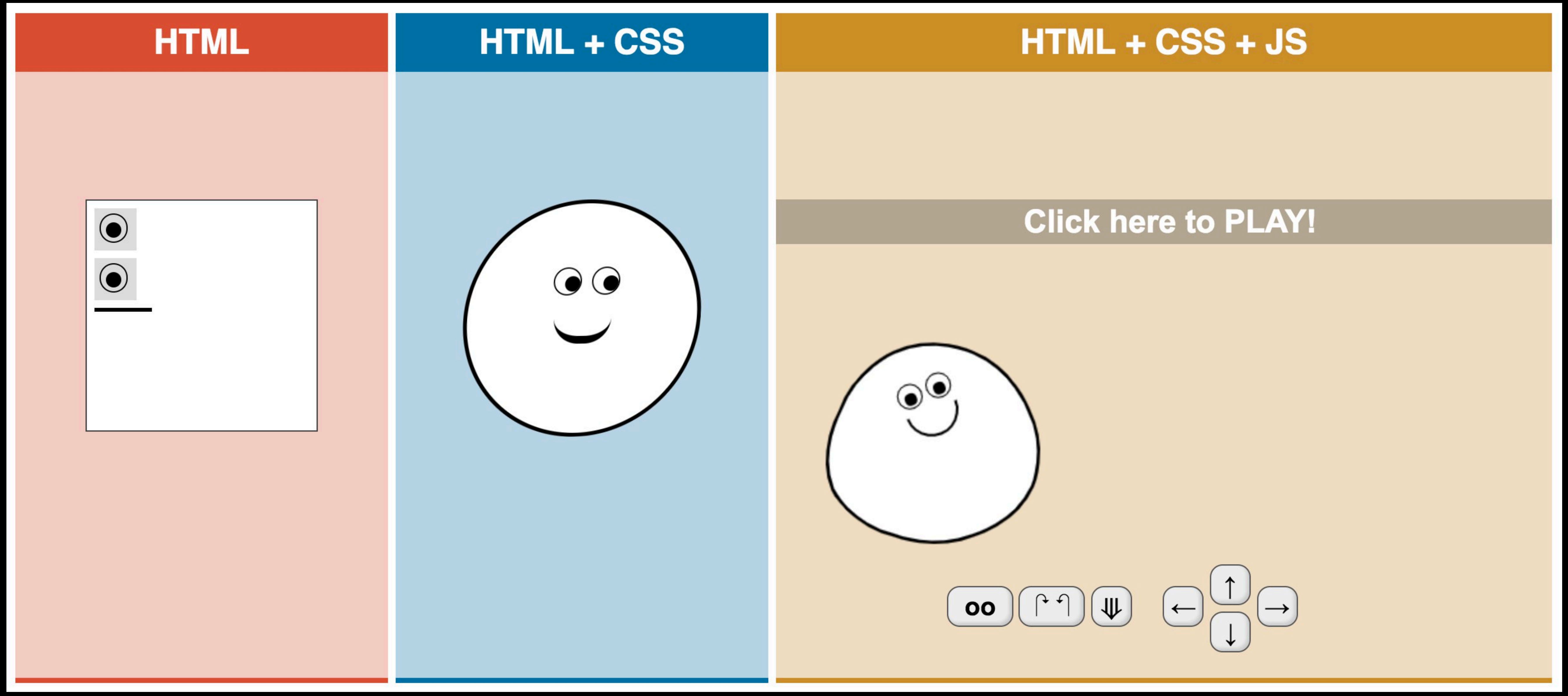
Coding

Front-end Web App

HTML + CSS + JavaScript

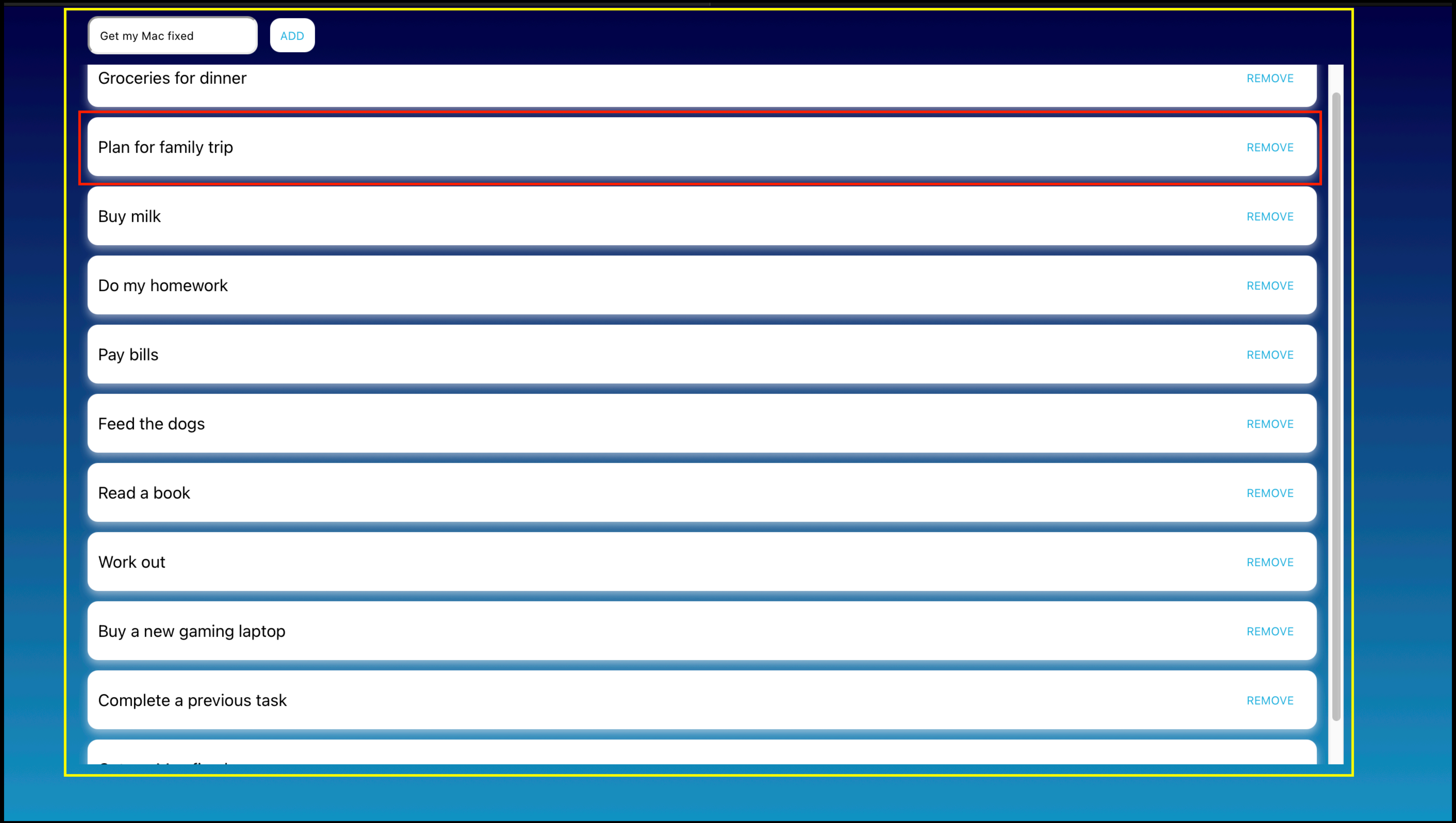
- HTML: Skeleton of the website
- CSS: Skin of the website
- JavaScript: Spirit of the website



Vanilla JavaScript is not enough

- User interface is getting more and more complicated
- e.g. **62 million** lines of code (Facebook)

