


Start of Class Prep

In a directory where you would create projects for labs, run this:

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

This will take a few minutes, so start it up and we'll look at it later in class

Why React?

- "View" only - unopinionated about the rest
 - Makes it more flexible
- React is Declarative
 - Makes it less complex to read/change
- React maps closely to HTML output
- React components map closely to JS functions
 - Same best practices
 - Non-React skills increase React skills
 - React skills are useful even without React
- Also very popular right now (most? )
 - Means helps with jobs

State and Render

We've seen JS apps:

- having a local state
- rendering HTML when state changes

Automate Render on state change

We can imagine automatically re-rendering when our state changes.

- React will do that for us!

HTML is Declarative

HTML is "declarative"

- says what it is
- not how to do it
 - ex: Button is clickable, looks clickable

JS is "imperative"

- You give list of instructions

JSX is declarative

React uses **JSX**

- Looks like HTML
- Declarative
- actually is a JS function that returns HTML
- can call other JSX functions for HTML
- can insert HTML

JSX Example

```
function Greeting() {  
  return (  
    <p>Hello World</p>  
  );  
}  
//...elsewhere  
<Greeting/>
```

NOT js, but JSX

- browser can't handle without translation
- much friendlier to use
- output is HTML and JS

More JSX Example

```
function TodoItem({ task, done }) {  
  const complete = done ? 'todo__text--complete' : '';  
  return (  
    <li><span className={complete} >{task}</span></li>  
  );  
}  
//...elsewhere  
<TodoItem task="Pounce" done={false} />
```

A few differences!

- `className` instead of `class`
- `{}` to replace with values
 - notice no template literals (```) here
 - not strings!
 - no `${}` unless you have template literals

More JSX differences

```
function TodoItem({ task, done }) {  
  const complete = done ? 'todo__text--complete' : '';  
  return (  
    <li><span className="{complete}" >{task}</span></li>  
  );  
}  
//...elsewhere  
<TodoItem task="Pounce" done={false} />
```

A few differences!

- `{false}` instead of "false"
 - actual boolean, not a string!
- attribute-like values passed to function
 - "props", more on these soon

Important: React owns the DOM

Big change: Do not access the DOM!

- no `document.querySelector` or `document.getElementById`
- React is managing our DOM
- If we change it, we can confuse React

Why did we learn those parts then?!

- Know what React is doing
- Good without React

Create React App

React is great, but can have a lot of set up

- So we will have someone else do the hard work
- `create-react-app` is a program to set up:
 - react
 - building (converting react to HTML+JS)
 - linting (syntax warnings, hints, and help)
 - live reload
 - a development server
- CRA isn't required for React, but is convenient

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

Our new app

create-react-app takes a moment to run

- only when starting 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

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

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

Live Reloading

Change auto-shows in browser!

App.js "imports" App.css

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

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
 - our SOC of `fetch()` from showing results

A word about their CSS

The default CRA files use capitalized CSS class names

- 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

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`

What did that do?

We now have a `build/` directory

- contains HTML/CSS/JS files
 - plus some images
- Files have weird names
 - cache-busting, that's a 6250 topic

These files are all you need

- can put on any static webserver
- no CRA, no special programs

When do we build?

Do all your development with the development server

- uses `npm start` to run

If done and putting up web app for the public

- then `npm run build`

Summary - React

React will let us auto-render when state changes

React uses JSX

- JS that looks like HTML
- can embed HTML
- uses `className` instead of `class`
- uses `{}` to replace with variable values
- can have non-strings (unlike HTML)

Summary - Create React App

CRA is a program that makes React easy to use

- not required to use React
- has other features (e.g. live reloading) added

CRA creates a directory for the app

- start dev server with `npm start`
- build prod files with `npm run build`

Summary - Editing

Edit files in `src/`

- Instructor demands rename `App.js` to `App.jsx`
 - because it is information about file
 - Remember to do on new projects
- Instructor demands use kebab-case class names
 - change/replace `className="App"`, etc
- Can rename/replace `App.css`
 - just `import` needed css file(s) in `App.jsx`