Starting with React

Yay! Finally, REAL web dev

- Um, Actually...

This course covers multiple REAL ways of webdev

- Server-side HTML generation
- Service development
- Vanilla JS HTML manipulation
- React

There's a lot this course doesn't cover

- Better ways of HTML generation server-side
    - Including React! (SSR + SSG)
- Lots of details about webservers
- Other service types beyond REST
- So much a11y, i18n, HTML, CSS

Just too much to cover

- Goal is to get you to where you can grow
    - But you can do webdev NOW
- "Bad" code can still benefit the world
    - So benefit the world as you learn

Is React hard to learn?

- All depends on the mindset
- I've tried to create patterns
    - Event to State to Render
- If you are overwhelmed
    - Simplify what you are trying to understand
    - Not understanding is natural!
        - The process is called "learning"
            - Not automatic
            - Don't try to force it

`create-react-app` (CRA) is a program

Not required for React

- Convenient setup and configuration
- Other, similar programs exist
- Creates/configures package.json
- Installs npm packages
- Common configs
- Webpack for bundling
- Eslint for linting
- Babel for transpiling
- Notable: fairly un-opinionated

Create a test app

```
npx create-react-app test-app
```

Tells NodeJS to download and run create-react-app

- to create app "test-app"
- You can give any name you want

Creates a `test-app/` directory

- Where you run the command
- Puts in all the pieces

Can run `npx create-react-app .` (current directory)

Our new app

`create-react-app` takes a moment to run

- Only when creating a new app
- Lots of output!
    - We can look at it all later
    - Don't need to read it all now

Before we look at the details, let's see what we created

```
cd test-app
npm start
```

Umm...neat?

It started a server and is showing a spinning logo

- You can inspect the HTML
- The spinning is just CSS animation

Follow the suggestion and open `src/App.js`

- Leave the server running

What did `create-react-app` do?

- Gave instructions (these are in the README)
- Installed a `package.json` and the dependencies
    - "react-scripts" package
        - Wraps webpack/babel/etc
        - Also has configs for these
        - Some config is in package.json directly
- Created a `/src` directory
- Created a `/public` directory

What now?

Some commands are defined in `package.json`:

- `npm run eject` - removes CRA, leaves code/configs
  - Don't use this, but try it out on a test project
- `npm run build` - creates static files in `/build`
  - NOT `public/`! Like `/src`, `/public` is **input** here
  - NO SERVER - you have to write or use one
- `npm start` - starts a dev server
  - Auto reloads/rebuilds when the code changes
  - Not for final use - use `npm run build` and use the `/build` results for a real server

Dev Server

`create-react-app` uses the webpack dev server

- This means we can't modify the server!
    - no services!
- More on this later, no worries
- Today, we are looking at client-side only
- dev server is for development
    - not the actual final webserver

Opening src/App.js

This looks like a mix of JS and JSX

- some weird `import` statements
- function App() returns HTML
    - not as a string, just HTML
    - has some values in `{}`
    - uses `className` instead of `class`

Now look at HTML for the page

What do we have?

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

But what is `App`?

This Course: `src/App.js` to `src/App.jsx`

Change in filename

- Change the file `App.js` to `App.jsx`
    - Actual file, not the text

JSX files will work with either `.js` or `.jsx`

- For this course you must use `.jsx`
- It is extra information
    - there will be js files that have no JSX in them
    - JSX is for UI, not functionality
        - "View" of MVC
        - returned JSX is like our render() method

HTML of Page

```
<div id="root">
```

has inner HTML as the output of the App() function

- the `<App/>` JSX
- classNames became classes
- `{}` were replaced with links

now make a text change to App.js and save

Non-JS imports

`import` is a standard, despite requiring a bundler

Some **non-standard** transpiling 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
- Standard imports are available
    - And we will be using them

```
import logo from './logo.svg'; // NON-STANDARD, path to img
import './App.css'; // NON-STANDARD, adds CSS to page
```

Live Reloading

Change auto-shows in browser!

App.js "imports" App.css

- Make a change: `background-color: #e6e;`
- Browser shows this too!

A word about their CSS

Default CRA files: hyphenated mixed-case CSS classes

- I'm not biased
- This is an abomination upon the face of the earth
    - In reality, all depends on your CSS approach
    - This course: stick with our semantic style

For this course, class names must be kebab-cased

- All lowercase
- Hyphen-separated (BEM-style allowed)

Where is the HTML?

The `import` brought in the css file

- you can import additional/different css
- CSS filename(s) do not need to be Capitalized

The HTML is in `public/index.html`

- BUT we won't be changing it
    - Except `<title>` and webfonts (if any)
- Make all your changes in the js/jsx/css files in `src/`

Building

`create-react-app` is a tool to help develop

- In the end we want static HTML/JS/CSS
- We can put those on any server

Stop your server (ctrl-C)

- Then run `npm run build`

Build Process

All bundled using `webpack` and transpiled by `babel`

`npm start`

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

`npm run build`

- Generates static built/bundled files

This is just pre-configured work we've done before

What did `npm run build` do?

We now have a `build/` directory

- Contains HTML/CSS/JS files
  - Plus some images
- Files have weird names
  - cache-busting

These files are all you need

- Can put on any static webserver
  - Using any backend language
- No CRA, no special programs

When do we build?

Do all your development with the development server

- Uses `npm start` to run (default config)

If done and putting up web app for the public

- `npm run build`

Later we can change the specific commands

- By modifying `package.json` "scripts"

Summary - Create React App

`create-react-app`

- A program to create a configured react project
- not required for react, but convenient
- includes not just react
    - babel, webpack, linting, etc

Summary - cra commands

- Found in README.md of created project
- `npm start` - runs the dev server
- `npm run eject` - removes cra wrapper
    - great to learn from
    - not usually used
- `npm run build` - creates static files
    - usable in most static webservers
    - for deployment use, not development
    - we will use this soon!

Summary - special imports

CRA babel/webpack config

- includes common but non-standardized imports
    - import CSS
    - import asset paths
- You can import `.js` or `.jsx` files
    - You can omit file extension on these

Additional options exist we won't use

- But TypeScript, SASS, etc
    - good to explore for future

Summary - Required Setup Changes

For this course

- Rename `App.js` to `App.jsx`
- Use `kebab-case`/BEM css classes
    - not `MixedCase` like `class="App"`
- Always have exactly 1 component per `.jsx` file
- Components filenames match component name
    - `MixedCase.jsx`
- JS filenames
    - `camelCase.js`
    - No JSX in them

Summary - Editing

All our source files will be in `src/`

- not `public/`