social computing

UNIVERSITÄT DUISBURG ESSEN

Open-Minded

React

Lecture 1

Prof. Dr. Mohamed Amine Chatti

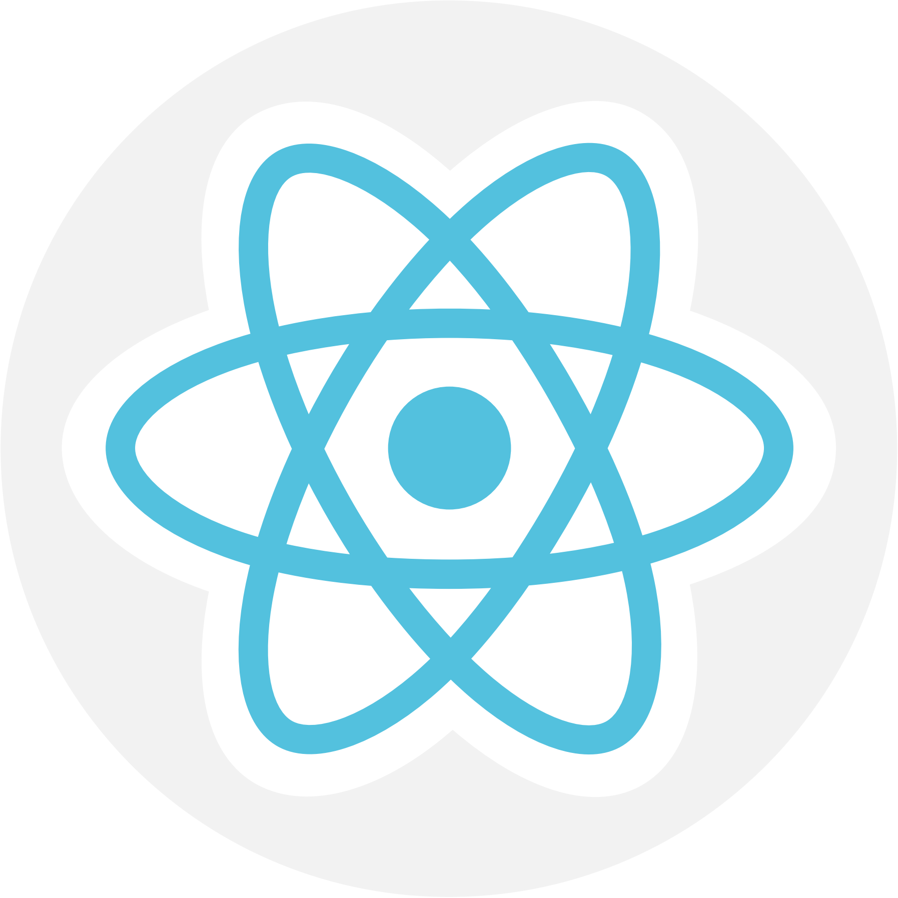
M.Sc. Shoeb Joarder

Social Computing Group, University of Duisburg-Essen

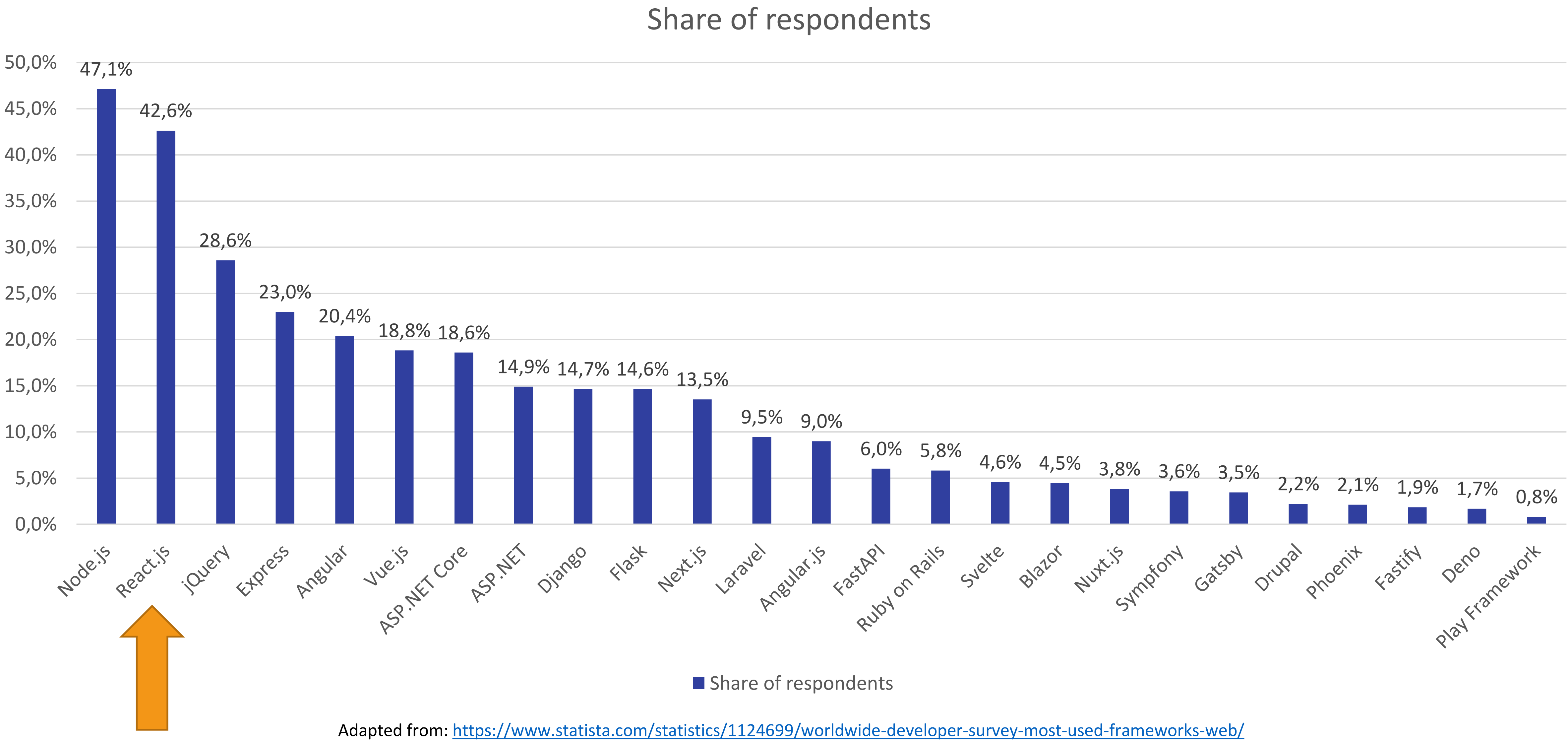
www.uni-due.de/soco

What is React? Why learn it?

- Front-end JS library for web development
  - Developed by Jordan Walke on Facebook’s newsfeed in 2011, later Instagram in 2012
  - Officially announced open-source in May 2013
  - The most popular frontend JS library in the industry (for now)
  - Huge community support, frequent updates, and developer tools available
- Simple and high performant tool
- Quick development of interactive UIs and responsive single-page applications
- Based on independent, isolated & reusable components



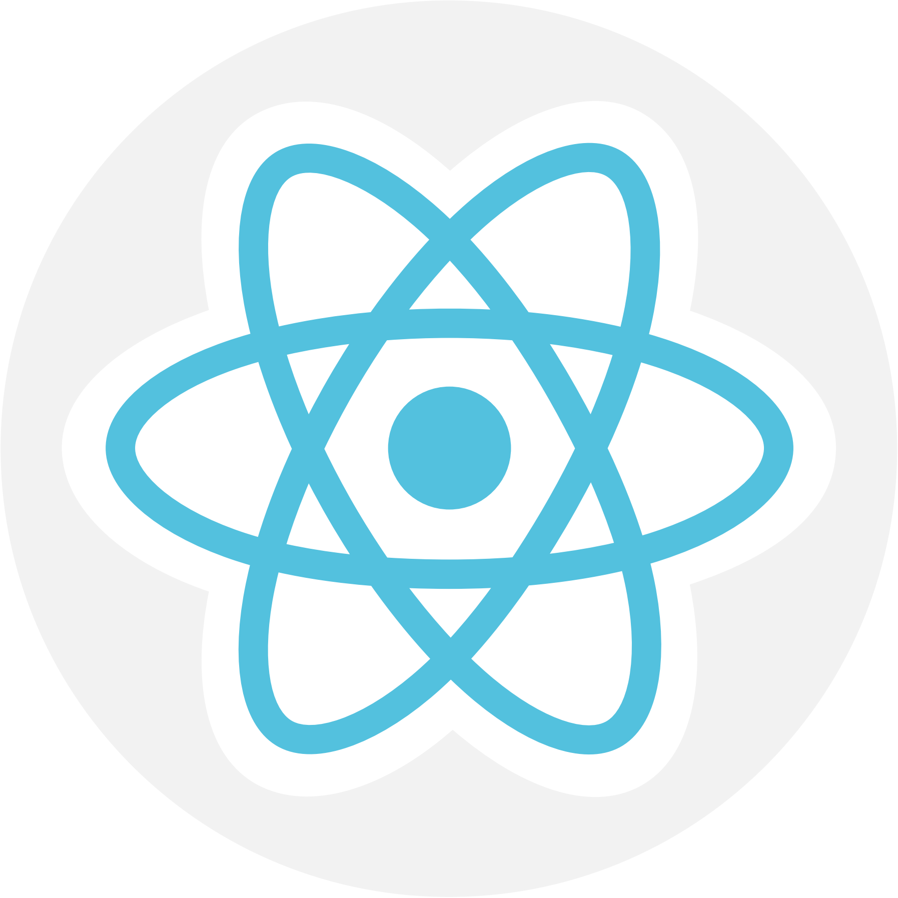
Most Used Web Frameworks Worldwide – 2022



Adapted from: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/

What should you know? Will learn?

- Fundamentals of JS, e.g., Objects, Arrays, Conditionals, etc.
- Knowledge of HTML and CSS
- Additional knowledge from the latest JS standard, e.g.,
  - Classes
  - Destructure objects
  - Array methods (map, forEach, spread operator […])
  - Arrow functions (syntax: ( ) => { })
  - Fetch API & promises



Class Topics

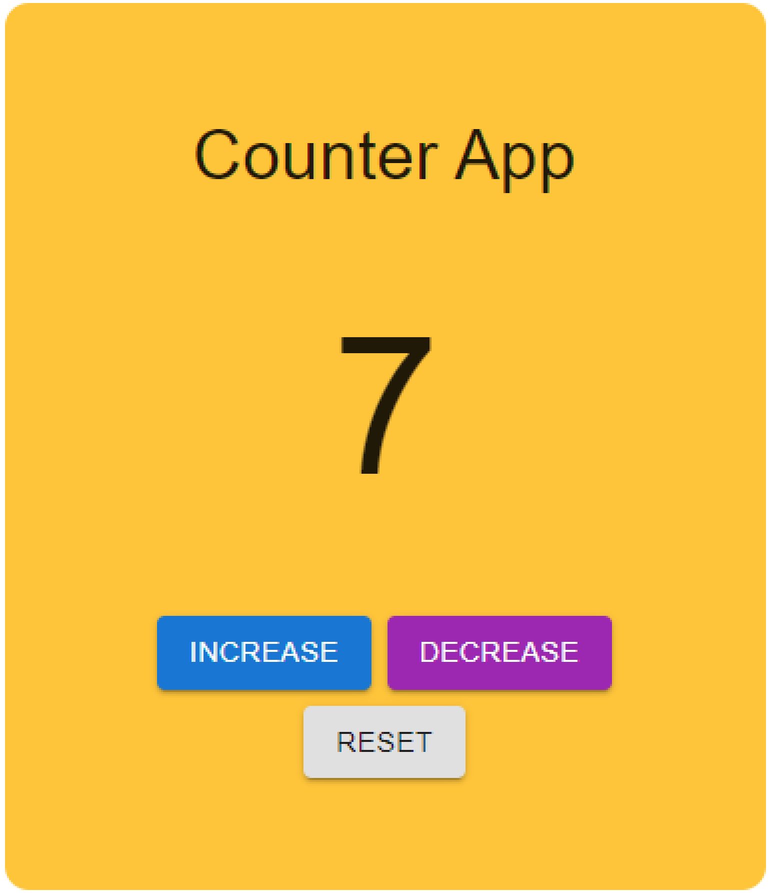
React Lecture 1

- Components
- States
- JSX, React Element, and Virtual DOM
- Event handlers
- Props
- Data binding

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

Simple Counter



Interactive buttons and display counts

[Codes available here]

