# 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? `¯\_(ツ)_/¯` )
  - 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

Imagine automatic re-rendering when state changes

- React will do that for us!

Also only changes DOM

- Where changes actually happen

# 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 (`jsx` - the X is XML)
- Declarative
- Actually a JS function that returns HTML
- Can call other JSX functions for HTML
- Can insert HTML
- NOT a text string!

# JSX Example

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

JSX NOT js, NOT text string

- Browser can't handle without translation
- Friendlier to use
    - Mimics HTML output
    - Like a function call
- 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
  - No template literals (``) here
  - No quotes!
  - 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 more 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`
- React is managing our DOM
- If we change it, we can confuse React

Why did we learn those parts then?!

- Basic skill, used often
- Know what React is doing
- Good without React

# Summary - React

React will let us auto-render when state changes

React uses JSX

- JS that looks like HTML
    - All tags must close
- 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 JSX files
  - For this course: don't need to load CSS in each component