React Components

A FRONTEND LIBRARY

Using bootstrap in react app

>npm i bootstrap

>npm install font-awesome

Then you can use

Use below code in either index.js or in any component

```
import "bootstrap/dist/css/bootstrap.css";

import "font-awesome/css/font-awesome.css";
```

Function and Class Components

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Using a Component

<Welcome name="Sara" />

Attributes passed to a component are props

Props are read only

Don’t be afraid to split components into smaller components.

HTML ? FancyBorder Tag?

```
<FancyBorder color="blue">
        <Welcome name="Sara" />
</FancyBorder>
```

Composing Components

```
function App() {
  return (
    <div>
      <Welcome name="Sara" />
      <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>
  );
}
```

Welcome Component

```
import React from "react";
const Welcome= (props) => {
  return <div>Welcome {props.name}!</div>;
};
export default Welcome;
```

FancyBorder

```
import React from "react";
const FancyBorder = (props) => {
  return (
    <div style={{ border: "1px solid", borderColor: props.color }}>
      {props.children}
    </div>
  );
};
export default FancyBorder;
```

Welcome Component inside FancyBorder

```
<FancyBorder color="blue">
        <Welcome name="Sara" />
</FancyBorder>
```

Using a Component

```
 import Welcoome `from
"./components/examples/Welcome";

 ReactDOM.render(<Welcome name=“Usman”/>,
document.getElementById("root"));
```

//instead of using App we are using Welcome

Using a Component inside another

```
import React from "react";
import Welcome from "./Welcome";
const ManyWelcome = () => {
  return (
    <div>
      <Welcome name="Usman" />
      <Welcome name="Hassan" />
     </div>
  );
};
export default ManyWelcome;
```

Component can return only a Single Element

```
import React from "react";

const Welcome= (props) => {

  return <div>Welcome {props.name}!</div>;

};

export default Welcome;
```

Embedding Expressions

```
const count=0;
    return (
      <div className="App">
        {count}
      </div>
    );
```

If Short Hand

{count === 0 ? "Zero" : count}

{count !== 0 && count} //print count if not zero

State

```
const [count, setCount] = React.useState(5);

//A special variable to control Component State
```

React will remember its current value between re-renders,

Provide the most recent one to our function.

Update the current count, we can call setCount.

Counter With State

{count < 5 && Count is in dangerous state }

{count < 5 ? Red Counter : Green Counter }

Object Destructuring

```
var o = {p: 42, q: true};
var {p, q} = o;

console.log(p); // 42
console.log(q); // true
```

A Clean Approach

```
formatCount() {
const count=7;
return count === 0 ? "Zero" : count;
    }
      return (
        <div>
          <span>{formatCount()}</span>
        </div>
      );
```

formatCount can also return jsx expression

```
formatCount() {
  const count=5;
  return count === 0 ? "Zero" :
    <span> {count}</span>;
}
```

Setting Attributes

```
return <img src={imageURL} alt="" />;
```

Setting CSS Classes

```
const count = 0;
    return (
      <div className="App">
        {count === 0 ? "Zero" : count}
        {count !== 0 && count}
      </div>
    );
```

Setting Styles from jsx

```
styles = {
   fontWeight: 'bold'
 }
const count = 0;
   return (
     <div className="App" style={styles}>
       Hello
     </div>
   );
```

Dynamic Classes

```
getBadgeClasses() {
    let classes = "badge m-2 badge-";
    classes += count === 0 ? "warning" : "primary";
    return classes;
  }
//Then in component
//className={getBadgeClasses()}
```

Handling Lists

```
consttags: ['Funny', 'SciFi', 'Horror'];
return (
 <ul>
 {tags.map(tag => <li>{tag}</li>)}
 </ul>
);
}
//This will generate Warning. Each li should have a key
```

Specifying Key

```
const numbers = [1, 2, 3, 4, 5];
const listItems = numbers.map((number) =>
  <li key={number.toString()}>
    {number}
  </li>
);
```

Specifying Key

```
const todoItems = todos.map((todo, index) =>
  // Only do this if items have no stable IDs
  <li key={index}>
    {todo.text}
  </li>
);
```

Conditional Rendering

```
renderTags() {
    if(tags.length===0) return <p>No Tags</p>;
    return tags.map(tag => <li>{tag}</li>);
  }
//Then in Render
//{tags.length === 0 && "Please Create Tags"}
//{renderTags()}
```

