

Intro to React

Agenda

- Modules
- Promises
- What is React?

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()`

Something you have already seen

`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.

JS Modules

- Provide the ability to encapsulate and abstract code
- JS has several module systems (common, AMD and ES6)
 - Can be used both in the browser and on the server
 - Most common JS module library today is Webpack

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.

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
- npm start
- npm run build

Let's Create an app!

In gitbash/terminal run:

- `npx create-react-app firstapp`
- **Navigate to where you want to create the app**
- `create-react-app firstapp`
- `cd firstapp`
- `npm start`

What is create-react-app?

Installs react as well as:

- Development Server to run the app on
- Webpack
- Babel (more later)

Create-react-app handles many of the properties and tools we would have to otherwise tinker with ourselves!

Important Vocabulary

Babel- JavaScript transpiler/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.

[Example](#)

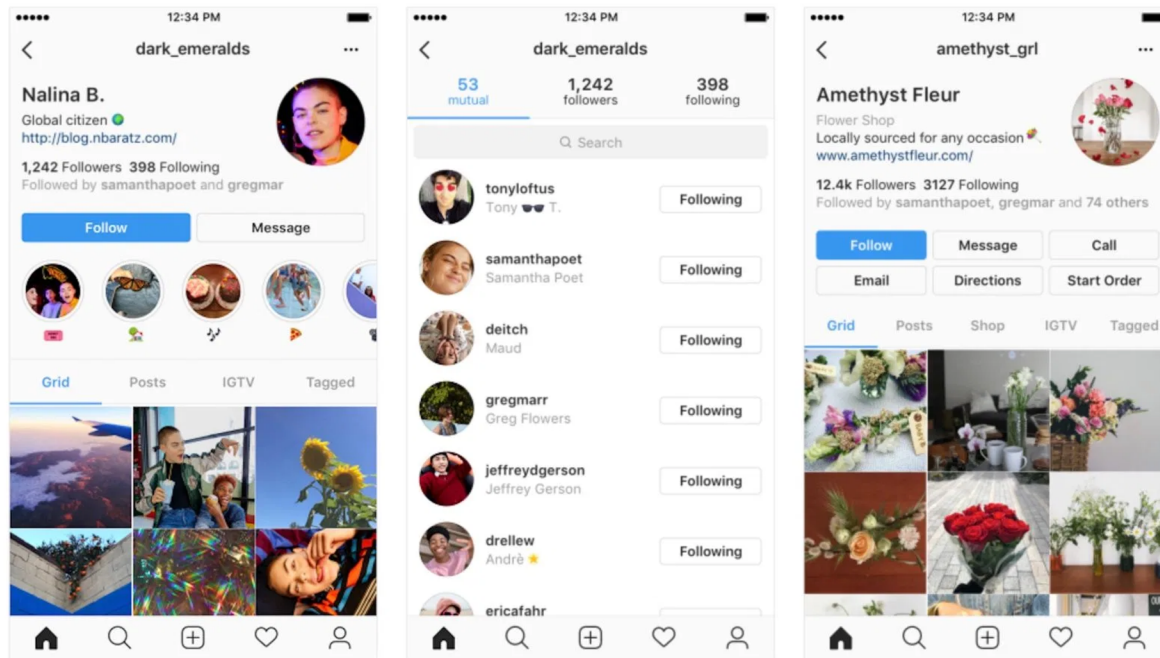
Virtual DOM

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.



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