Component-based

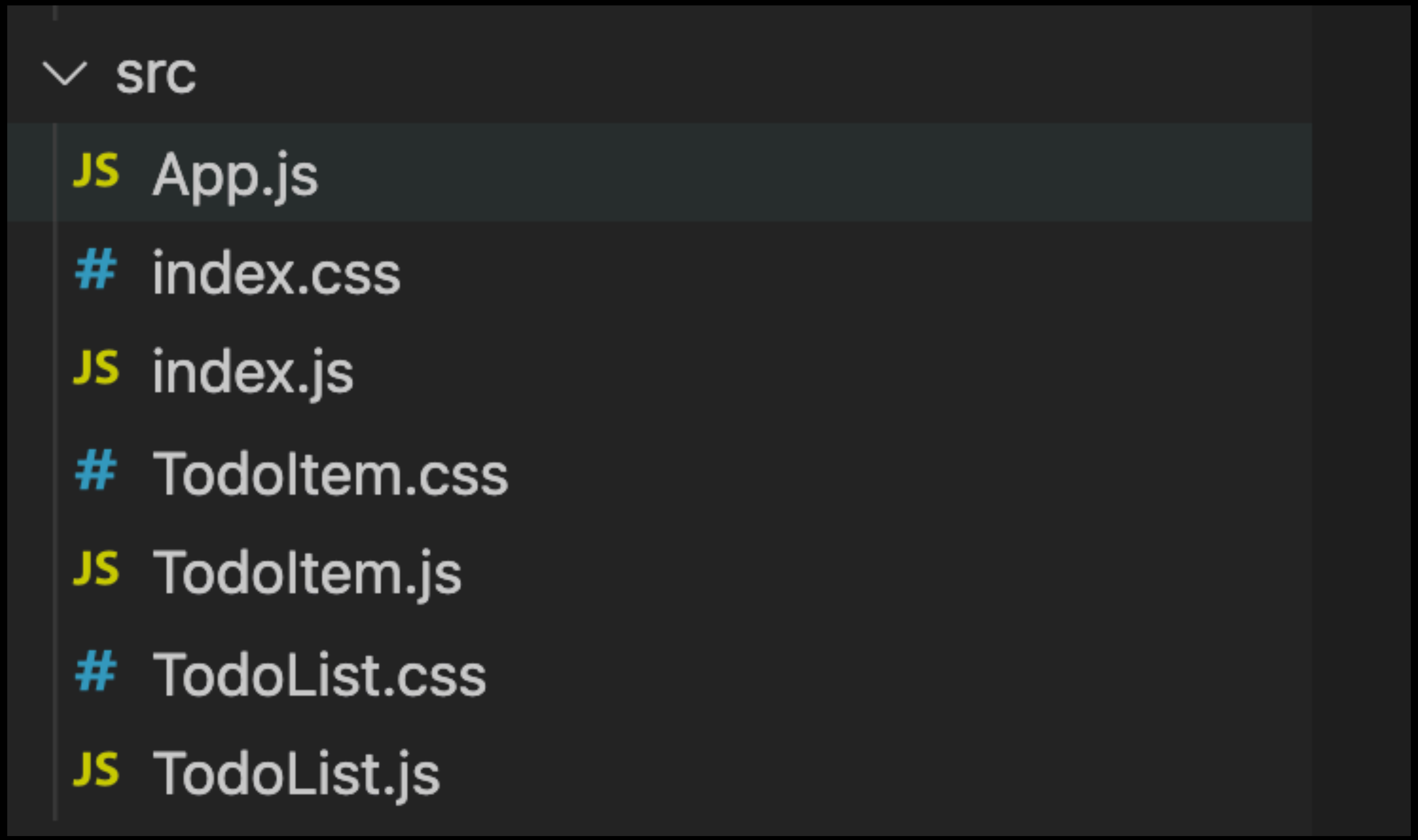


Get my Mac fixed

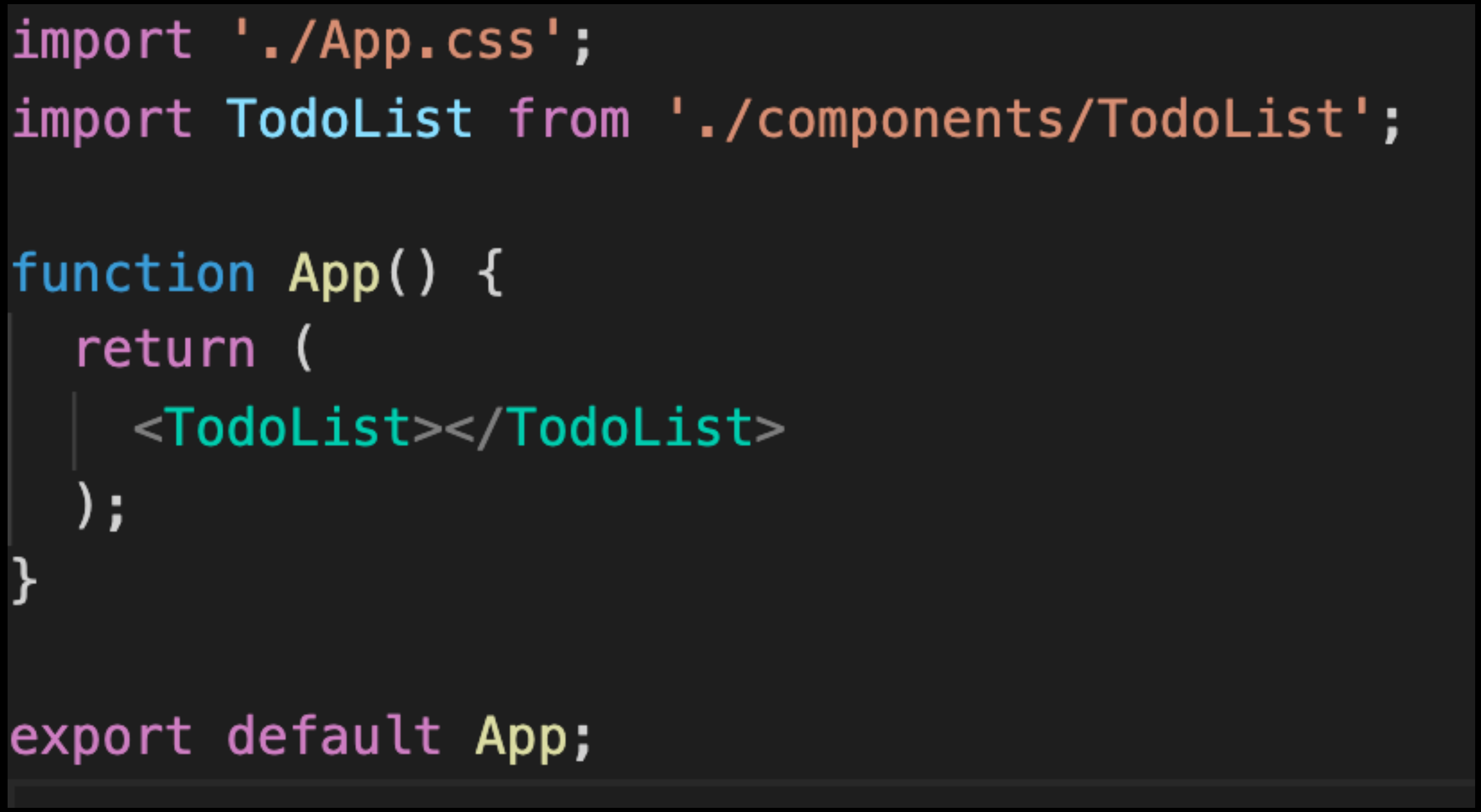
- Groceries for dinner REMOVE
- Plan for family trip REMOVE
- Buy milk REMOVE
- Do my homework REMOVE
- Pay bills REMOVE
- Feed the dogs REMOVE
- Read a book REMOVE
- Work out REMOVE
- Buy a new gaming laptop REMOVE
- Complete a previous task REMOVE

https://wereheavyweight.medium.com/how-were-using-component-based-design-5f9e3176babb

Component-based



```
src
  JS App.js
  # index.css
  JS index.js
  # TodoItem.css
  JS TodoItem.js
  # TodoList.css
  JS TodoList.js
```

