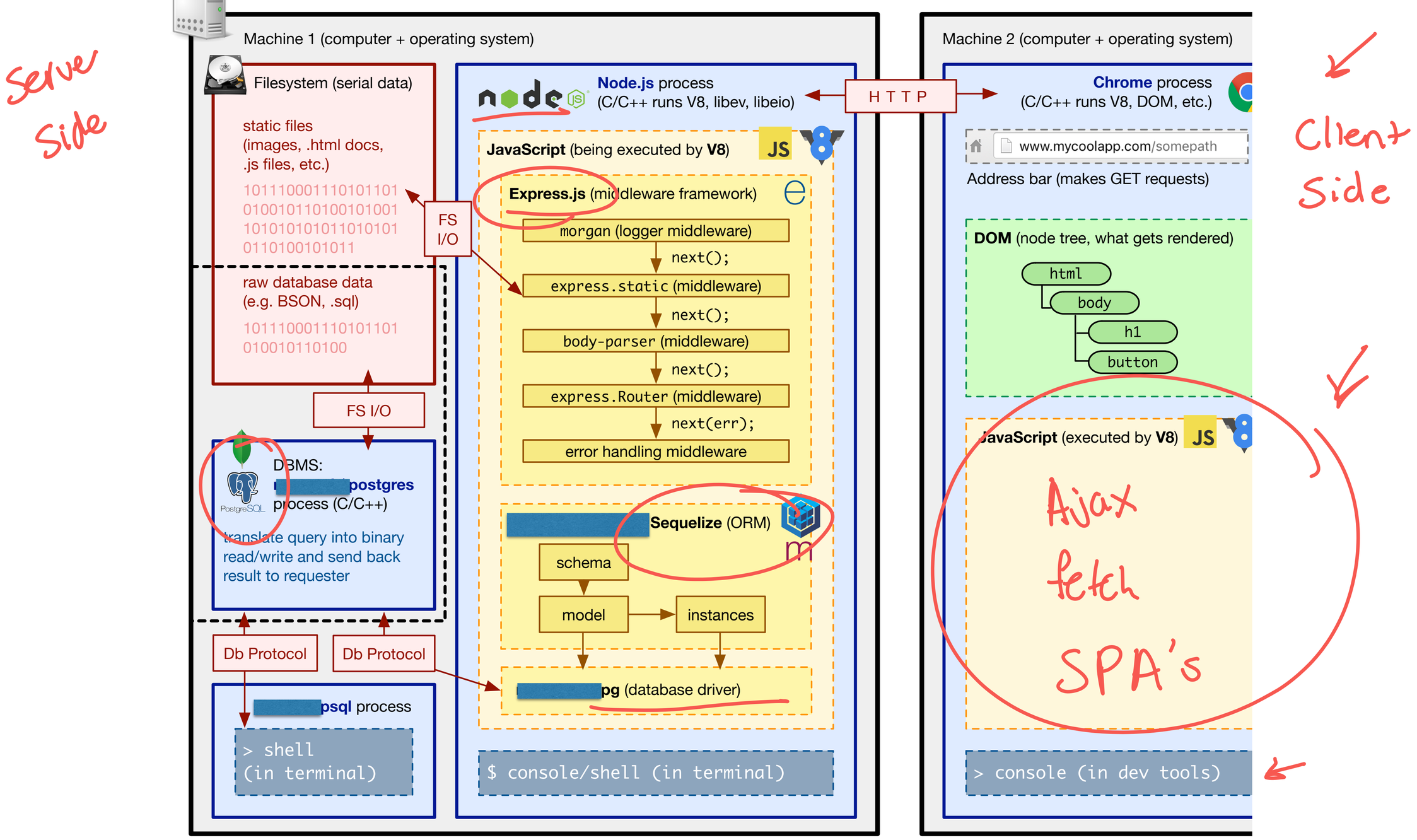
FULLSTACK
ACADEMY

Front-end with React.js



Wikistack Front-End

```
module.exports = (page) => layout(html`
  <h3>${page.title}
      <small> (<a href="/wiki/${page.slug}/similar">Similar</a>)</small>
  </h3>
  <h4>by <a href="/users/${page.author.id}">${page.author.name}</a></h4>
  <ul>
    ${page.tags.map(tag => html`<li>${tag}</li>`)}
  </ul>
  <hr/>
  <div class="page-body">$${marked(page.content)}</div>
  <hr/>
  <a href="/wiki/${page.slug}/edit" class="btn btn-primary">edit this page</a>
  <a href="/wiki/${page.slug}/delete" class="btn btn-danger">delete this page</a>

`);
```

