React Day 2

Agenda

- Modules
- Modifying Our React App
- Props

Important Vocabulary

Babel- JavaScript compiler that can translate elements into JavaScript.

JSX- An extension of JavaScript that merges the gap between HTML and JavaScript

React- JavaScript library created for dynamically creating User Interfaces.

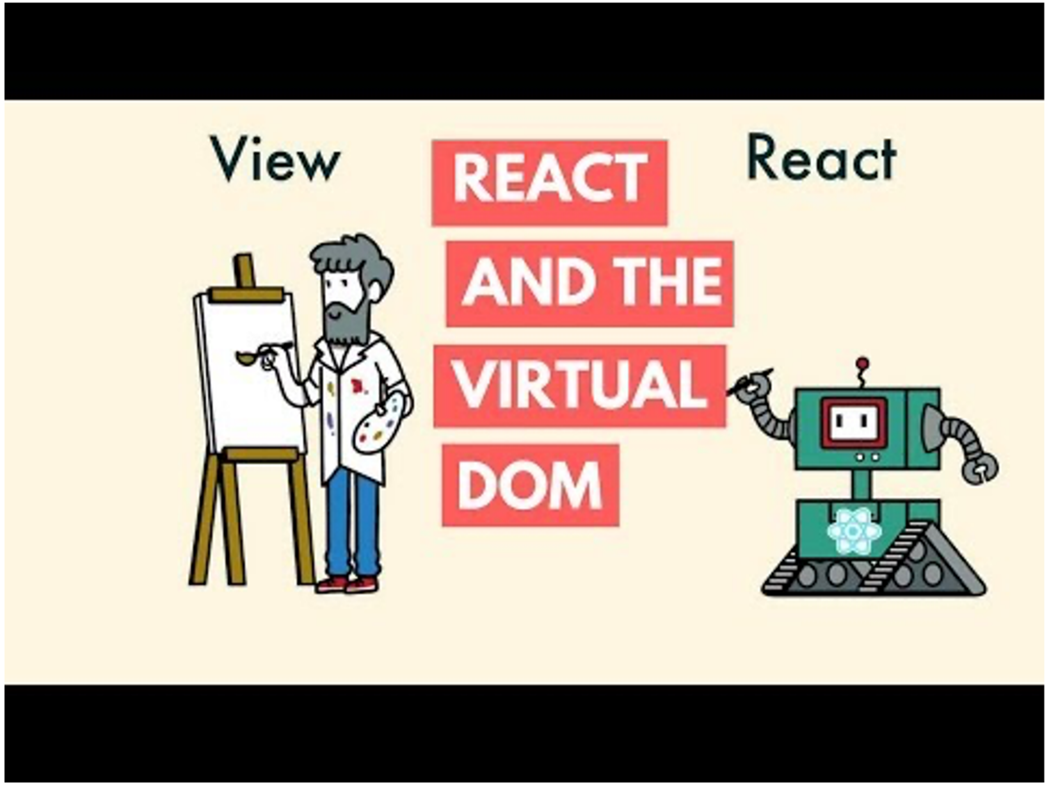
Importing Modules

Please always include:

import React from ‘react’

import React.DOM from ‘react-dom’

Virtual DOM



Let’s Create an app!

In gitbash/terminal run:

- npm i -g create-react-app (might have to use sudo)
- npx create-react-app testApp
- cd testApp
- npm start

Open Your App in Visual Studio!

Make sure your terminal is also open and in the app's folder.

React rendering

Use ReactDOM.render(*item*, *location*);

Example:

ReactDOM.render(<h1>Hi</h1>,

    document.getElementByID(‘root’));

For now, our first argument will contain JSX!
Location will use vanilla JS

General Rules of Thumb

- If you are embedding data use {data}
- Do **not** use quotes around the html you are rendering
    - “<h1>Words</h1>”
- When you return JSX, surround your text with parenthesis

Rendering with Functions

- More formally called “Functional Components”

- Functions **must** be “pure”

Pro tip: Use self closing tags for now! <tag />

Rendering with Classes

class ComponentName extends React.Component{}

Make sure you have a render() method, where your return the JSX you would like to use.

Props

Now, when we call our components we can pass data through the properties (props).

They are read only!

<Item data = “Data”></Item> or <Item data = “Data” />

WTWAW (What To Walk Away With)

- Be able to create and export a module
- Be able to render simple JSX
- Create a Functional Component
- Create a Class Component
- Use props to pass information