Class Topics

React Lecture 1

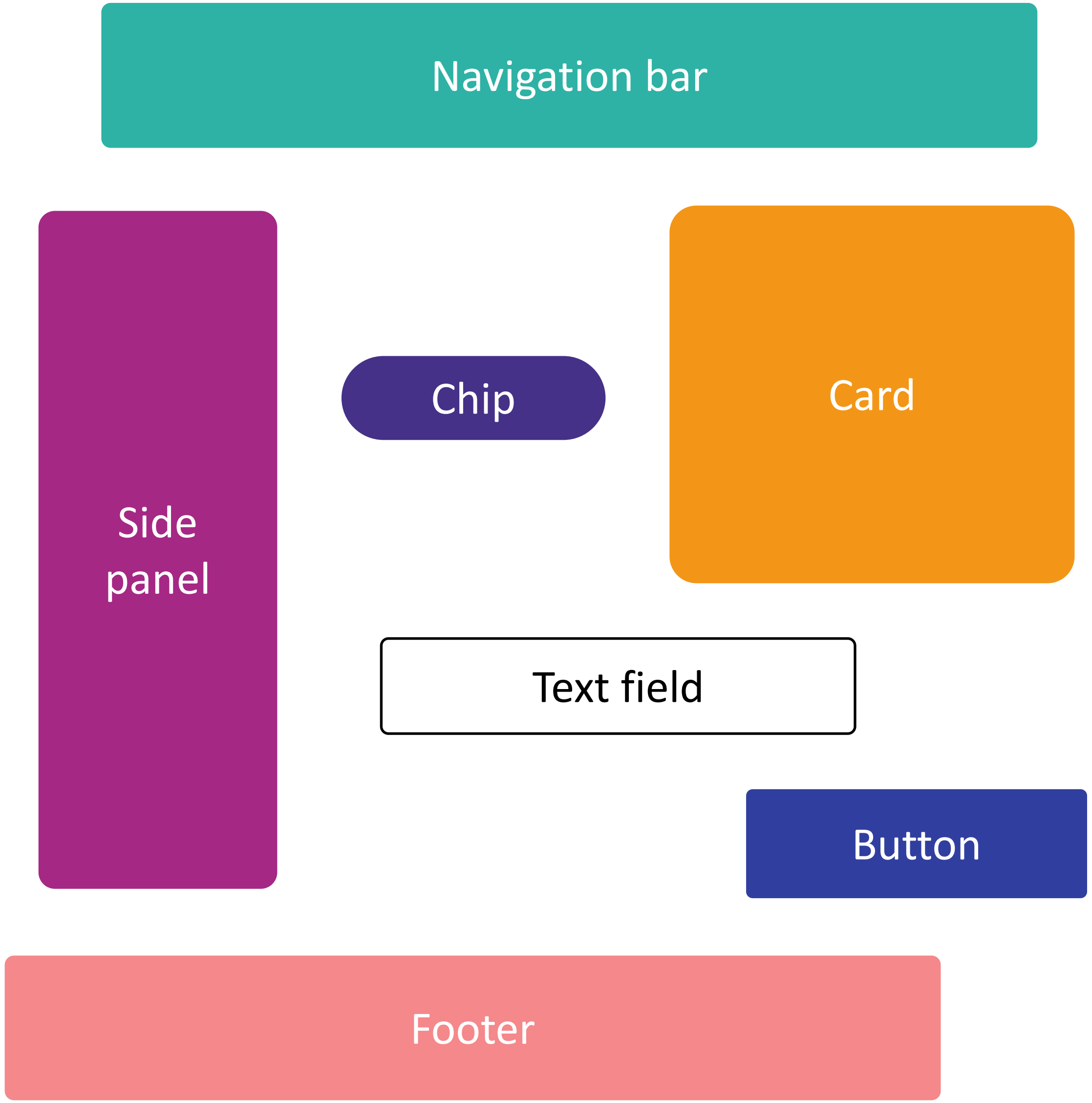
- **Components**
- States
- JSX, React Element, and Virtual DOM
- Event handlers
- Props
- Data binding

React Lecture 2

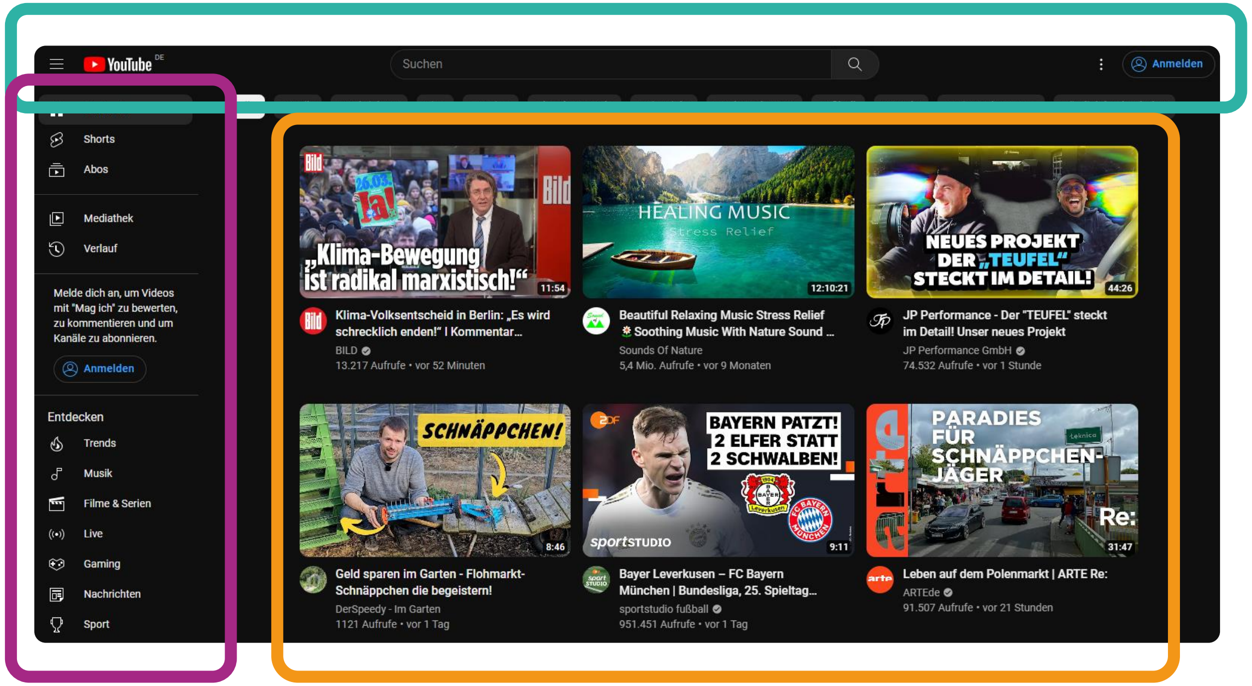
- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

Component

- Building block of a User Interface
- Independent, isolated, and reusable
- Multiple components work together to form a complex UI



Example: Component



Melde dich an, um Videos mit "Mag ich" zu bewerten, zu kommentieren und um Kanäle zu abonnieren.

Klima-Volksentscheid in Berlin: „Es wird schrecklich enden!" I Kommentar...
BILD
13.217 Aufrufe • vor 52 Minuten

Beautiful Relaxing Music Stress Relief 🌼 Soothing Music With Nature Sound ...
Sounds Of Nature
5,4 Mio. Aufrufe • vor 9 Monaten

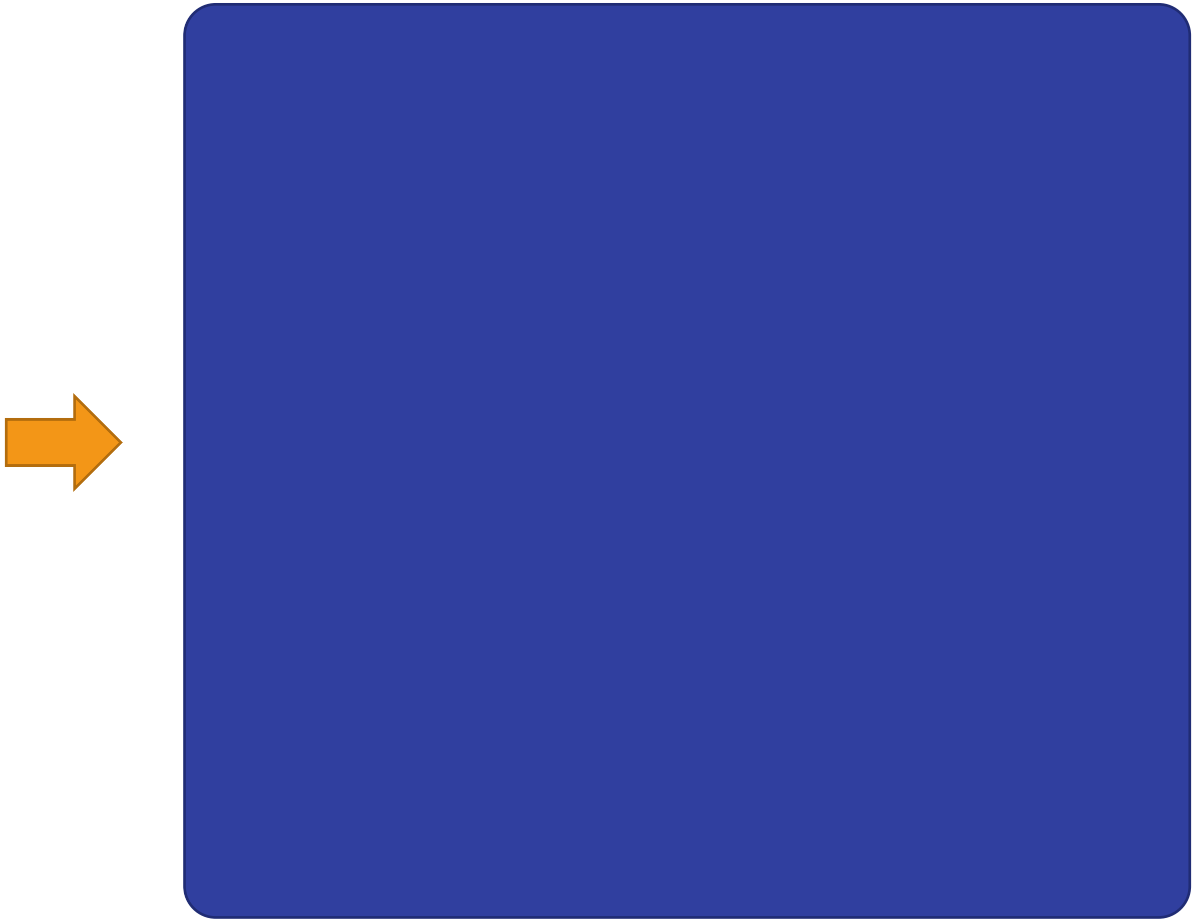
JP Performance - Der "TEUFEL" steckt im Detail! Unser neues Projekt
JP Performance GmbH
74.532 Aufrufe • vor 1 Stunde

Geld sparen im Garten – Flohmarkt-Schnäppchen die begeistern!
DerSpeedy - Im Garten
1121 Aufrufe • vor 1 Tag

Bayer Leverkusen – FC Bayern München | Bundesliga, 25. Spieltag...
sportstudio fußball
951.451 Aufrufe • vor 1 Tag

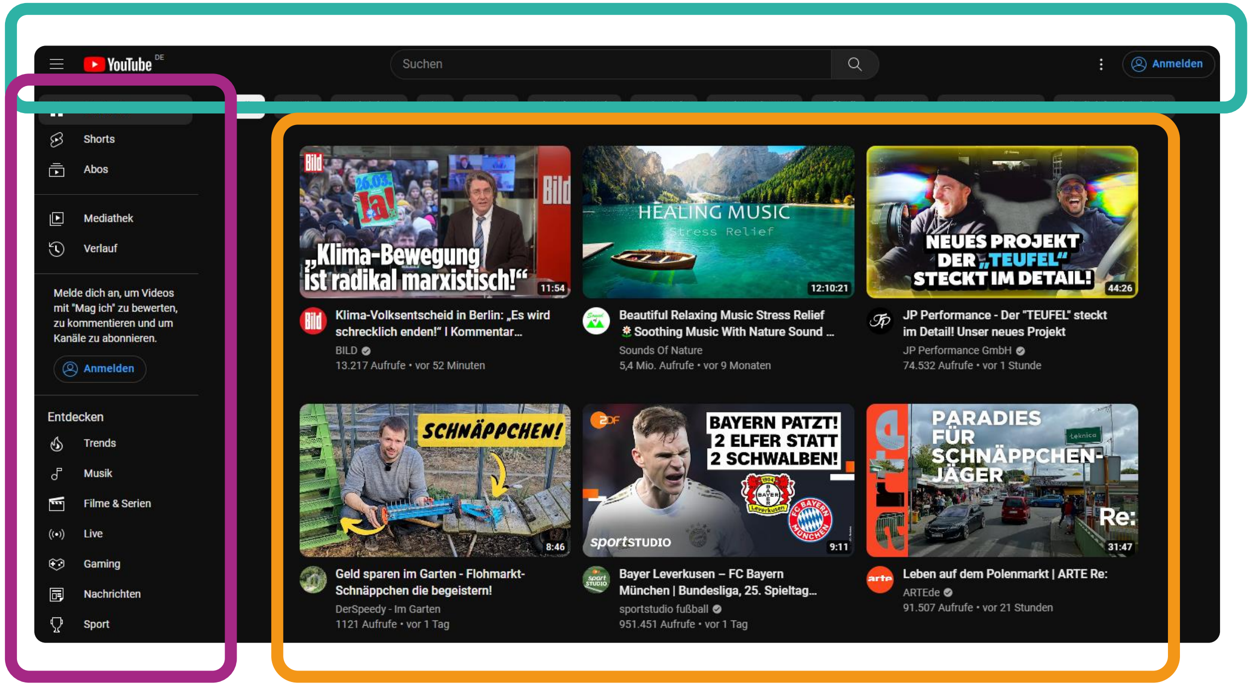
Leben auf dem Polenmarkt | ARTE Re:
ARTEde
91.507 Aufrufe • vor 21 Stunden

YouTube



One big application

Example: Component



Melde dich an, um Videos mit "Mag ich" zu bewerten, zu kommentieren und um Kanäle zu abonnieren.