HTML

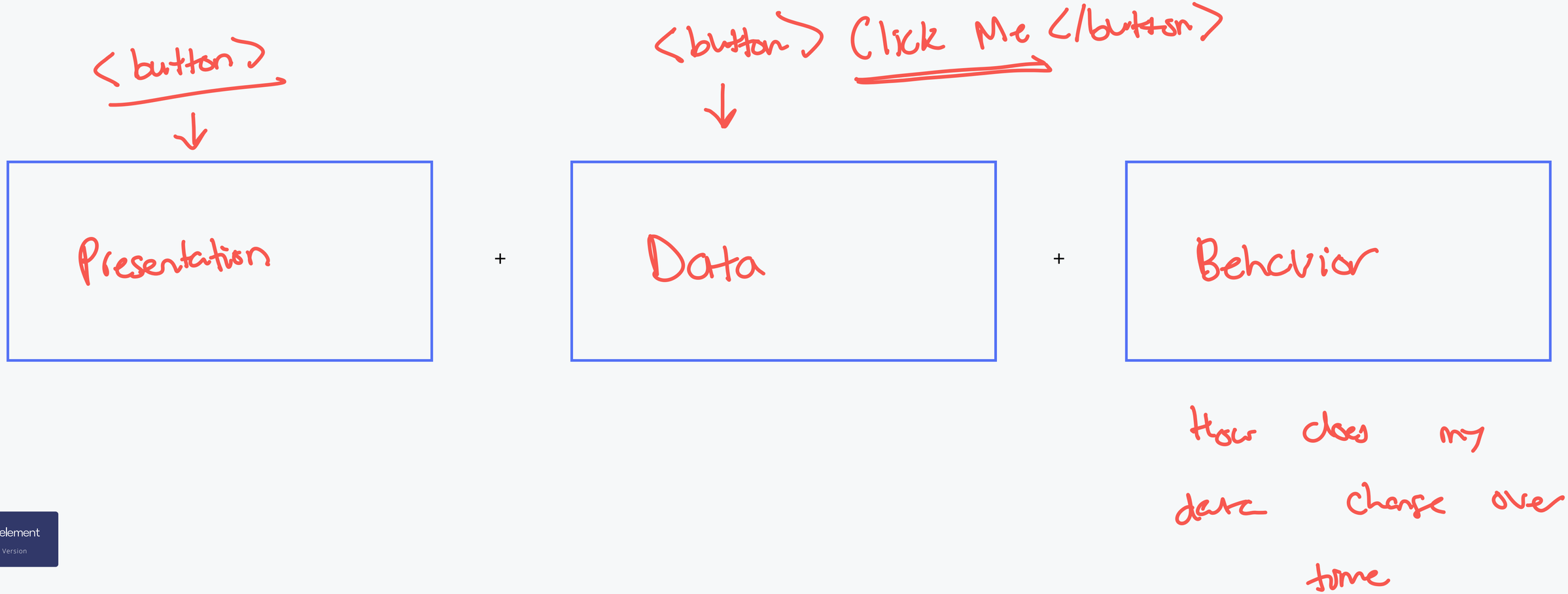
Trip-Planner

```
export function addItineraryItem (attraction) {

    //Make a dom element and append

    const itineraryItem = document.createElement('li');
    itineraryItem.className = 'itinerary-item';
    itineraryItem.append(attraction.name);

    //Some other function for making my remove button
    const removeButton = makeRemoveBtn(itineraryItem, attractio
n)
    itineraryItem.append(removeButton)


    //more DOM append logic, etc. etc.
  }
```