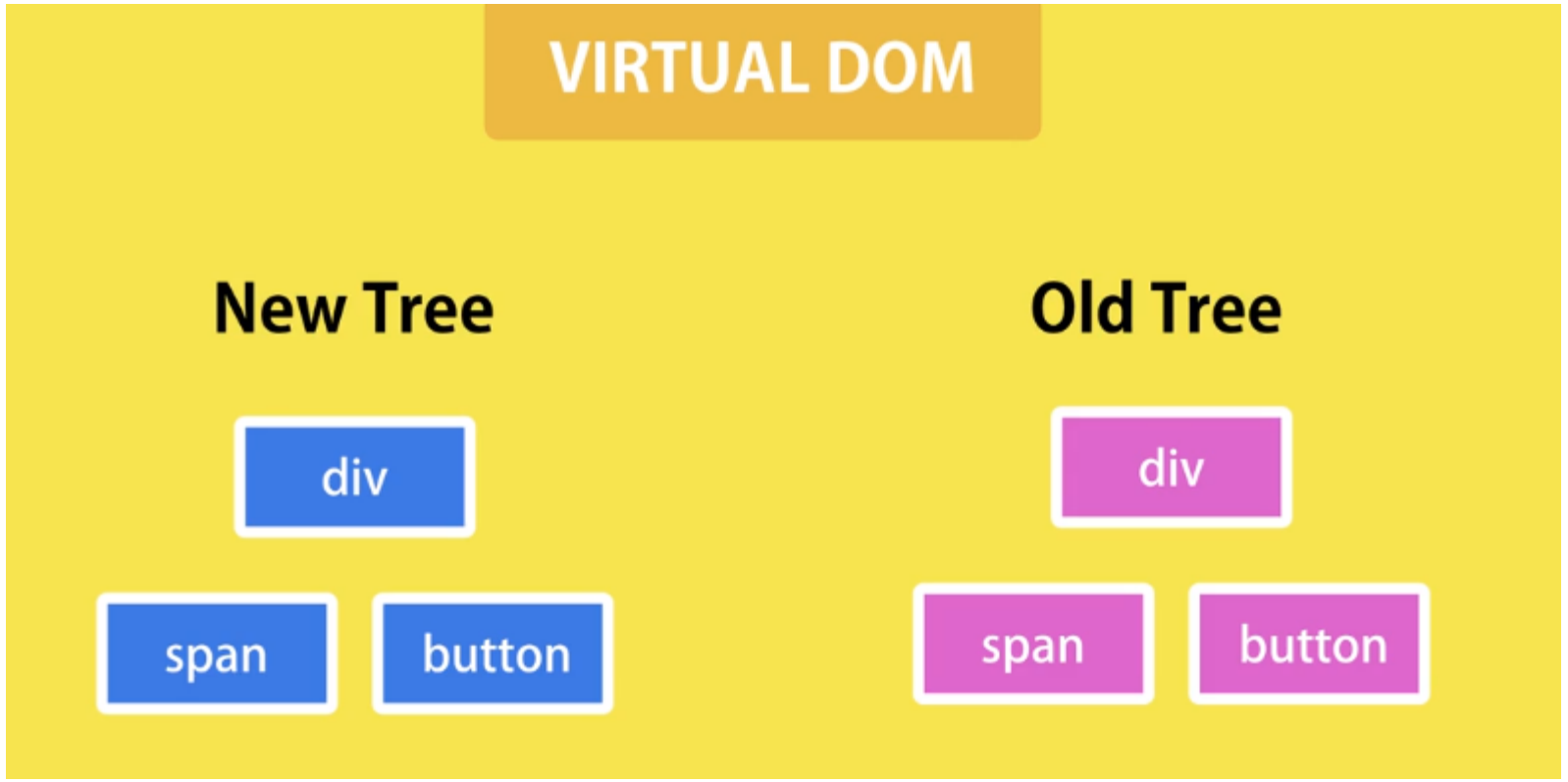
Handling Events
Raising an Event

```
<Button variant="contained" color="primary" onClick={countUp}>
    <AddIcon />
</Button>
```

Handling Events
Capturing an Event

```
const countUp = () => {
    setCount(count + 1);
    // count++;
    // alert(count);
  };
```

What Happens when state change



Passing Parameters to event handler

```
handleIncrement = (id) => {
//Handle ID
};
```