Klima-Volksentscheid in Berlin: „Es wird schrecklich enden!" I Kommentar…
BILD
13.217 Aufrufe • vor 52 Minuten

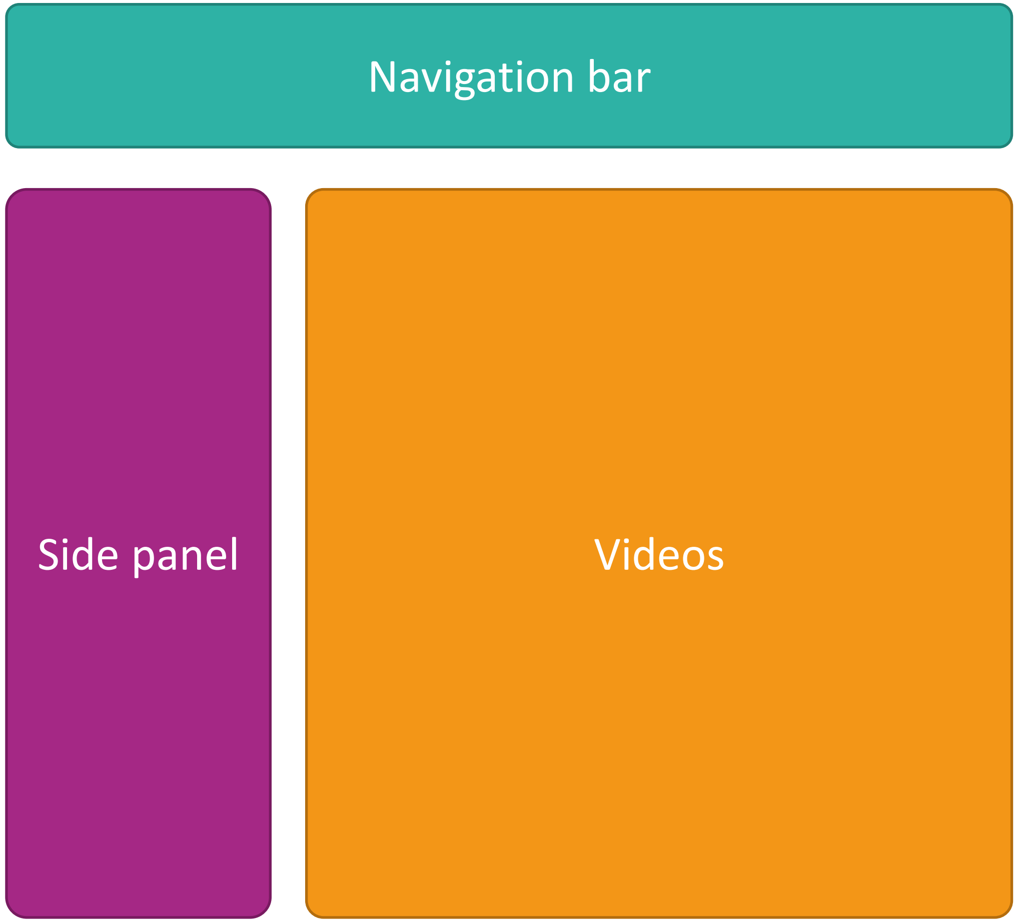
Beautiful Relaxing Music Stress Relief 🌼 Soothing Music With Nature Sound …
Sounds Of Nature
5,4 Mio. Aufrufe • vor 9 Monaten

JP Performance - Der "TEUFEL" steckt im Detail! Unser neues Projekt
JP Performance GmbH
74.532 Aufrufe • vor 1 Stunde

Geld sparen im Garten - Flohmarkt-Schnäppchen die begeistern!
DerSpeedy - Im Garten
1121 Aufrufe • vor 1 Tag

Bayer Leverkusen – FC Bayern München | Bundesliga, 25. Spieltag…
sportstudio fußball
951.451 Aufrufe • vor 1 Tag

Leben auf dem Polenmarkt | ARTE Re:
ARTEde
91.507 Aufrufe • vor 21 Stunden

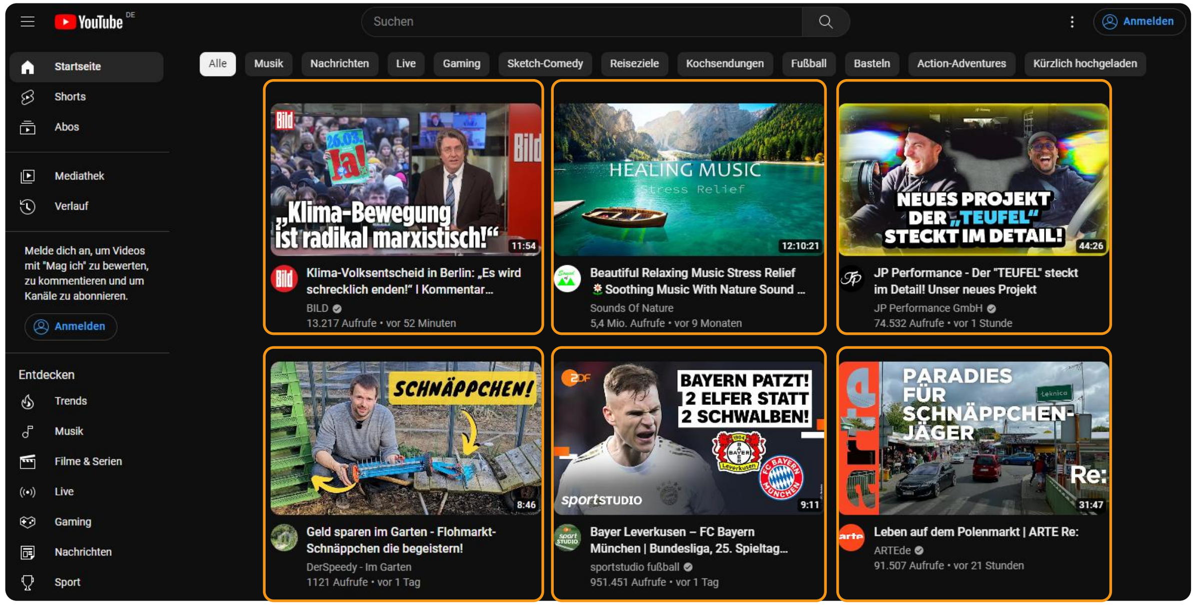


Navigation bar

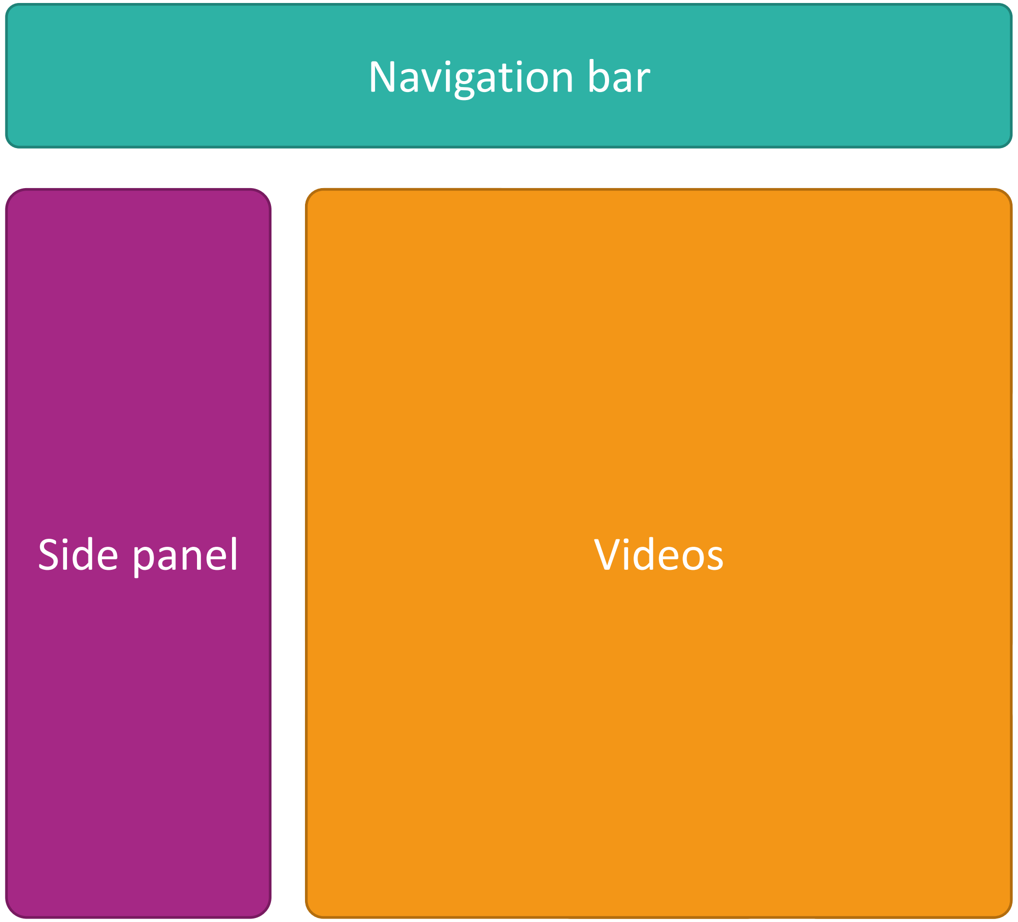
Side panel

Videos

Example: Component

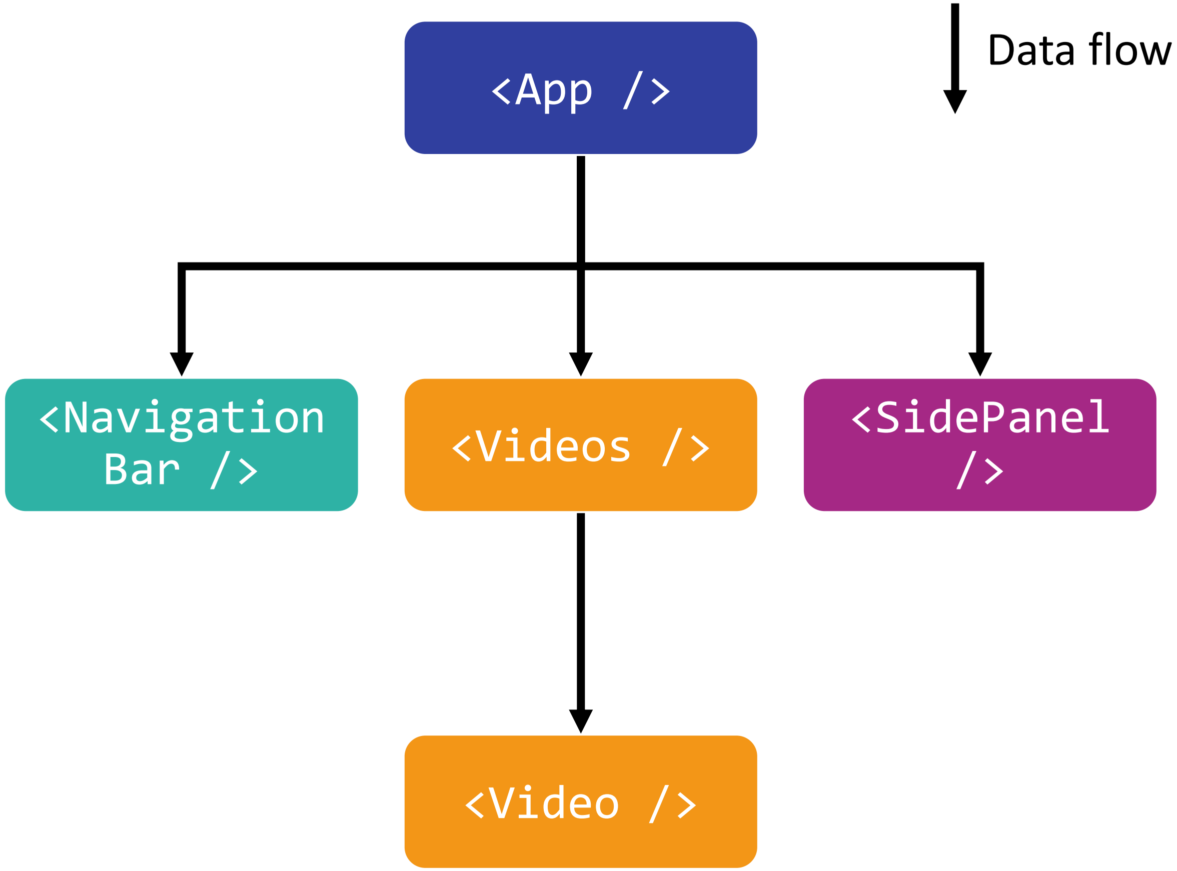

YouTube DE
Suchen
Anmelden
Startseite
Shorts
Abos
Mediathek
Verlauf
Melde dich an, um Videos mit "Mag ich" zu bewerten, zu kommentieren und um Kanäle zu abonnieren.
Anmelden
Entdecken
Trends
Musik
Filme & Serien
Live
Gaming
Nachrichten
Sport
Alle
Musik
Nachrichten
Live
Gaming
Sketch-Comedy
Reiseziele
Kochsendungen
Fußball
Basteln
Action-Adventures
Kürzlich hochgeladen
Klima-Volksentscheid in Berlin: „Es wird schrecklich enden!" I Kommentar...
BILD
13.217 Aufrufe • vor 52 Minuten
Beautiful Relaxing Music Stress Relief Soothing Music With Nature Sound ...
Sounds Of Nature
5,4 Mio. Aufrufe • vor 9 Monaten
JP Performance - Der "TEUFEL" steckt im Detail! Unser neues Projekt
JP Performance GmbH
74.532 Aufrufe • vor 1 Stunde
Geld sparen im Garten - Flohmarkt-Schnäppchen die begeistern!
DerSpeedy - Im Garten
1121 Aufrufe • vor 1 Tag
Bayer Leverkusen – FC Bayern München | Bundesliga, 25. Spieltag...
sportstudio fußball
951.451 Aufrufe • vor 1 Tag
Leben auf dem Polenmarkt | ARTE Re:
ARTEde
91.507 Aufrufe • vor 21 Stunden