DOM manipulation

Making a user-interface is hard

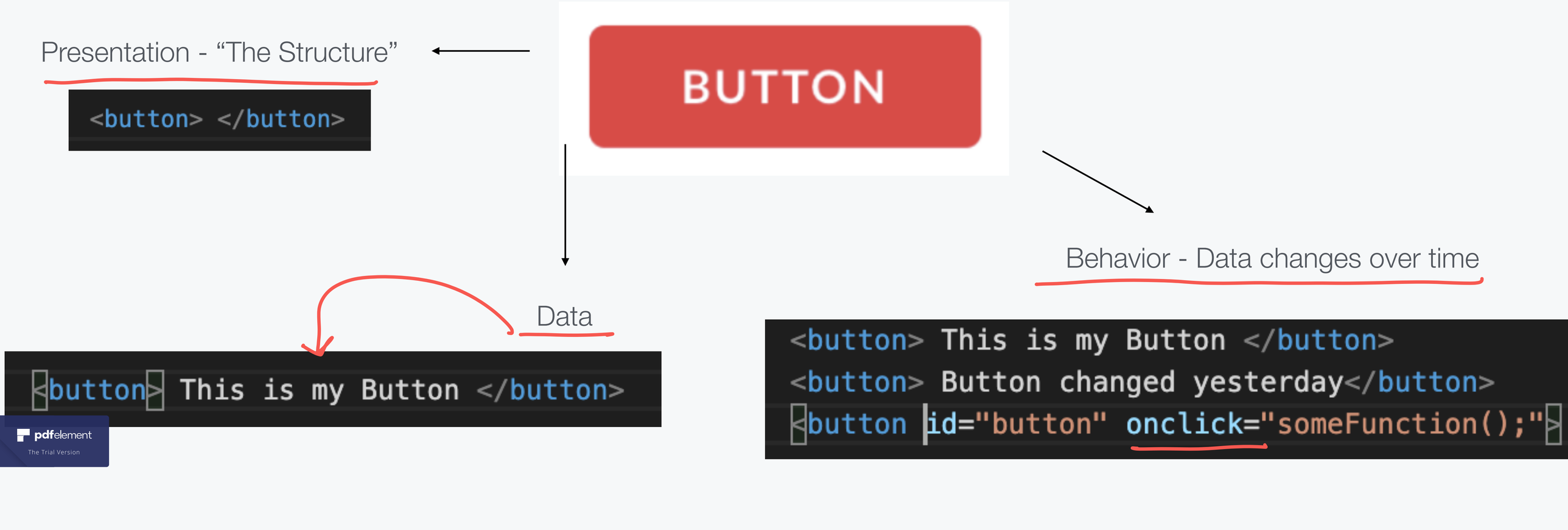
Historically, managing all the code for our front-end has been a nightmare for developers….



What if....

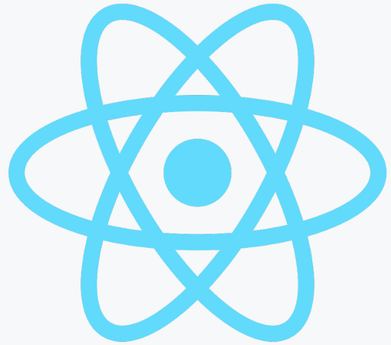
Front-end engineers brainstormed how can we make this better?

How can we make it more manageable?