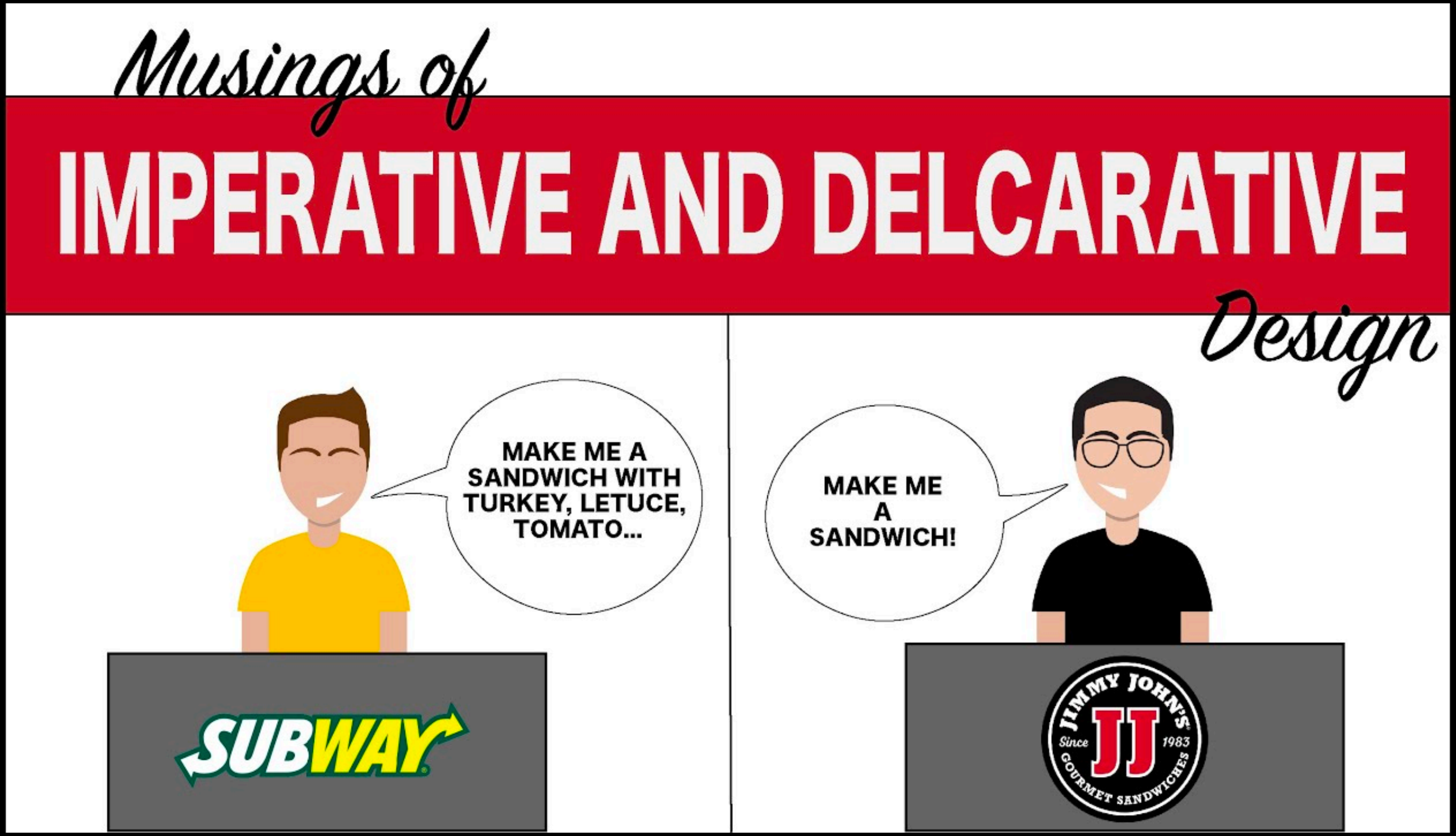


```
import './App.css';
import TodoList from './components/TodoList';

function App() {
  return (
    <TodoList></TodoList>
  );
}

export default App;
```

https://codepen.io/yi-hung-chou/pen/PomqEyX

Declarative when render views

- Imperative (Vanilla JavaScript) v.s. Declarative (React)


Musings of
IMPERATIVE AND DELCARATIVE
Design
MAKE ME A SANDWICH WITH TURKEY, LETUCE, TOMATO...
MAKE ME A SANDWICH!
SUBWAY
JIMMY JOHN'S Since JJ 1983 GOURMET SANDWICHES