YouTube



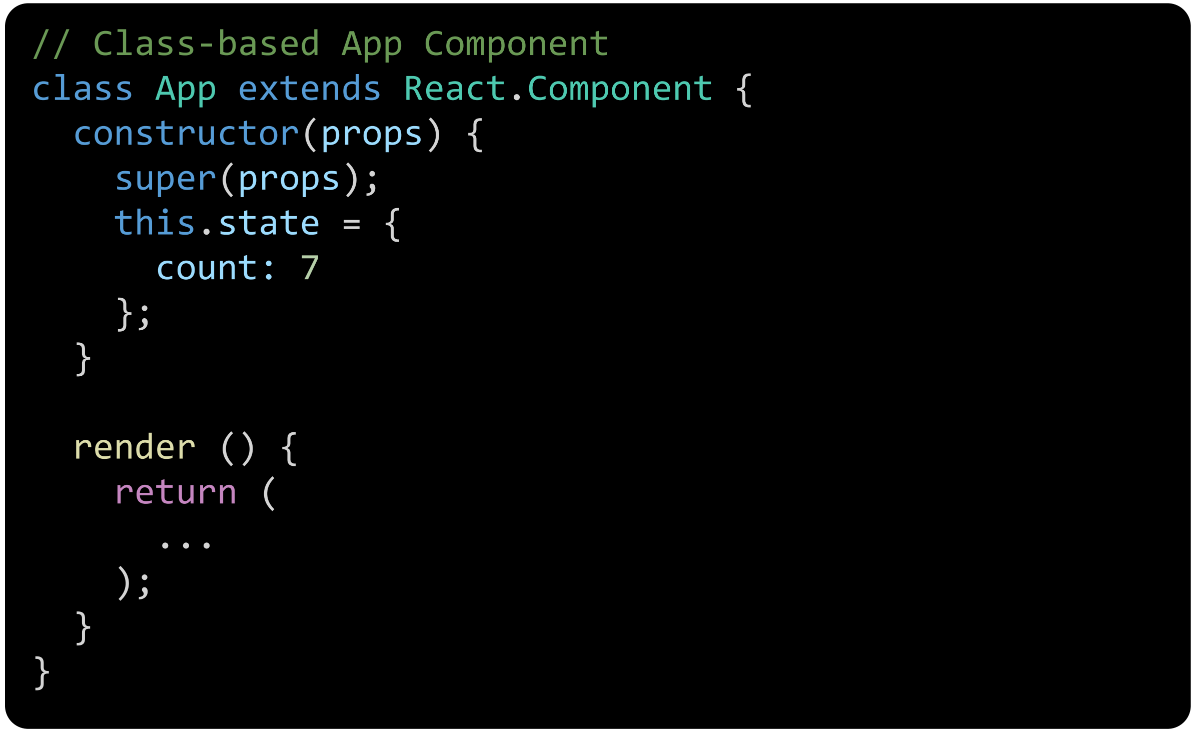
React Component

- Root component = App component
- Parent-child relationship
- Tree of components
- Top-down data flow



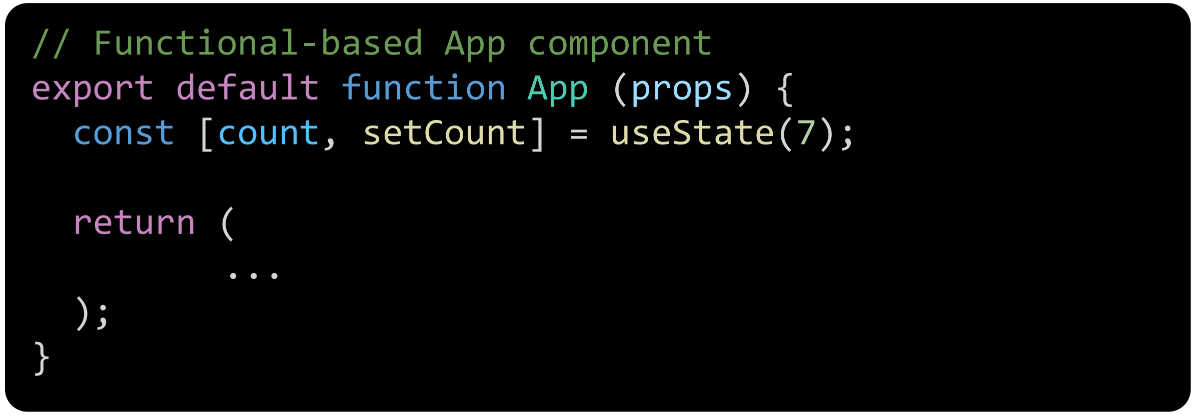
Component Types

- ~~Class-based component~~
  - ~~Constructor~~
  - ~~Super(props)~~
  - ~~State JS object~~
  - ~~Render method~~



```
// Class-based App Component
class App extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 7
    };
  }

  render () {
    return (
      ...
    );
  }
}
```

- Functional-based component
  - React Hooks
  - Return statement



```
// Functional-based App component
export default function App (props) {
  const [count, setCount] = useState(7);

  return (
         ...
  );
}
```

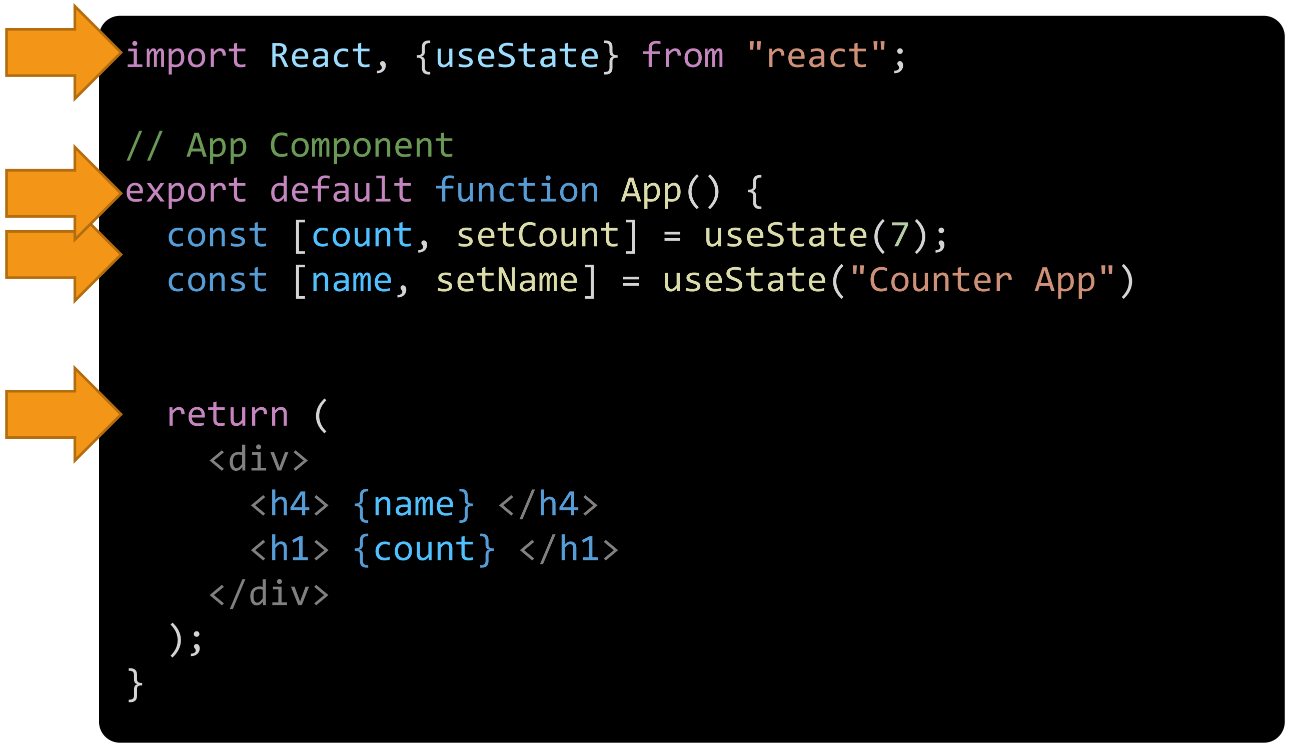
Component – Structure & Features

- Import statements and declare component name

- State
  - Contains data or information about a component
  - Re-renders component when it changes

- Return statement
  - Returns data from component
  - Describes how the UI should look

Typical structure of React components



```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")


  return (
    <div>
      <h4> {name} </h4>
      <h1> {count} </h1>
    </div>
  );
}
```

Demo

Class Topics

React Lecture 1

✓ Components

- **States**
- JSX, React Element, and Virtual DOM
- Event handlers
- Props
- Data binding

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

React States

- Defined using useState hook
  - One input parameter
    - Initial value of the state
  - Returns an array with two values
    - Current value of the state
    - Function to update the state

- Various types
  - Numbers/strings, e.g., 9, "abc"
  - JS objects e.g., {name: "John"}
  - Arrays, e.g., [9, 2, 10]
  - HTML + CSS codes
  - and more…

- Re-renders component when state changes

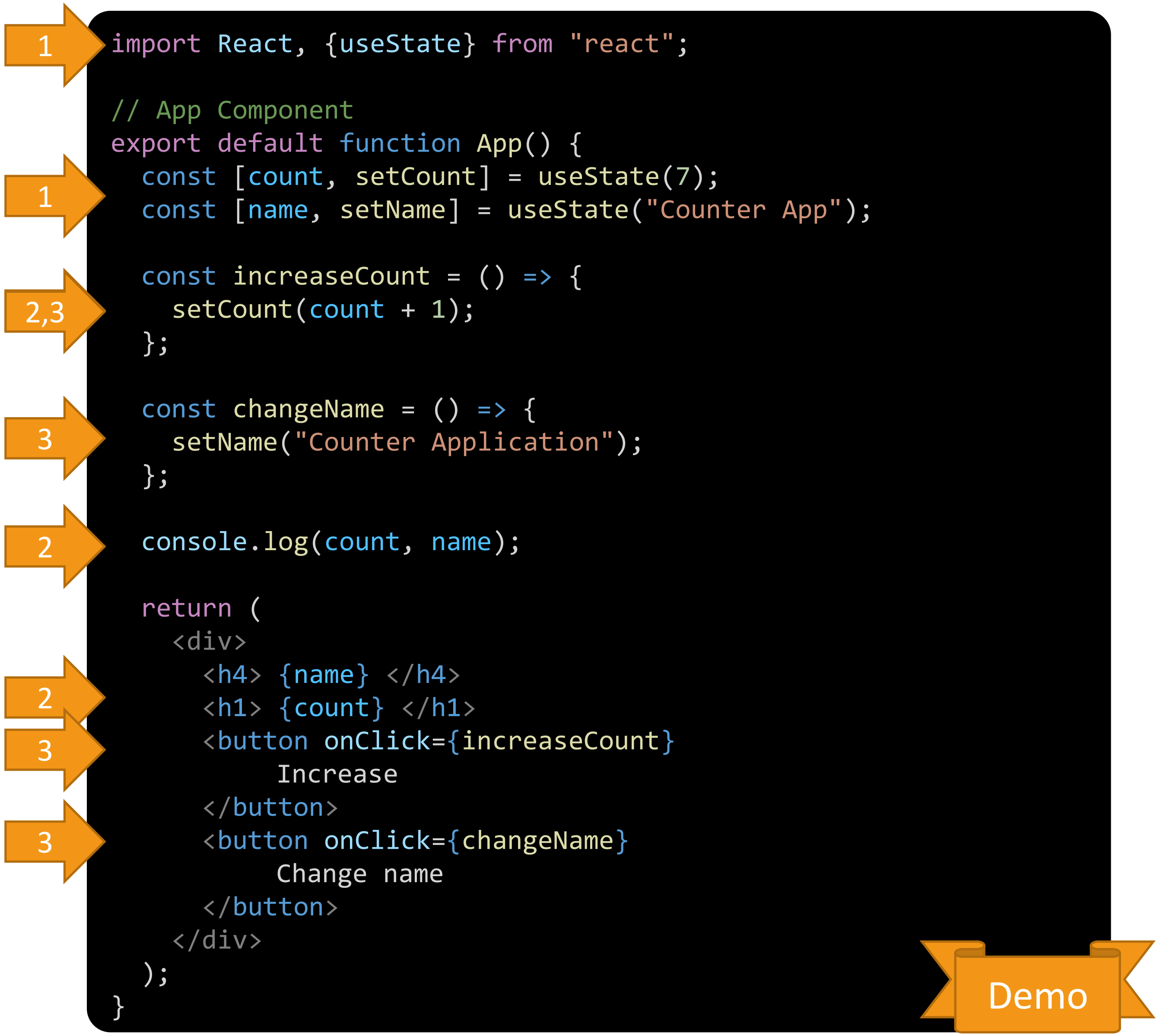
```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App");

  const increaseCount = () => {
    setCount(count + 1);
  };

  const changeName = () => {
    setName("Counter Application");
  };

  console.log(count, name);

  return (
    <div>
      <h4> {name} </h4>
      <h1> {count} </h1>
      <button onClick={increaseCount}
           Increase
      </button>
      <button onClick={changeName}
           Change name
      </button>
    </div>
  );
}
```