Introducing React.js

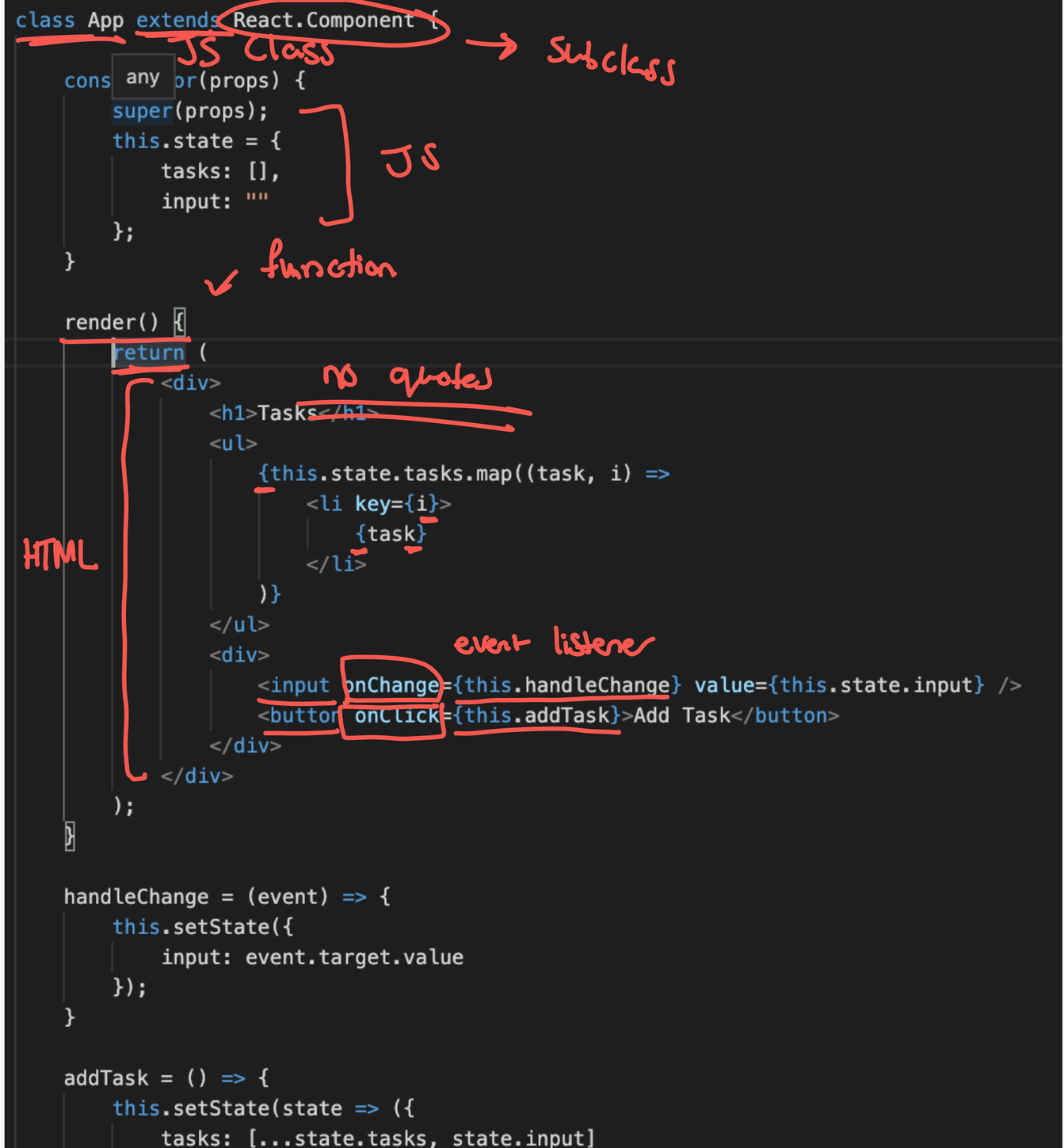
Our Front-End library of choice.



- Declarative (vs Imperative)
- Components (& props)
- JSX
- Component State Management (& Immutability)

A React.js preview

Note all the things that look familiar



Live Code - React.js

Component Types

In React.js there are two ways to write a component

Class

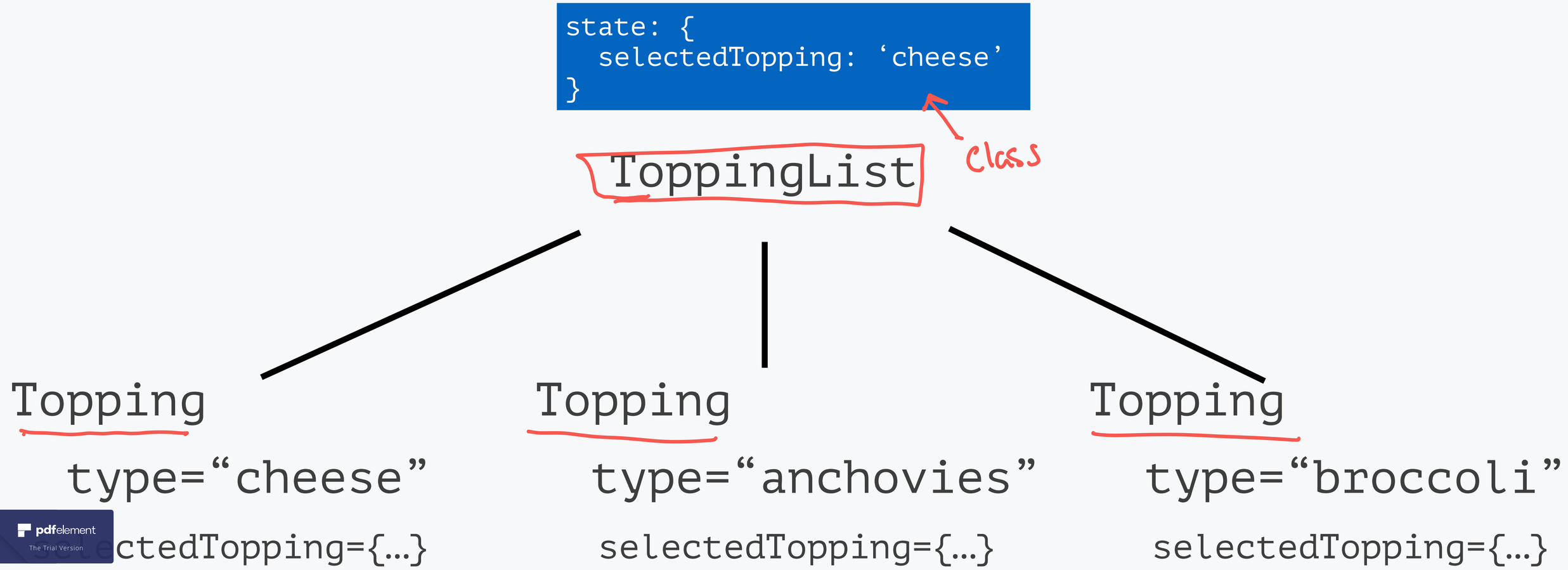
```
class Pizza extends React.Component {
  render () {
    return <div>Pizza Pie!</div>
  }
}
```

