Written to be used with React (looks a bit like HTML)

#### This means JSX is not valid JavaScript and must be compiled and translated to JavaScript before reaching a web browser

## Basic unit of JSX is called a JSX element

#### Example: <h1>Hello World</h1> looks like HTML, but in a .js file

#### JSX element treated like JavaScript expression in that it can be:

##### Saved in a variable

##### Passed to a function

##### Stored in an object or array

###### const navBar = <nav>thing goes here</nav>;

###### const myTeam = { center: <li>Tim</li>, pointGuard: <li>Jim</li>, … };

##### Etc.

## JSX elements can have attributes

#### Looks like HTML element (can have one or multiple)

##### const navBar = <nav id=”nav-bar”>thing goes here</nav>;

## Nested JSX

#### To make it readable use HTML-style line breaks and indentation

#### If expression takes up more than one line, then you must wrap the multi-line JSX expression in parenthesis

#### Can be saved as variables, passed to functions, etc.

##### const nestedExample = (

#### <a href=“link here”>

#### <h1> Click link </h1>

#### </a>

#### );

#### JSX Outer Elements

##### A JSX expression must have exactly one outermost element

###### i.e. the first and closing tag of a JSX expression must be the same

###### You can always just wrap it in a <div> if this is an issue

## Rendering JSX - Make it appear on the screen

#### ReactDom

##### Name of the JavaScript library that deal with the [DOM](https://www.w3schools.com/js/js_htmldom.asp)

#### ReactDOM.render()

##### Most common way to render JSX

###### Only updates DOM elements that have changed (called “diffing”)

React is so successful because of this significant ability

Accomplishes this because of [*the virtual DOM*](https://www.codecademy.com/articles/react-virtual-dom)

Entire Virtual DOM gets updated

Virtual DOM is compared to snapshot of DOM right before the update

React figures out which objects have changed and change only those objects in the real DOM

Changes on the real DOM cause the screen

##### Takes the JSX expression, creates corresponding tree DOM nodes, and adds that tree to the DOM

##### The first argument (HTML looking thing) being passed should evaluate to a JSX expression, and it will be rendered on the screen

###### It doesn’t have to literally be a JSX expression

###### It could be a variable as long as it evaluates to a JSX expression

##### The second argument tells where to put the first argument on the screen

###### Example: document.getElementById(‘app’)

###### Note: The first argument is appended to whatever element is selected by the second argument

# Advanced JSX

### Grammar in JSX is mostly the same as HTML with subtle differences

#### class vs className

##### class in HTML is className in JSX because class is a reserved word in JS which JSX get translated you can’t use class

###### JSX className attribute automatically render as class attributes

#### Self-Closing Tags

##### Must include the / in self closing tags with JSX (optional in HTML)

###### <br /> is JSX is ok but <br> is not (even tho both ok in HTML)

### JavaScript in JSX (which is in JavaScript file)

#### Wrap in { } for JSX code to be read as JavaScript

##### Example: <h1>{2 + 3}</h1> will show 5 but without the { } it will literally show 2 + 3

#### Injected JavaScript is part of same environment as rest of file so you can access variables inside of JSX expressions even if variable declared outside

#### Object properties are often used to set attributes (organize code)

#### Event Listeners ([valid event names](https://reactjs.org/docs/events.html#supported-events))

##### Attribute value should be a valid/defined function

##### Written in camelCase for JSX not all lowercase like HTML

#### Conditionals: If statements that don’t work (can’t use an ‘ if ’ in JSX)

##### Explained [here](https://reactjs.org/docs/jsx-in-depth.html)

##### Common to keep the if else outside of JSX tags, not injected between

##### Ternary Operator – more compact way to write conditionals

###### [Explanation](https://stackoverflow.com/questions/6259982/how-do-you-use-the-conditional-operator-in-javascript): x ? y : z (if x truth return y, if x false return z)

##### && operator

###### Works best in conditionals that will sometimes do an action but other times do nothing at all

##### .map()

###### Is best bet for creating lists in JSX for example:

const arrays = [‘thing1’, ‘thing2’, ‘thing3’];

const listArray = arrays.map( arrayItem =>

<li>{arrayItem} </li>);

ReactDom.render(<ul>{listArray}</ul>, document.get … );

##### Keys – JSX attribute and the value should be unique (like and id)

###### React uses them internally (don’t see it) to track lists

###### React might scramble lists if you don’t use keys correctly

###### Needs keys if either of the following is true:

The list-items have ‘memory’ from one render to the next

i.e. was something checked off a list?