State Conventions

1. Initialize states using useState hook

2. Access state
   - Return statement
   - Inside a function
   - Browsers console

3. Update state
   - Takes new value
   - Re-renders component



```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App");

  const increaseCount = () => {
    setCount(count + 1);
  };

  const changeName = () => {
    setName("Counter Application");
  };

  console.log(count, name);

  return (
    <div>
      <h4> {name} </h4>
      <h1> {count} </h1>
      <button onClick={increaseCount}
          Increase
      </button>
      <button onClick={changeName}
          Change name
      </button>
    </div>
  );
}
```

Demo

Class Topics

React Lecture 1

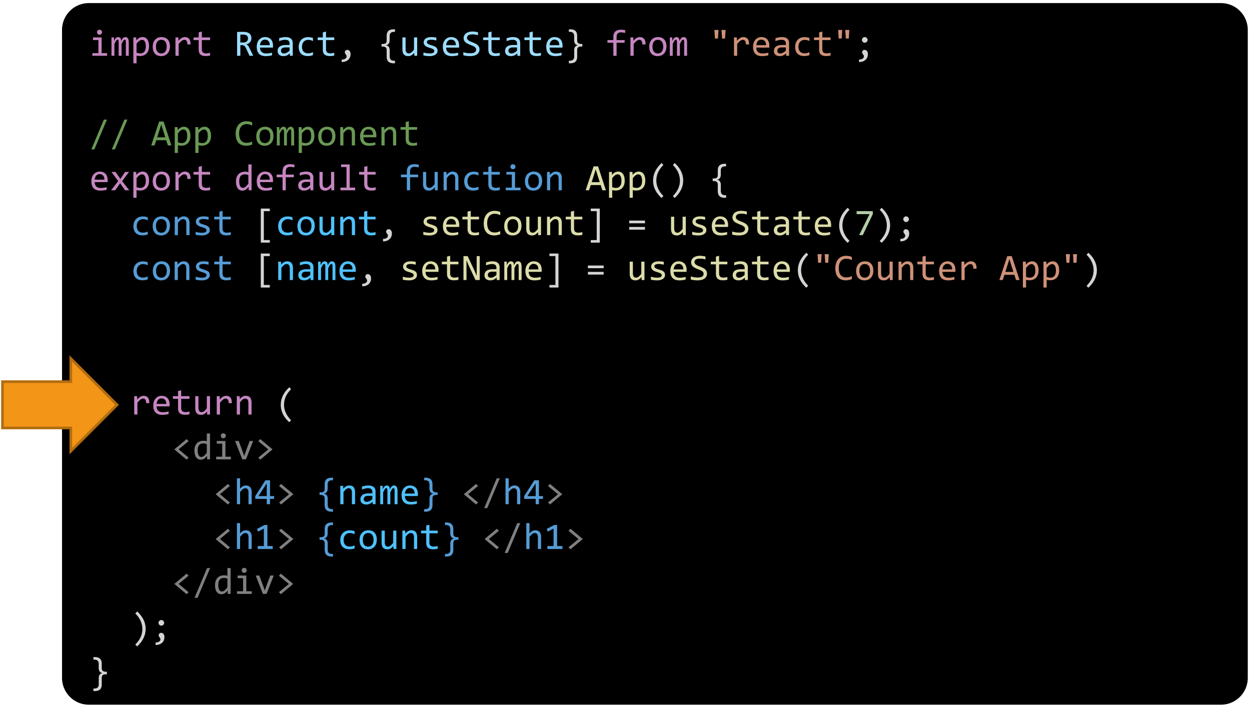
- ✓Components
- ✓States
- **JSX, React Element, and Virtual DOM**
- Event handlers
- Props
- Data binding

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

JavaScript XML (JSX)

- JavaScript code + HTML/XML
- Transpiler required, e.g., Babel.js, to convert JSX to JS
- Benefits
  - Optimized code translation and faster than regular JS
  - Brings different technologies together
    - JavaScript, HTML, CSS
    - Easy to create templates
  - Type safe
- Transpiler creates React Element object



```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")


  return (
    <div>
      <h4> {name} </h4>
      <h1> {count} </h1>
    </div>
  );
}
```

React Element

- Building block of UI and specifies how UI looks like
- Method `createElement` structure
  - `type` of HTML element, e.g., div, h1 tags, etc.
  - `props`, e.g., style, Eventhandlers, etc.
  - `children` are things to be displayed, e.g., states
- Render the App component using the `render` method
  - Element that needs to be rendered
  - The place to render in DOM
- React interacts with the Virtual DOM

Method to create a React Element

```
React.createElement(type, props, ...children);
```

```
import React, { useState } from "react";
import { createRoot } from "react-dom/client";

// App component
const App = () => {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App");

  return React.createElement(
    "div",
    {},
    React.createElement("h4", null, name),
    React.createElement("h1", null, count)
  );
};

const rootElement = document.getElementById("root");
const root = createRoot(rootElement);

root.render(<App />);
```

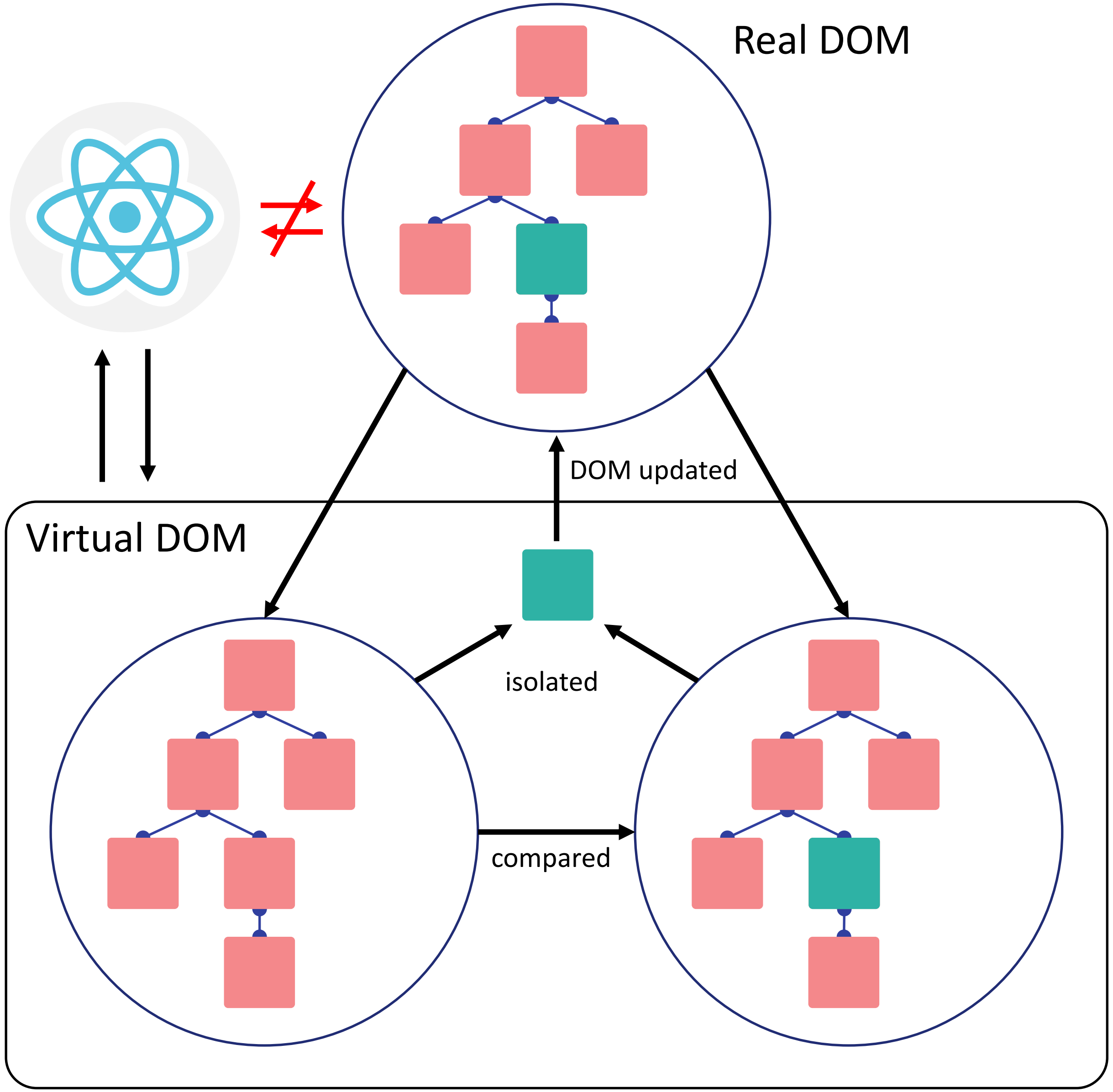
Demo

Virtual DOM

- Lightweight React Element (JS) object
- Copy of real DOM
- Benefits
  - Re-rendering real DOM is costly
  - In-memory (fast!)

- Works in three steps
  - Re-renders UI when data changes in the Virtual DOM
  - Calculates the difference between copies of virtual DOM
  - Updates real DOM

- React doesn’t read from real DOM
- React interacts with the Virtual DOM



Class Topics

React Lecture 1

✓Components

✓States

✓JSX, React Element, and Virtual DOM

- **Event handlers**
- Props
- Data binding

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

Event Handlers

- Events?
  - Actions triggered by users
  - E.g., pressing a key, mouse click, etc.
- Event handlers
  - Determines what kind of action to take
  - E.g., onClick, onChange, etc.
- Naming convention in camelCase
- Pass a function as the event handler

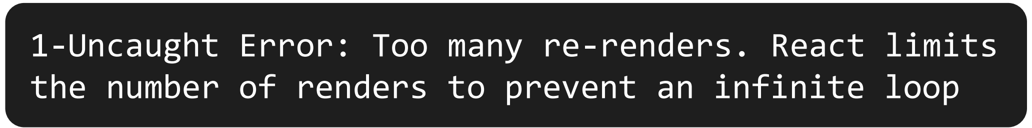
```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);

  const increaseCount = () =>
    setCount(count + 1);
  };

  const decreaseCount = () => {
    setCount(count - 1);
  };

  const resetToDefault = () => {
    setCount(7);
  };

  return (
    <div>
      <h4> {name} </h4>
      <h1> {count} </h1>
      <button onClick={increaseCount}> Increase </button>
      <button onClick={decreaseCount}> Decrease </button>
      <button onClick={resetToDefault}> Reset </button>
    </div>
  );
}
```

Demo

Event Handlers – Passing Arguments

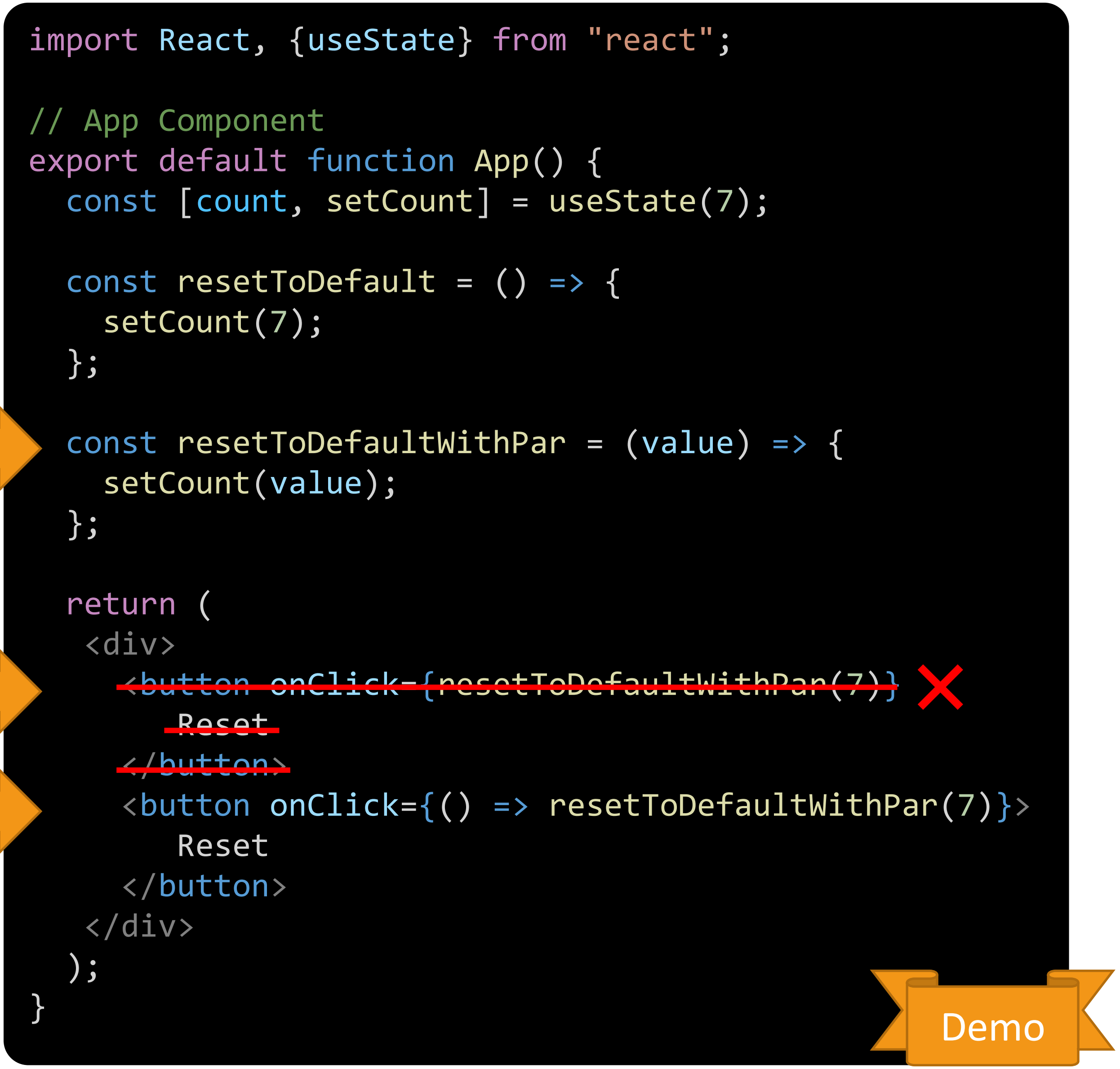
- Function accepts one input parameter value (in our case!)