Call it like this

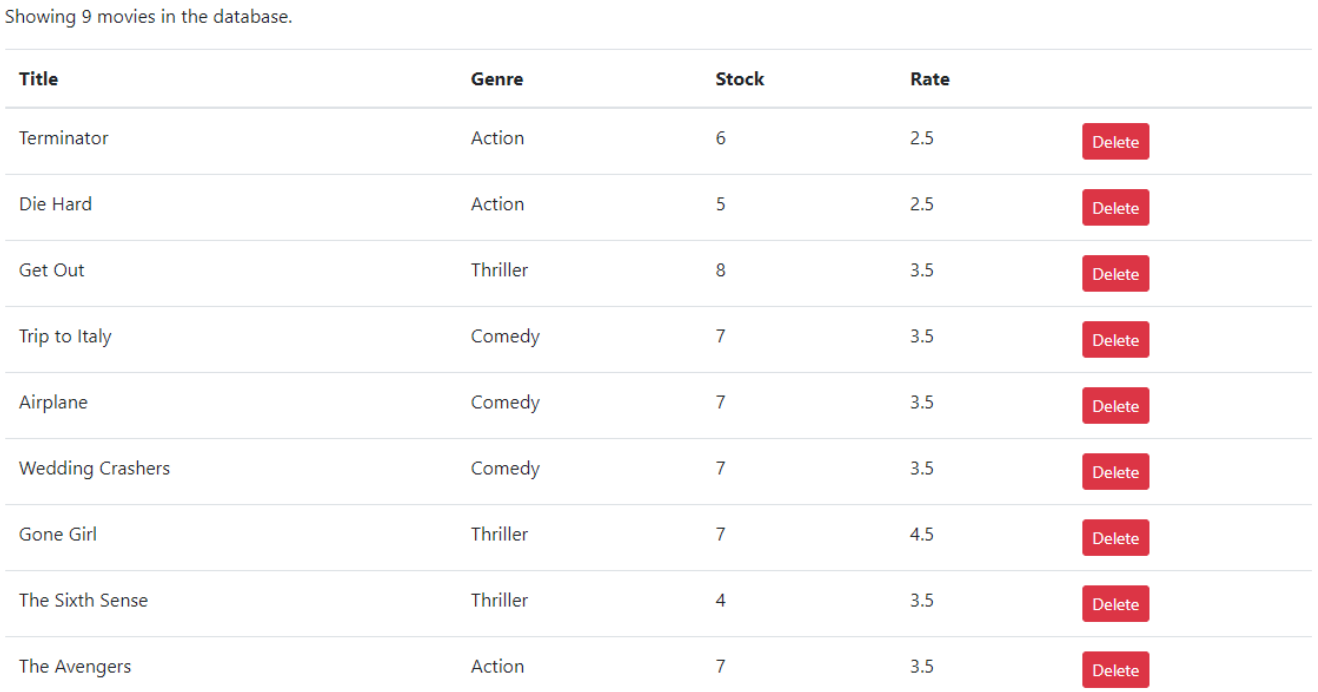
```
onClick={()=>handleIncrement(3)}
```