A list’s order might be shuffled

i.e. maybe a lists search results

Otherwise you don’t have to use keys (but doesn’t hurt if you do)

### [React.createElement](https://reactjs.org/docs/react-api.html#react.createelement)

#### You can write React code without using JSX (majority of programmers do use JSX, but don’t have to)

##### Example in JSX

###### const title = <h1>Hello World</h1>

##### Example of React without JSX

###### Const title = React.createElement(

###### “h1”,

###### null,

###### “Hello World”

###### );

##### When a JSX element is compiled the compiler transforms the JSX into the method above

# React Components

## A component is a small, reusable chunk of code that is responsible for one job. That job is often to render some HTML.

## import React from 'react';

#### // create a variable named React: import React from 'react'; // evaluate this variable and get a particular, imported JavaScript object: React // { imported object properties here... }

#### This imported object contains methods that you need in order to use React. The object is called the React library.

## *import* *ReactDOM* *from* *'react-dom'*;

#### *The methods imported from ‘react-dom’ are meant for interacting with the* [DOM](Link%20---%20https:/www.w3schools.com/js/js_htmldom.asp)

#### *The methods imported from ‘react’ don't deal with the DOM at all. They don't engage directly with anything that isn't part of React.*

#### *To clarify: the DOM is used in React applications, but it isn't part of React. After all, the DOM is also used in countless non-React applications. Methods imported from ‘react’ are only for pure React purposes, such as creating components or writing JSX elements.*

## Component Class

#### Every component must come from a component class (component class is not a component)

#### If you have a component class, you can create as many components as you want

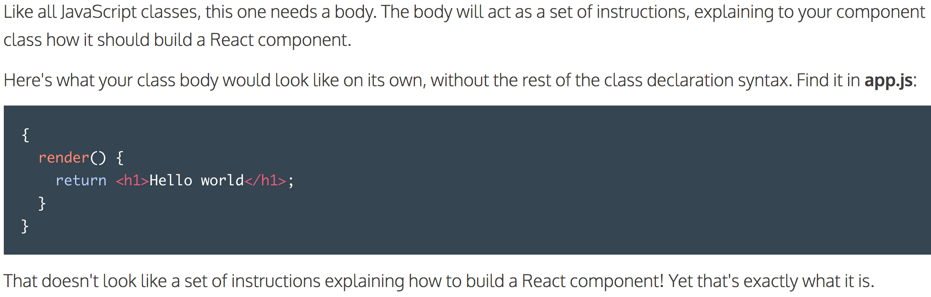
#### To make a component class you use a base class from React library (React.Component)

#### Links to more info on classes: [1](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes) [2](https://reactjs.org/docs/react-api.html#react.createelement) [3](https://hacks.mozilla.org/2015/07/es6-in-depth-classes/) [4](http://exploringjs.com/es6/ch_classes.html)

#### Component class variable names must begin with capital letters

#### This adheres to JavaScript’s class syntax (and [broader programming convention](Component%20class%20variable%20names%20must%20begin%20with%20capital%20letters!%20%20This%20adheres%20to%20JavaScript's%20class%20syntax.%20It%20also%20adheres%20to%20a%20broader%20programming%20convention%20in%20which))

## Review Components



## Render Function

#### This property must be included, name is render and value is a function

## Component Instance

### JSX elements can be either HTML-like, or component instances.

### JSX uses capitalization to distinguish

#### That is why component class names begin with capital letters – says “I’m a component instance, not an HTML tag”

# Components and advanced jsx

## Render( ) must have a return, but can also contain more

#### Example: Math.floor(Math.random() \* 10 + 1);

## If statement is located *inside* the render, but *before* the return statement

## Using this. in a component

#### This refers to an object on which this’s enclosing method (often .render()) is called

## Event Listeners

#### Render () {

##### Return (

##### <div onHover= {myFunc}> </div>

##### );

##### }

## Review Component

# Components render other components

## Component Instances: when you render a component in another component

## By default every JavaScript file is invisible to other JavaScript files

#### Use import statement to use variables between files

##### Also will need an export statement (exporting variable you hope to grab)

##### Rarely will you see import without export and visa versa

#### If string at end of import is a / or . then import treats the string as a file path

##### .js is assumed so is not necessary at the end of file name

#### This [Module system](http://eloquentjavascript.net/10_modules.html) is not specific to React.

##### React’s import/export specific [module system](https://hacks.mozilla.org/2015/08/es6-in-depth-modules/) comes from ES6