- Functions with parameters called directly inside the event handler?
  - Function called when a component is loaded, causing an endless loop:



```
1-Uncaught Error: Too many re-renders. React limits
the number of renders to prevent an infinite loop
```

- Triggered only when clicked using an arrow function

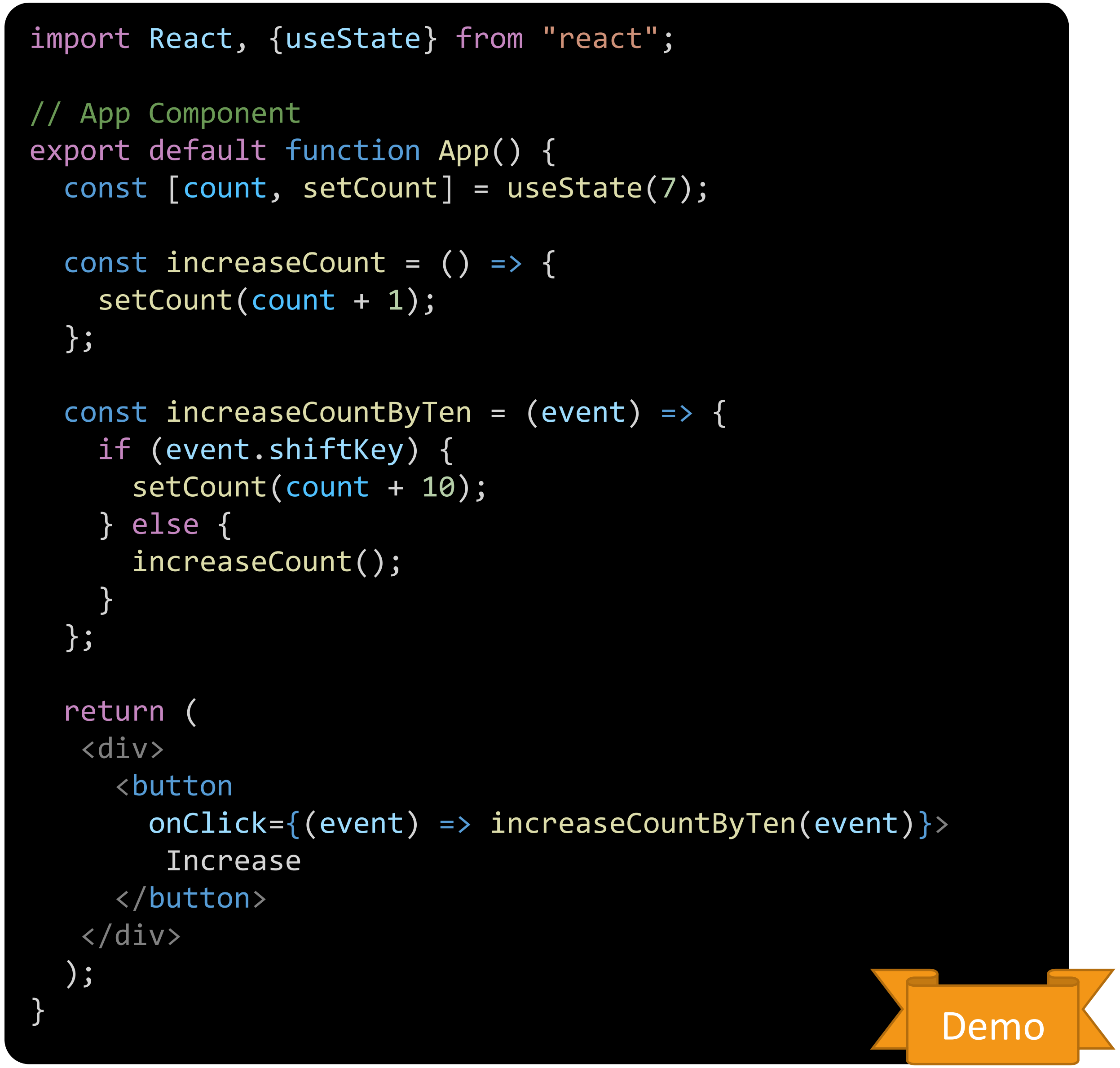


```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);

  const resetToDefault = () => {
    setCount(7);
  };

  const resetToDefaultWithPar = (value) => {
    setCount(value);
  };

  return (
   <div>
     ~~<button onClick={resetToDefaultWithPar(7)}~~ ✗
       ~~Reset~~
     ~~</button>~~
     <button onClick={() => resetToDefaultWithPar(7)}>
       Reset
     </button>
   </div>
  );
}
```

Demo

Event Handlers – Event Param/Properties/Object

- Provides extra details specific to a type of event
- Function with an event parameter
- Access to events such as `shiftKey`, `ctrlKey`, `altKey`



```
import React, {useState} from "react";

// App Component
export default function App() {
  const [count, setCount] = useState(7);

  const increaseCount = () => {
    setCount(count + 1);
  };

  const increaseCountByTen = (event) => {
    if (event.shiftKey) {
      setCount(count + 10);
    } else {
      increaseCount();
    }
  };

  return (
   <div>
     <button
       onClick={(event) => increaseCountByTen(event)}>
        Increase
     </button>
   </div>
  );
}
```

Demo

Class Topics

React Lecture 1

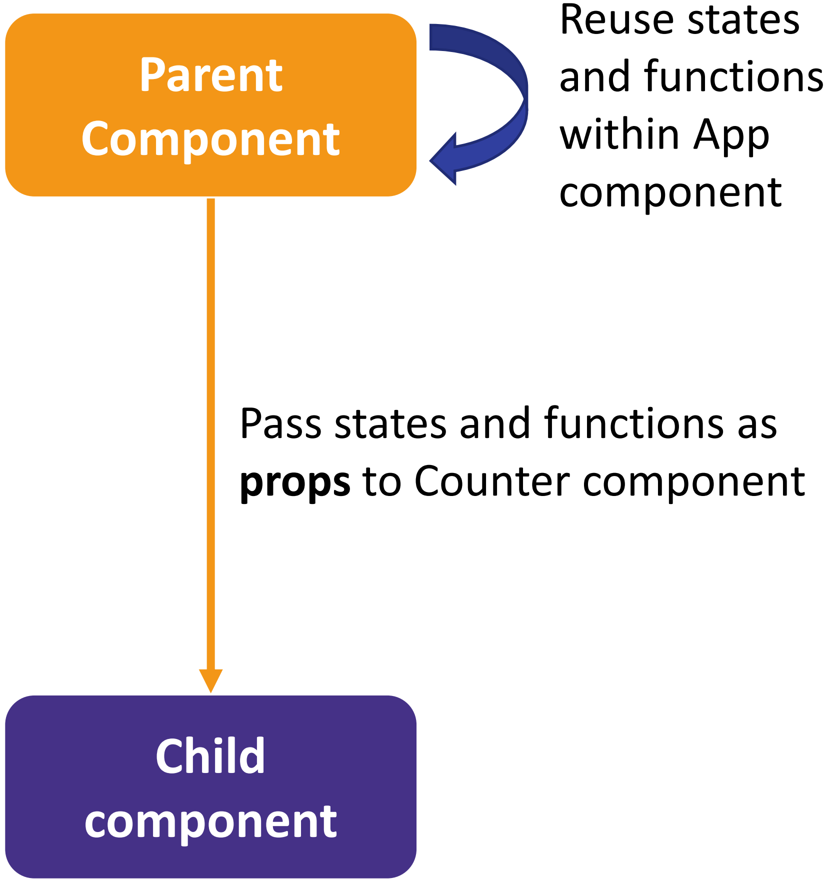
- ✓Components
- ✓States
- ✓JSX, React Element, and Virtual DOM
- ✓Event handlers
- **Props**
- Data binding

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

Props

- Props, a.k.a. properties
- Inputs to components
  - Single values
  - Objects with a set of values
  - States
  - Functions
- Pass from parent component to child component(s)
- Can trigger a change in state
- Immutable



Parent Component
Reuse states and functions within App component
Pass states and functions as **props** to Counter component
Child component

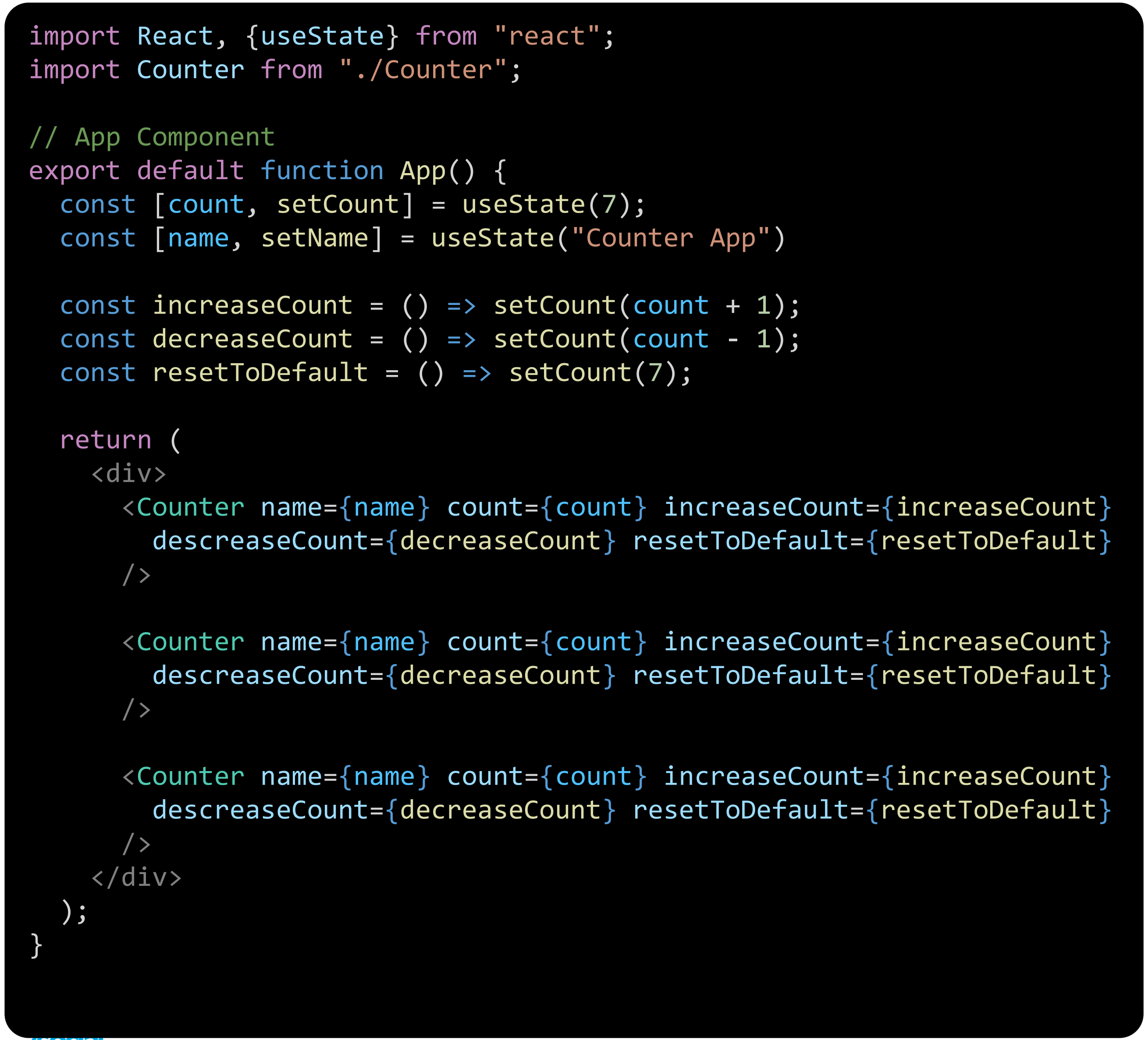
Props – Counter Component

```
import React, {useState} from "react";
import Counter from "./Counter";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")

  const increaseCount = () => setCount(count + 1);
  const decreaseCount = () => setCount(count - 1);
  const resetToDefault = () => setCount(7);

  return (
    <div>
      <h4> {name} </h4>
      <h1> {count} </h1>
      <button onClick={increaseCount}> Increase </button>
      <button onClick={decreaseCount}> Decrease </button>
      <button onClick={resetToDefault}> Reset </button>

      // Pass props to child component “Counter”
      <Counter
        name={name}
        count={count}
        increaseCount={increaseCount}
        descreaseCount={decreaseCount}
        resetToDefault={resetToDefault}
      />
    </div>
  );
}
```