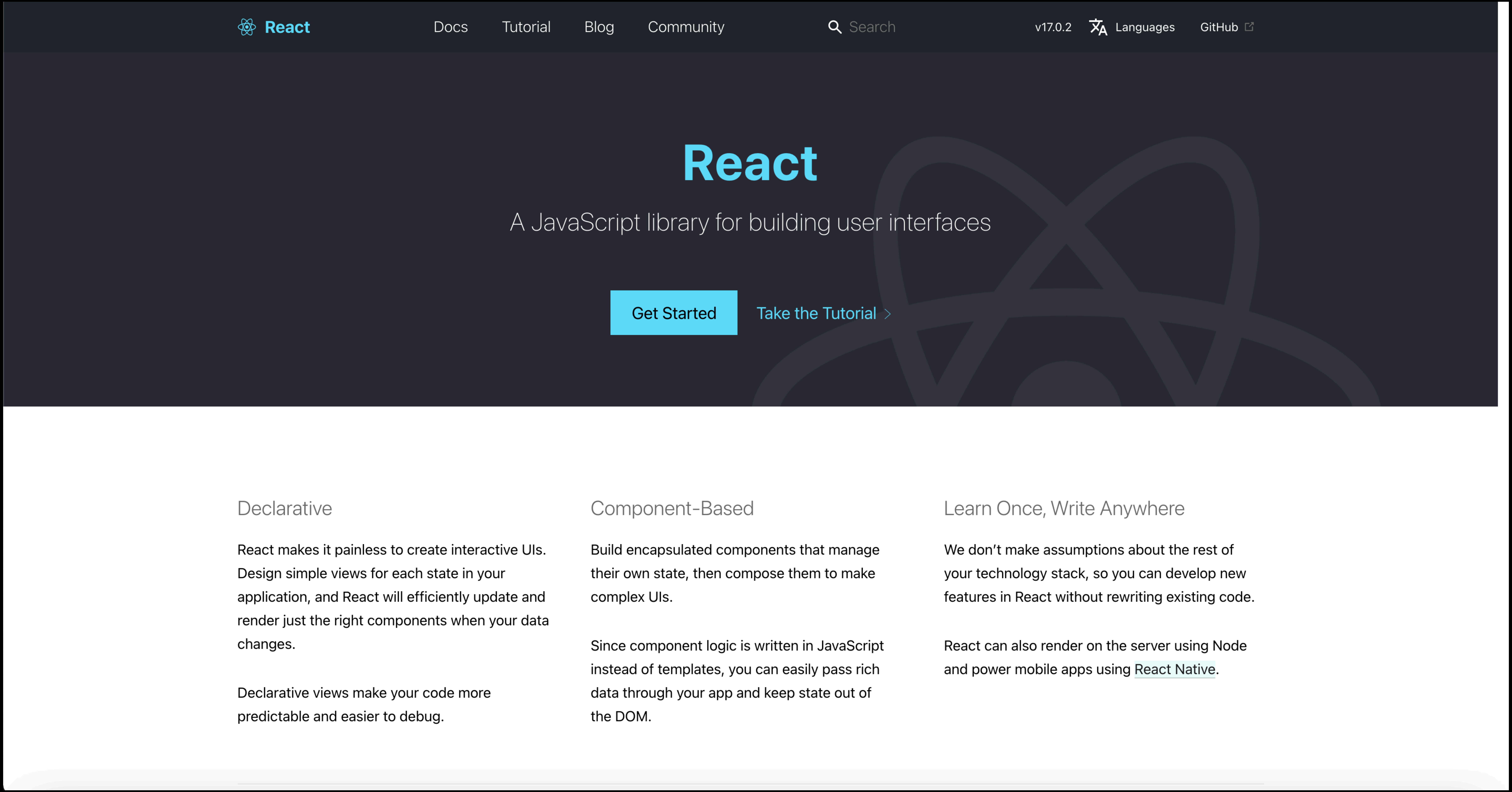
https://www.wikihow.com/Order-a-Subway-Sandwich

Declarative when render views

	Vanilla JavaScript	React
todoItems = empty list	const todoItems = []	const [todoItems, setTodoItems] = useState([])
Describe the view	In HTML: <div id=“todo-items”></div>	<div>{todoItems}</div>
Update the view accordingly	updateViewWithData(todoItems)	X
When user click the button: Update data	todoItems.push(newItem)	setTodoItems([…todoItems, newItem]);
Update the view again	updateViewWithData(todoItems)	X

Learn once, write anywhere

- Easlily integrated with legacy code
    - CrowdNews
- Portable
    - ReactNative (Mobile App)
    - Electron + React (Desktop App)
    - ReactVR (Virtual Reality)



React

A JavaScript library for building user interfaces

Declarative

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

Component-Based

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

Learn Once, Write Anywhere

We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using React Native.