

Intro to React

Agenda

- Modifying DOM
- Props
- Promises?
- What is React?
- Basics

Exercise 2

Any Questions?

Goal For Today

Create a sample react app!

JavaScript DOM Manipulation

Accessing Information:

- `document.getElementById('myID');`
- `document.getElementsByTagName('p');`
- `document.getElementsByClassName('mainMenu');`

Basics of Writing To Document from JavaScript

For now, we will only learn one way to dynamically write html from our JavaScript:

```
document.writeln("html tags and text here");
```

For example: `document.writeln("<p>Paragraph Text</p>");`

Basics of Writing To Document from JavaScript

You may also embed variables into your html now!

For example:

```
let x = "Station Wagons";
```

```
document.writeln("<p>My favorite cars are " + x + "</p>");
```

Most of the examples posted use this, so test it out!

Advanced DOM Manipulation

- `element.innerHTML = new html`
- `element.attribute = new values`
- `element.style.property = new style`
- `element.setAttribute(attribute, value);`

Advanced DOM Manipulation

- `document.createElement(element)`
- `document.removeChild(element)`
- `document.appendChild(element)`
- `document.replaceChild(new, old)`
- `document.write(text)`

Promises!

JavaScript is single threaded, which means we may run into issues when:

- We have a taxing calculation
- We are waiting on a response

Promises!

- let x = new Promise();
 - In the parenthesis we need to add the parameters resolve and reject
- Example:

Let x = new Promise((resolve, reject) => {});

Inside the curly braces you can decide to resolve() or reject()

That's so fetch

fetch(url)

- A Promise based way of doing a GET request
- Allows us to then decide what to do if it:
 - Is successful: Use then()
 - Fails: use catch()

Promises will always either be successful or throw an error.

What is React

A JavaScript Library that allows us to dynamically create user interfaces.

Developed by Jordan Walke, an engineer for Facebook. in 2010ish. Facebook has been using it since 2011.

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.

Important Vocabulary

Babel- <script type= “text/babel”>

JSX- const element = <p>Hi All</p>

React- ReactDOM.render(...);

Givens

React Apps are all built as a tree (as we've seen before), the root node is where everything is created.

In React:

- Elements are rendered
- Elements are immutable

Core Concepts of React

React cares entirely about rendering components to the DOM

React prefers to work with fully formed components that can be reused to design a webpage.

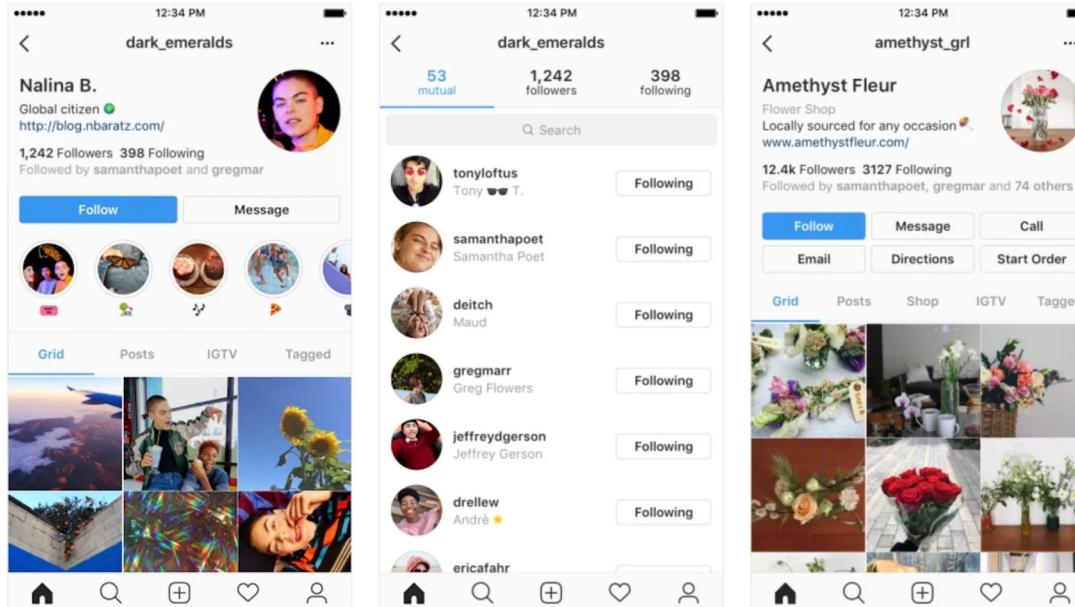
Virtual DOM

One of the core reasons why React is so popular:

- React keeps a lightweight version of the actual DOM for your webpage
- As changes are made, the virtual DOM is compared to the real DOM to see what needs to be updated

React Native

Same principles behind designing web apps with React can be used to design mobile apps-- using React Native.



Before we begin

Make sure you have Node.js installed!

We will be using the *Node Package Manager (npm)* a lot, which comes automatically installed with Node.

- npm install <package>
- npm start
- npm run build

Let's Create an app!

In gitbash/terminal run:

- `npx create-react-app testApp`
- `cd testApp`
- `npm start` (can also do `npm run start`)

Let's walk through the app we created

React rendering

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

Example:

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

```
document.getElementById('root'));
```

For now, our first argument will contain JSX!

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

WTWAW (What To Walk Away With)

- Be able to access HTML elements from JS
- Be able to write and interpret a basic Promise
- Create a sample React App
- Know what JSX is!