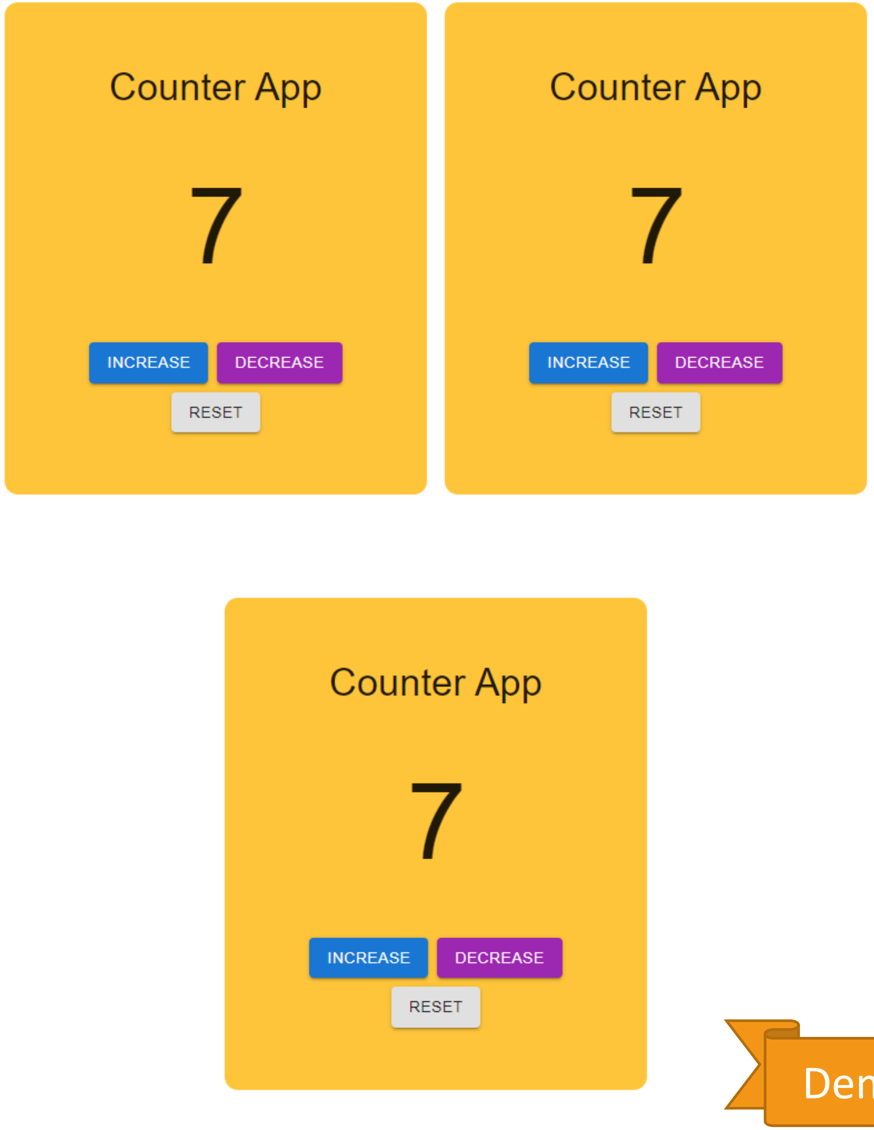
```
import React from "react";

// Counter Component
export default function Counter({
  // Read props inside the child component “Counter”
  name,
  count,
  increaseCount,
  descreaseCount,
  resetToDefault,
}) => {

 // Child component “Counter” has no states

  return (
    <>
      <h4> {name} </h4>
      <h1> {count} </h1>
      <button onClick={increaseCount}> Increase </button>
      <button onClick={decreaseCount}> Decrease </button>
      <button onClick={resetToDefault}> Reset </button>
    </>
  );
}
```

Demo

Props – Modular Component Counter

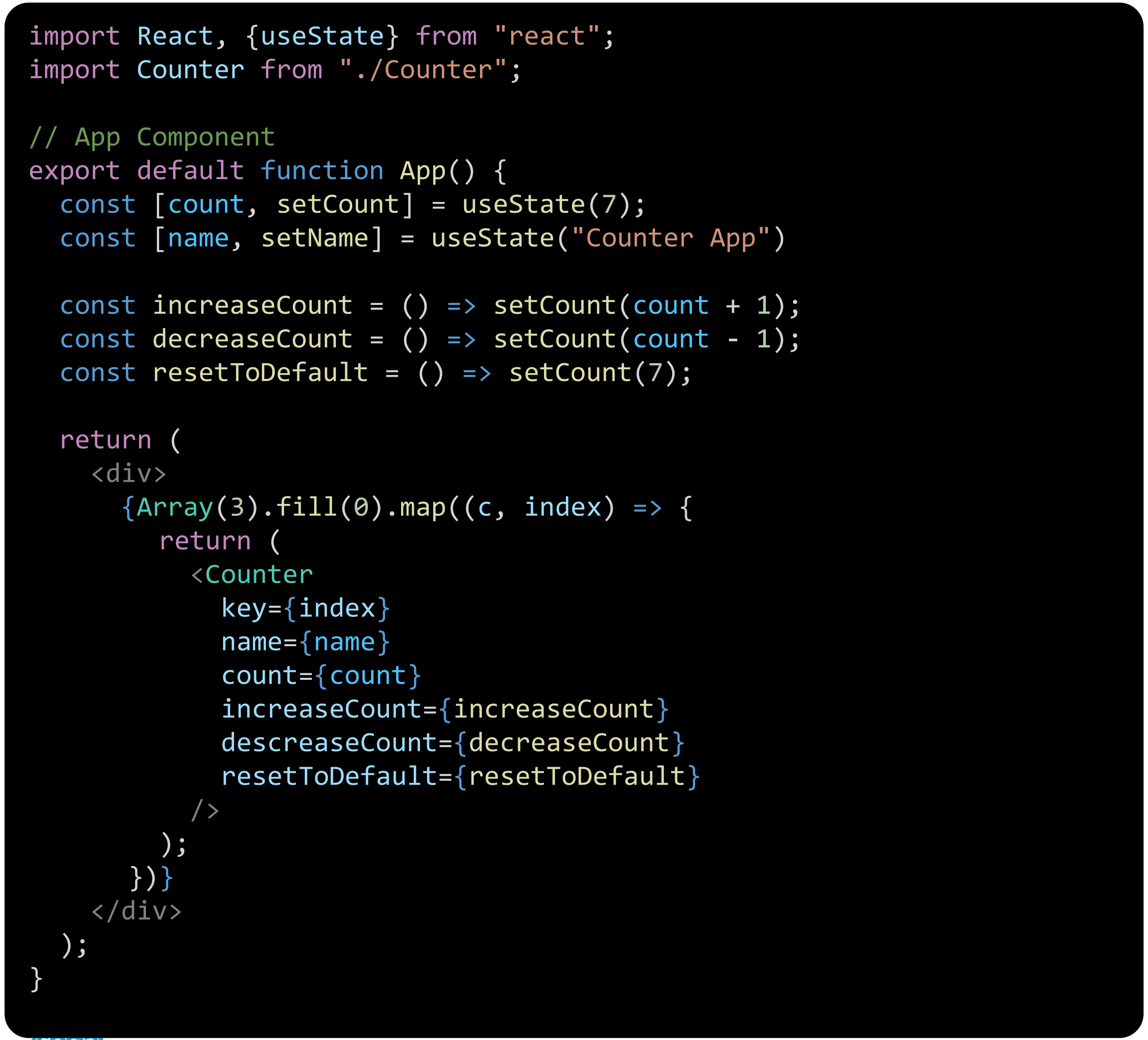


```
import React, {useState} from "react";
import Counter from "./Counter";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")

  const increaseCount = () => setCount(count + 1);
  const decreaseCount = () => setCount(count - 1);
  const resetToDefault = () => setCount(7);

  return (
    <div>
      <Counter name={name} count={count} increaseCount={increaseCount}
        descreaseCount={decreaseCount} resetToDefault={resetToDefault}
      />

      <Counter name={name} count={count} increaseCount={increaseCount}
        descreaseCount={decreaseCount} resetToDefault={resetToDefault}
      />

      <Counter name={name} count={count} increaseCount={increaseCount}
        descreaseCount={decreaseCount} resetToDefault={resetToDefault}
      />
    </div>
  );
}
```

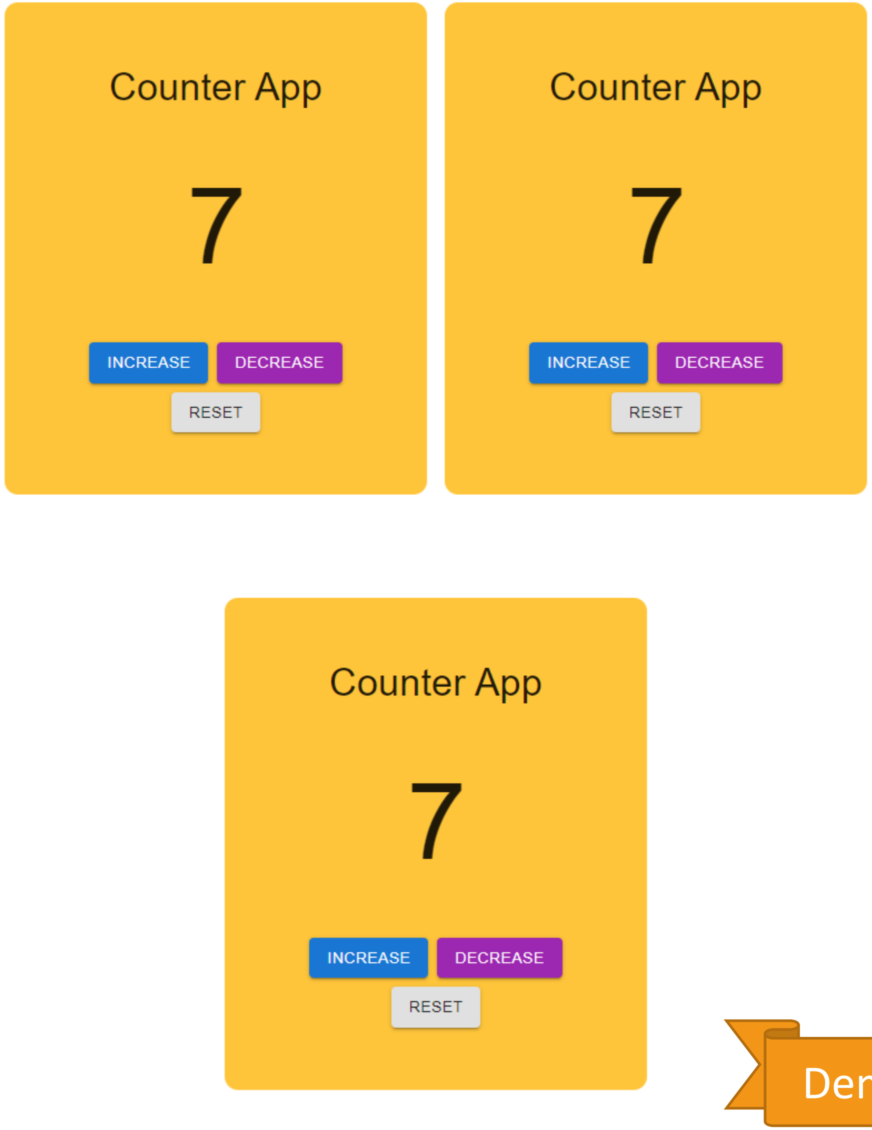


Demo

Map method



```
import React, {useState} from "react";
import Counter from "./Counter";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")

  const increaseCount = () => setCount(count + 1);
  const decreaseCount = () => setCount(count - 1);
  const resetToDefault = () => setCount(7);

  return (
    <div>
      {Array(3).fill(0).map((c, index) => {
        return (
          <Counter
            key={index}
            name={name}
            count={count}
            increaseCount={increaseCount}
            descreaseCount={decreaseCount}
            resetToDefault={resetToDefault}
          />
        );
      })}
    </div>
  );
}
```



Demo

Class Topics

React Lecture 1

- ✓Components
- ✓States
- ✓JSX, React Element, and Virtual DOM
- ✓Event handlers
- ✓Props
- **Data binding**