Vidly Project

https://1drv.ms/f/s!AtGKdbMmNBGd1GXjgiJoFiImccTx

>npm install

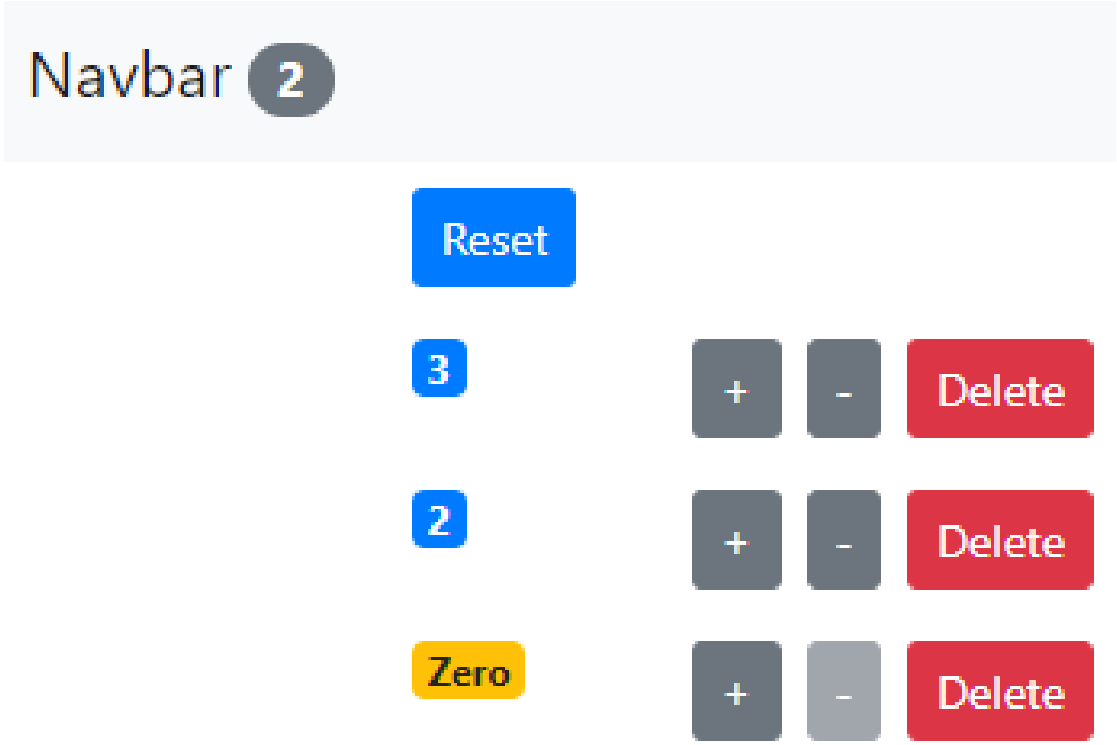
>npm start



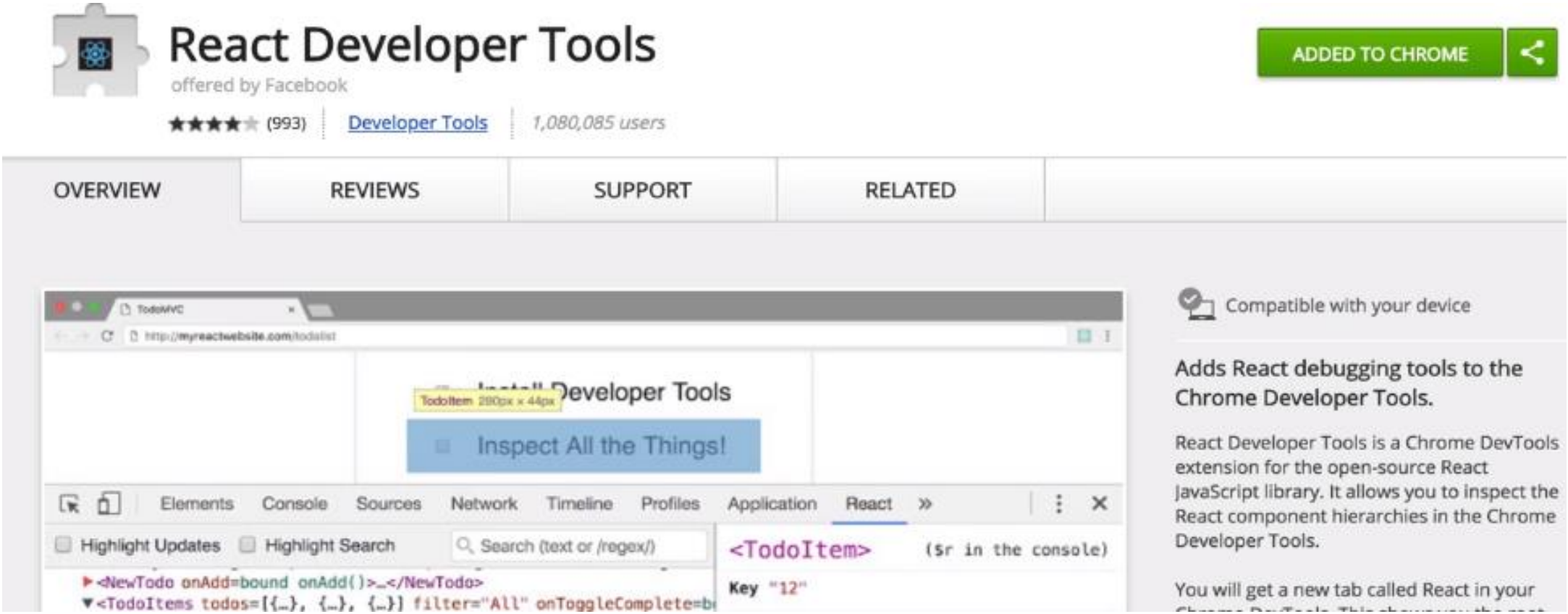
Composing Components

https://1drv.ms/f/s!AtGKdbMmNBGd1zi0xRJmY1kA7tqt

Start to Finish



React Debugging



Single Source of Truth

Try to make Stateless components as much as possible.

Data should be kept at only one location

E.g. Movies Components has many SingleMovie Components then movies data should only be placed in parent .

Passing Functions as props

```
<Like
        liked={movie.liked}
        onClick={() => handleLike(movie)}
      />
```

Calling Passed Function (Like.jsx)

```
<i
    onClick={props.onClick}
    style={{ cursor: "pointer" }}
    className={classes}
    aria-hidden="true"
  />
```