###### More in depth info [here](http://exploringjs.com/es6/ch_modules.html)

# Store dynamic information in react

## Dynamic Information – information that can change

#### React needs dynamic info to render

#### Two ways a component can get dynamic information

##### Props

###### Passed in from the outside

##### State

###### Component decides its own state

#### Every other component besides these two should always stay the same

#### React apps is really just components setting state and passing in props to one another

#### this.props

##### A component can pass information to another component

###### This information is known as “props”

A prop is an object

###### You can pass information to a React component by adding an attribute

Set name attribute equal to info you want to pass, use { } if passing something that is not a string

###### Most common way to use props is to pass info from one component to another

##### How to make a component display the info it is passed in

###### Find component class that will receive the info

###### Include this.props.name-of-info in that component class’s render method return statement

##### Props clarification

###### Props references the object that stores all the info

###### Props also is the plural of prop, which are the individual pieces of the props object

##### Props to make decisions

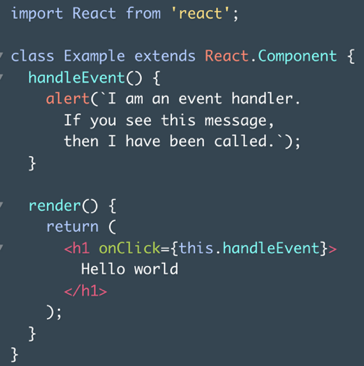
###### Props are not always shown on the screen, but often used to make decisions on what should be shown on the screen based on the attribute

##### Functions as props

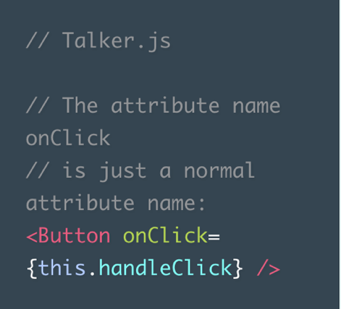
###### Especially common for event handlers

Must define event handlers in class before passing them anywhere

Define event handler as a method on component class (just like render)

example: 

##### Name like onClick only create event listeners if they’re used on HTML-life JSX elements. Otherwise, they’re just ordinary prop names

example: 

##### this.props.children

###### Will return everything between a component’s opening and closing JSX tags

###### If a component has more than one child they will be returned in an array, but if there is only one child it will just be returned (no array)

##### defaultProps

###### Set default props so that if there is no prop it isn’t left blank

#### this.props Recap (skills learned)

##### Passing a prop by giving an attribute to a component instance

##### Accessing a passed in prop via this.props.prop-name

##### Displaying a prop

##### Using a prop to make decisions about what to display

##### Defining an event handler in a component class

##### Passing an event handler as a prop

##### Receiving a prop event handler and attaching it to an event listener

##### Naming event handlers and event handler attributes according to convention

##### this.props.children

##### getDefaultProps

#### this.state example:

##### Should be equal to an object

###### That object is the initial state

##### Constructor and super are [features of ES6](https://hacks.mozilla.org/2015/07/es6-in-depth-classes/), not unique to React

###### React components always have to call super in their constructors to be set up properly

##### Note: methods should never be separated by a comma if inside of a class body.

###### This is to emphasize the fact that classes and object literals are different

##### To read a component’s state use: this.state.name-of-property

###### Just like this.props, this.state can be used from any property defined inside of a component class’s body

##### Update State using this.setState()

###### this.setState() takes two arguments

An object that will update the component’s state

A callback 🡪 you basically never need the callback

###### Bound the correct this [explanation](https://reactjs.org/docs/handling-events.html)

Simplified for now, in React when using an event handler that uses this, you need to add this.methodName = this.methodName.bind(this) to constructor function

###### As soon as this.setState() is called, it essentially then calls .render()

Thus, this.setState() cannot be called in .render() or it’d be an infinite loop