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

HTML is Declarative

HTML is **declarative**

- Says what it is
- Not how to do it
    - Ex: Button is clickable, looks clickable
    - Ex: A `<form>` is a form, an `<input>` is a field

JS is **imperative**

- You give list of instructions
    - "How" to do anything

We've kept HTML, CSS, and JS separate so far

- Hard to edit one in the other
  - No inline JS
  - No inline CSS
- But we're starting to feel limits
  - `.innerText` and `.innerHTML` put HTML in JS
  - JS uses a lot of class names from HTML
- State/render would do even more
  - Lots of HTML in JS

JSX is Declarative

React uses `JSX`

- Declarative
- Looks like HTML
- Actually a JS function that returns HTML
- Can call other JSX functions for HTML
- Can insert HTML
- Allows for easy editing of HTML in JS

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`
- No `document.getElementX`
- No `classList.toggle()`, etc
- 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

Vite

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

- So we will have someone else do the hard work
- `vite` is a program to set up:
    - React
    - Building (converting react to HTML+JS)
    - Linting (syntax warnings, hints, and help)
    - A **development server**
        - With Live reload!
        - ONLY for development, not final use
- Vite isn't required for React, but is convenient

A Note about Create React App

Course previously used `create-react-app` (CRA)

- A lot of tutorials/docs on web will refer to CRA
    - Common starting point for React SPAs
- Over time
    - CRA got slower to install/use
    - Alternatives got more attention
    - Alternatives were good for more than SPAs

Course now uses `vite`

- Still SPA-focused
- NextJS, Remix are more involved alternatives

Create a test app

```
npm create vite test-app -- --template react
```

Tells NodeJS to download and run create-vite

- Creates folder holding app "test-app"
- You can give any name you want

Creates a `test-app/` directory

- Where you run the app
- Puts in all the pieces
- You are not "in" that directory yet

Our new app

Vite installed and configured a lot

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

```
cd test-app
npm install
npm run dev
```

Umm…neat?

It started a server and is showing a page

- You can inspect the HTML

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

- Leave the server running

Opening src/App.jsx

This looks like a mix of JS and JSX

- imports
    - Some we know, some we don't
- function App() returns HTML
    - Not as a string, just HTML
    - Has some values in `{}`
    - Uses `className` instead of `class`
    - There's an `onClick`

Now look at HTML for the page in DevTools

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
- `{count}` was replaced with a number

Now make a text change to `App.jsx` and save

Live Reloading

Change shown in browser without manual reloading!

App.js **imports** App.css

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

.jsx files

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

- For this course **you must use** `.jsx`
- Filename is extra information for coders
  - `.js` files should have NO JSX in them
  - `.jsx` files should be our view files
  - JSX is for UI, other logic is plain `.js`
    - Separates UI logic from **business logic**
    - Separates UI from sending/getting data

A word about the default file

- They use `target="_blank"`
    - You should NOT do this: **https://css-tricks.com/use-target_blank/**
    - It denies the user the choice
- React brings new options to organize CSS
    - CSS-in-JS, CSS Modules, etc
    - CSS-in-JS looks to be fading
        - And we are learning so much already
    - Continue our **existing CSS conventions**

About default file contents

Example code in `App.jsx` does NOT use semicolons

- Course still **requires semicolons**
- You should add any that are missing
- Most contents of App.jsx will be replaced
    - This is example content
    - You define the App() function
    - And what App() returns
    - And any necessary imports
    - And export App

Where is the HTML?

The HTML is in `/index.html`

- BUT we won't be changing it
    - Except for anything in `<head>`
        - In particular, `<title>`
        - But also webfonts, more meta tags, etc
- Make all your changes in the js/jsx/css files in `src/`
    - `src/` for the files you edit!
    - These are NOT loaded by browser directly
        - Get **transpiled** into files for browser

Where is the CSS?

The `import` brought in the `src/App.css` file

- You can import additional/different css
- CSS filename(s) do not need to be Capitalized
  - But you can, to match the JSX file, if you want
  - No course requirement on the CSS filenames

There is also a `src/index.css`

- General page/element defaults
- vs `App.css` which styled the content of `App.jsx`
- Similar to what we've done
  - But broken up into separate files!

What did Vite do?

- Created `package.json`
- Created a `vite.config.js`
- Created `public/`
- Created `src/`
- Created `index.html`
- Created `.gitignore`
- Created `.eslintrc.cjs`

`package.json`

- All the dependencies/devDependencies for vite/react
- Added `scripts:`
    - `dev` (`npm run dev`) - Development server
    - `build` (`npm run build`) - Creates built/bundled/static files
    - `lint` - Runs linter (syntax checker)
    - `preview` - Runs static server for built files

`public/`

- Not quite like our webpack/express config
- Files that aren't changed during build
- NOT our document root!
- NOT for `.jsx` files
- If we import from `/`, it imports from `public/`

`src/`

- Very similar to our webpack setup
- Almost all files we edit are here
- Our start point is `src/App.jsx`
- Technical start point is `src/main.jsx`
    - Not MixedCase because not a component
    - Loaded via `index.html`
- Import from via relative, explicit paths`
    - `import Test from './Test';`

`index.html`

- The basic HTML skeleton
- Loads `src/main.jsx`
- We won't edit the `<body>`
- We WILL edit the `<head>`
- Not in `src/` or in `public/`

`.gitignore`

- Vite creates a `.gitignore` file
- Blocks `node_modules/` or built files
    - We like this!
    - Don't mess it up!
- Only blocks from THIS project
    - This folder or subfolders

`.eslintrc.cjs`

- Configuration for **linting**
    - Syntax checking for preferences
- You should modify:

```
  rules: {
    'react-refresh/only-export-components': 'warn',
    'react/prop-types': 'off',  // ADD THIS LINE NEAR END
  },
}
```

- prop-types aren't BAD, we just aren't covering them

Building

`vite` is a tool to help develop

- In the end we want static HTML/JS/CSS
- We can put those on ANY server
    - `npm run dev` is NOT a production server

Stop your server (Ctrl-C)

- Then run `npm run build`

What did that do?

We now have a `dist/` directory

- Contains static HTML/CSS/JS files
    - Plus some images
- Files have weird names
    - Cache-busting
    - Different content = different filename

These files are ALL you need

- Can put on ANY static webserver
- No Vite, no special programs

When do we build?

Do all your development with the development server

- Edit files in `src/`
- Uses `npm run dev` to run

If done and putting up web app for the public

- Then `npm run build`
- Use files inside `dist/` with your webserver
  - Such as `npx serve`, or Java, or C#, etc
  - Or our `server.js`! (more soon!)

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 values
- Can have non-strings (unlike HTML)
- All elements must close

Summary - Vite

Vite is a program that makes React easy to use

- Just one way to use React
- Includes a development server
    - NOT for production (final) use

Vite creates a directory for the app

- `npm create vite APP_NAME -- --template react`
- Start dev server with `npm run dev`
- Build prod files with `npm run build`

Summary - Editing

Edit files in `src/`

- **Course Requirement**: use semicolons
- **Course Requirement**: kebab-case/BEM classes
- Default `App.jsx` contents should be replaced!
    - Just an example
- Can rename/replace or just use `App.css`
    - `import` needed css file(s)
- Also use `src/index.css`
    - Replace/change any `.css` contents as needed