SPA (Single Page Application)

- Load an HTML Page
- HTML loads JS
- JS keeps replacing HTML contents
    - Looks like changing "pages"
    - No actual browser navigation
    - All on a "single-page"

Can make a SPA with plain JS

- Most SPAs make use of "service calls"
    - JS sends/gets data from server
    - No Page navigation/loads
- This course doesn't dive into service calls
    - My 6250 class DOES go into them
    - Other 6250 classes may not
- We haven't written SPAs until React
    - SPAs don't need to be React

Web Page/App/Site

Page

- Single load in browser
- May be .html file, may be server generated HTML

App

- User sees many pages (browser may or may not)
- Any mix of server generated/JS changed HTML

Site

- Any number Pages/Apps (incl just 1)
- Any mix of .html files and server generated HTML

User Experience of Page/App/Site

- Want consistent branding/"feel"
- May have different departments/units
    - Sales/Support
    - University Colleges/Programs
- Even when HTML generated by different systems
    - May have multiple sites
    - Each with mixed in Apps and Pages
    - Users largely don't care

Story Time! - What is a Website?

One place I worked:

- I was in charge of the general "website"
- Also wrote some "web apps" on that site
- A team in a diff org wrote specialized "web apps"

We argued often over urls and site appearance

- I (and designer) wanted one site "look"
- I (and designer) wanted url to reflect overall site
- Other teams didn't even understand those ideas

State

Previously we could alter HTML/classes "in place"

- What I call "state-render" loop was optional

State and Render

- "state" is in variable(s)
    - Ex: A list of news stories
- User actions change the "state" variables
    - Ex: "like" a story, block another
- Actions can trigger from the system too
    - Ex: more, newer list of news stories
- When state variables change, page "renders"
    - = Updates HTML to match new state
- **HTML based on current state, not on this action**

We have focused on UI/UX so far

- Makes sense, this is a UI/UX class
    - Involved UI complex without state/render
- This state/render cycle is FUNDAMENTAL to React
- State: variables with all values that can change
- Actions: Change the state
    - Do NOT change the HTML
    - Ex: user events can run actions
- Render: Replace the HTML to match new State
    - Separation between event and render (!)

Results of the state/render cycle

- State/render can feel like more work
- But as we add features, complexity remains level
- Old way would get more complex over time
    - Hard to make decisions based on state
        - No state variable, have to read HTML
        - Changing HTML changes how to read it
        - Every new feature changes HTML

We don't need to code a state/render cycle!

- This is a UI/UX class
- React will do this for us
- But must understand the concept

State describes current data

- NOT HTML, just data

We change state, not the HTML

- HTML updates to new state
  - This is our UI/UX skillset!

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 in what looks like HTML
<main>
  <Greeting/>
</main>
```

NOT JS, but JSX

- Browser can't handle without translation
- Much friendlier to use
- Output is HTML and JS
- JSX not a string in quotes!

More JSX Example

```
function Cat({ name }) {
  return (
    <div className="cat">{name}</div>
  );
}
//...elsewhere
<Cat name="Jort"/>
```

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 Cat({ name, isNapping }) {
  let className = 'cat';
  if (isNapping) { className += ' cat--nap';
  return (
    <div className={className}>{name}</div>
  );
}
//...elsewhere
<Cat name="Jort" isNapping={true}/>
```

A few differences!

- Plain JS before `return`
- `{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
- Not everything is React

Vite

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

- Someone else has done 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 Client-Side 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

- Some weird `import` statements
- 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
    - Can be `.js` files that have no JSX in them
    - 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**

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!

Where is the HTML?

The HTML is in `/index.html`

- We only change contents of `<head>`
    - In particular, `<title>`
    - Also webfonts, meta tags, etc
- All other 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

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 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

- 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

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

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