Functional

```
const Pizza = () => {
  return <div>Pizza Pie!</div>
}
```

State & Props

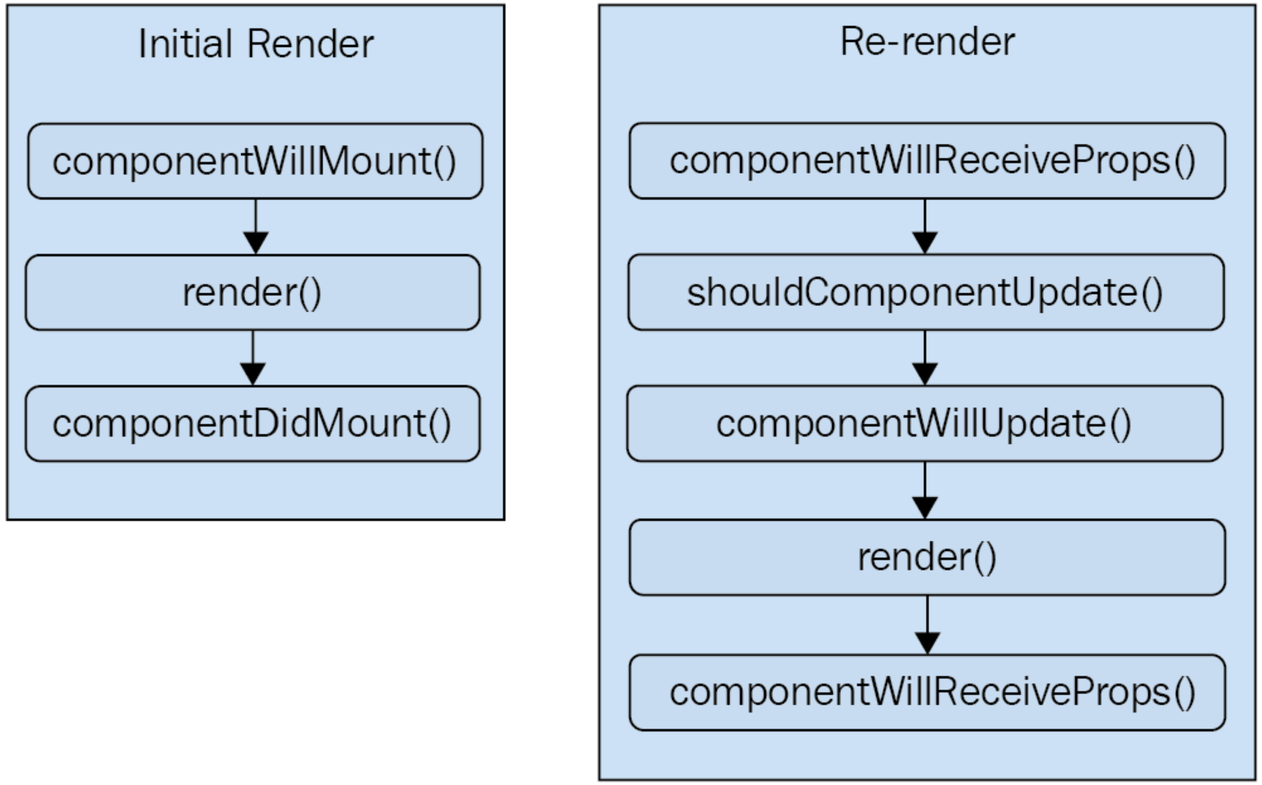
State and props of components always take a **uni-directional data flow**. They *almost* always flow down



Component LifeCycle

When we render a component, React components go through several stages in addition to the “render” stage

Note: setState causes a re-render of the component



Live Code - Pizza App