WHAT IS CREATE-REACT-APP

create-react-app (CRA) is a program that creates a new react application, including any needed non-react packages to make life convenient

- webpack for bundling
- eslint for linting
- babel for transpiling

This is known as a "boilerplate" app

Not required for React but awfully convenient

CRA is fairly un-opinionated

INSTALLING AND USING CRA

```
npm install -g create-react-app
```

WAIT!!!

What do you mean, a global install !?

Yes - this is a tool, not part of the code we're writing

```
create-react-app demo
```

WHAT DID THAT DO?

- installed a `package.json` all the dependencies
- Gave us a *HUGE* README file
  - Tons of info there, look into it
- Created a `/src` directory
- Created a `/public` directory
- Created a `yarn.lock` file (like `package-lock.json`)

WHAT NOW?

Some commands are defined in `package.json`:

- `npm eject` - removes CRA, leaves the code
  - Don't use this, but try it out on a test project
- `npm build` - creates real HTML and JS files in `/build`
  - NO SERVER - you have to write or use one
- `npm start` - starts a development server that will show the code in progress
  - Auto reloads/rebuilds when the code changes
  - Great to develop against
  - Not for final use - use `npm build` and copy the `/build` results for a real server

WHAT DO WE HAVE?

- in `/public` the `index.html` file has no real content
  - but does have `<div id="root"></div>`
- in `/src/` the `index.js` loads `App` and injects it into `#root`

But what is `App`?

What are these `import` commands?

To make ME happy, rename `src/App.js` to `src/App.jsx`

BUILD PROCESS

This code is all being bundled with `webpack` and transpiled by `babel`

Actual JS will be different than you see.

Run `npm start`

- This starts a server (keeps terminal busy)
- This transpiles and bundles the code

Fancy, but just like `browserify`, just more involved

IMPORT

First, let's talk `import`

`import` is like `require`, but isn't for node (and currently requires a transpiler/bundler)

```
import someValue from 'foo';
// similar to
// const someValue = require('foo');
```

Unlike `require()`, you can pull in more than just one default export.

```
import { other1, other2 } from 'foo');
```

You can do both:

```
import Foo, { other1, other2 } from 'foo';
```

EXPORT FOR IMPORT

```
export default someVar; // default export
export const foo = "imported with `import {foo} from './foo'`";
const bar = "another non-default export";
export bar;
```

NON-JS IMPORTS

`import` is a standard, even though it requires a bundler/transpiler.

Some **non-standard** extensions are often used to bring CSS and/or images into a web-page

- requires a **configured** transpiler/bundler to do
- CRA gives you that config

```
import React from 'react'; // standard default import from library
import ReactDOM from 'react-dom'; // standard default import from library
import './index.css'; // NON-STANDARD, adds CSS to page
import App from './App'; // standard default import from local file
```