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

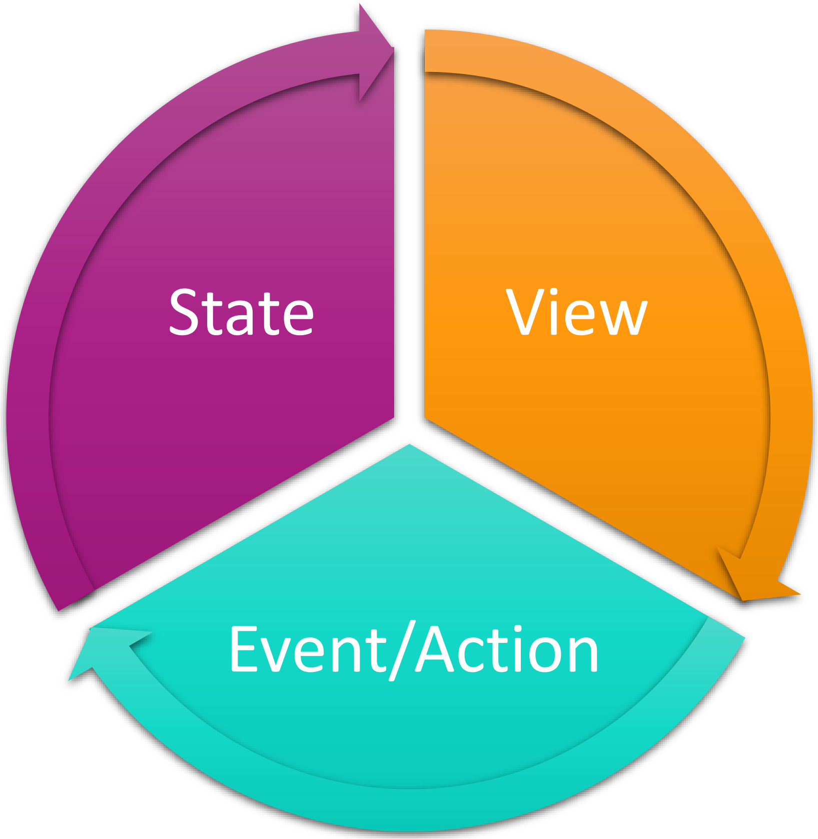
Data Binding

- Connection between the Model and the View

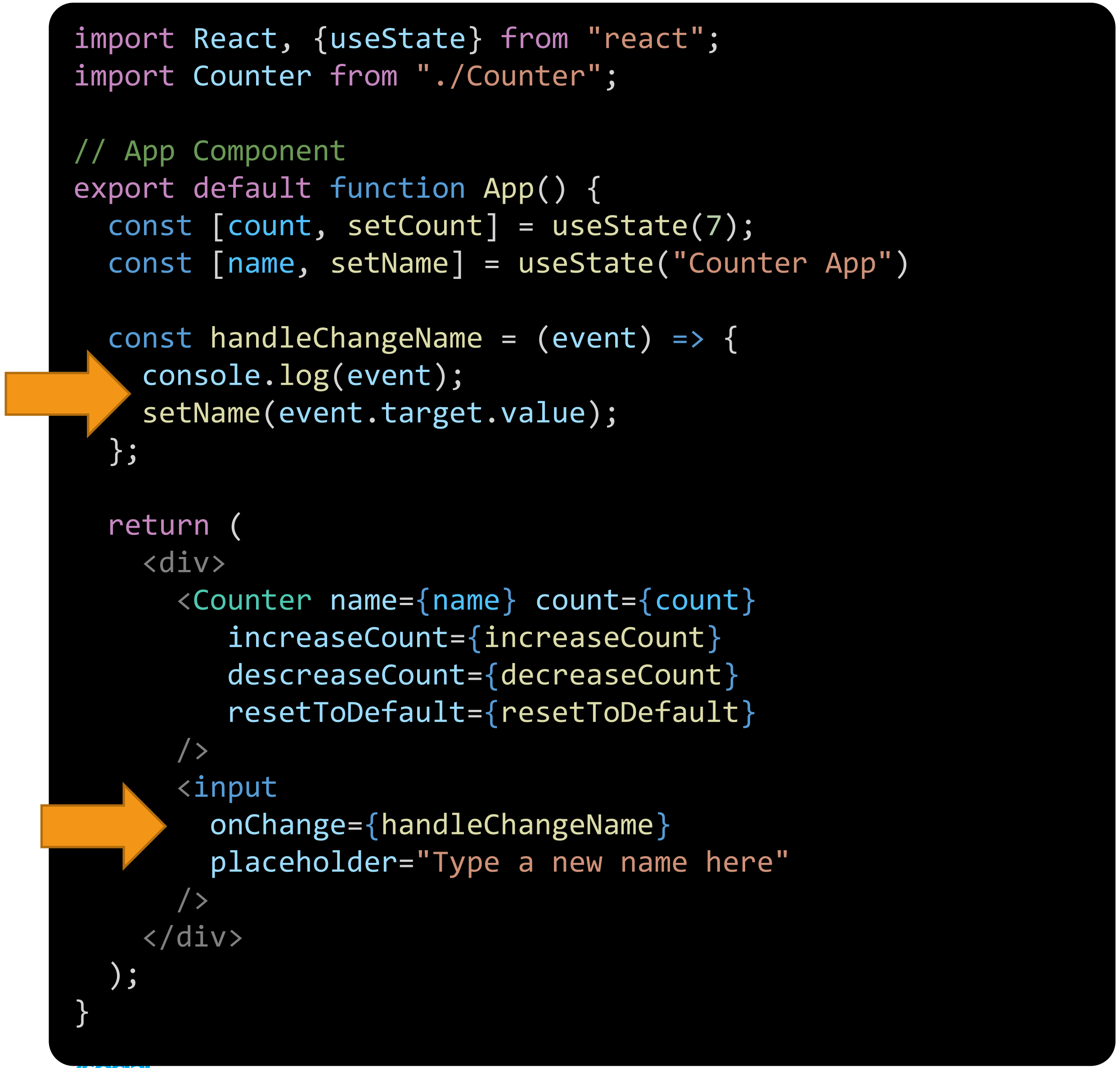
- Change in the model leads to a change in view and vice versa

- One-/Two-way data binding

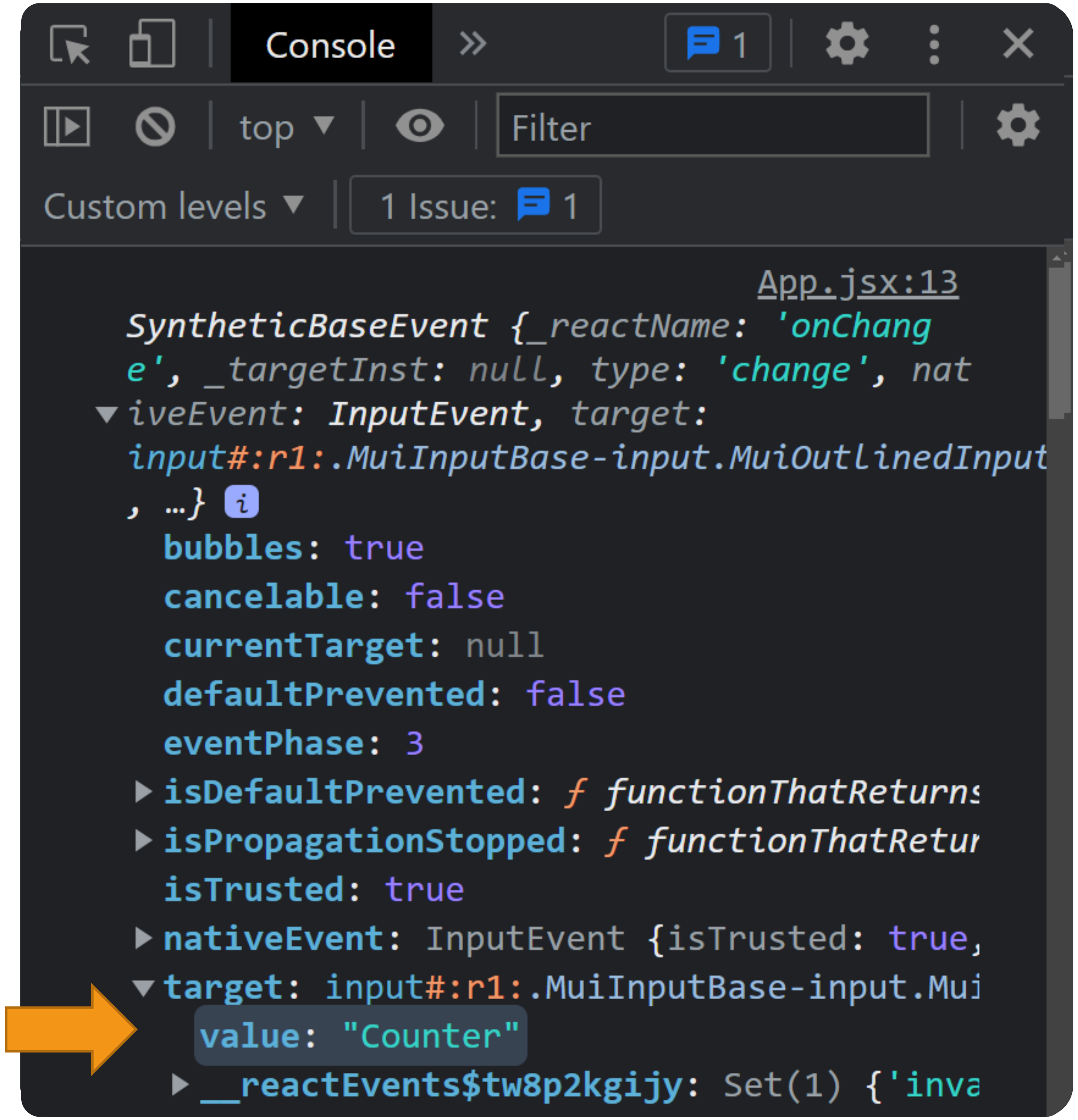
- Data binding in React
  - One-way data binding
  - View cannot change State
  - Only through an event/action



Data Binding

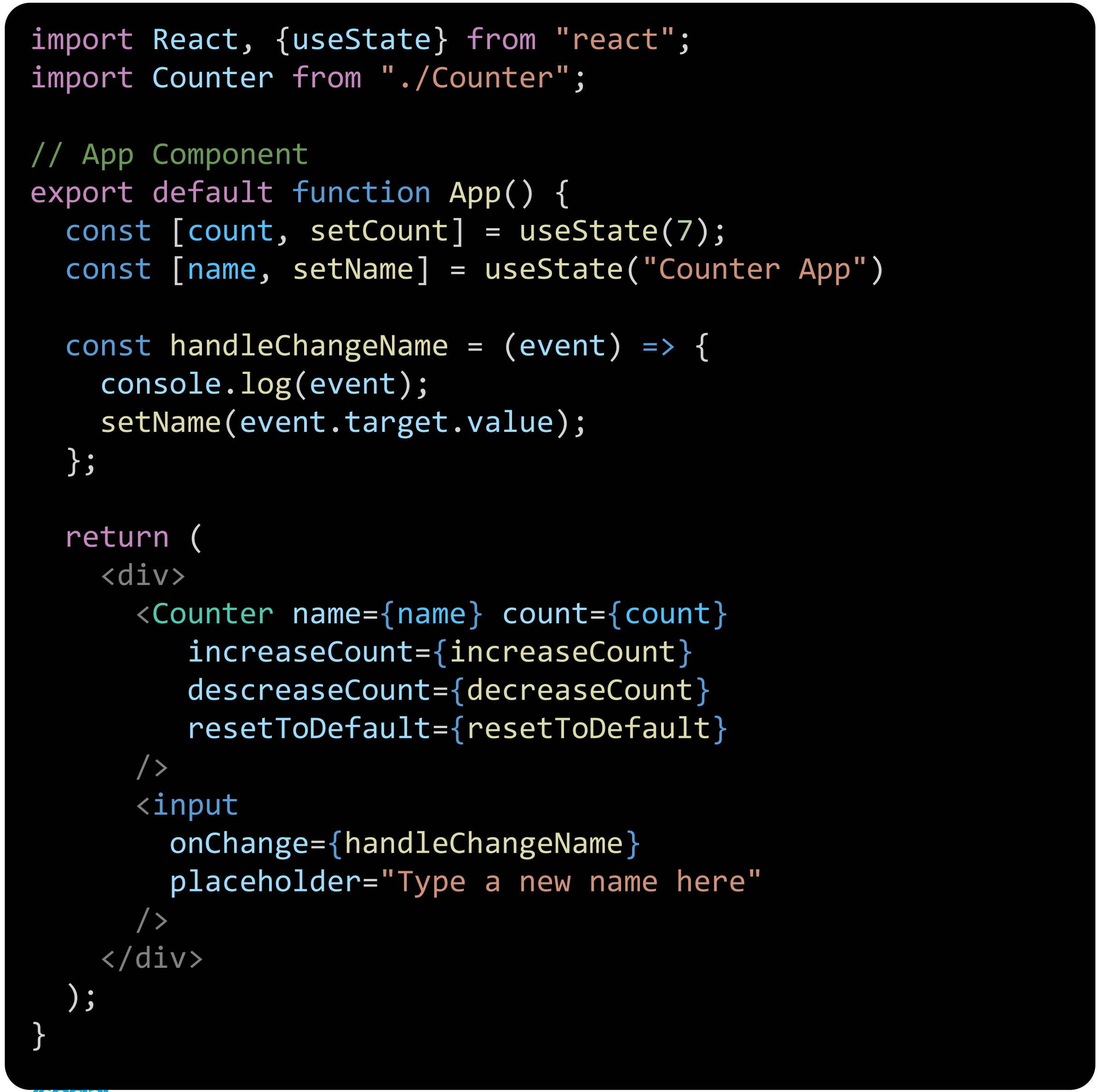


```
import React, {useState} from "react";
import Counter from "./Counter";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")

  const handleChangeName = (event) => {
    console.log(event);
    setName(event.target.value);
  };

  return (
    <div>
      <Counter name={name} count={count}
         increaseCount={increaseCount}
         descreaseCount={decreaseCount}
         resetToDefault={resetToDefault}
      />
      <input
        onChange={handleChangeName}
        placeholder="Type a new name here"
      />
    </div>
  );
}
```

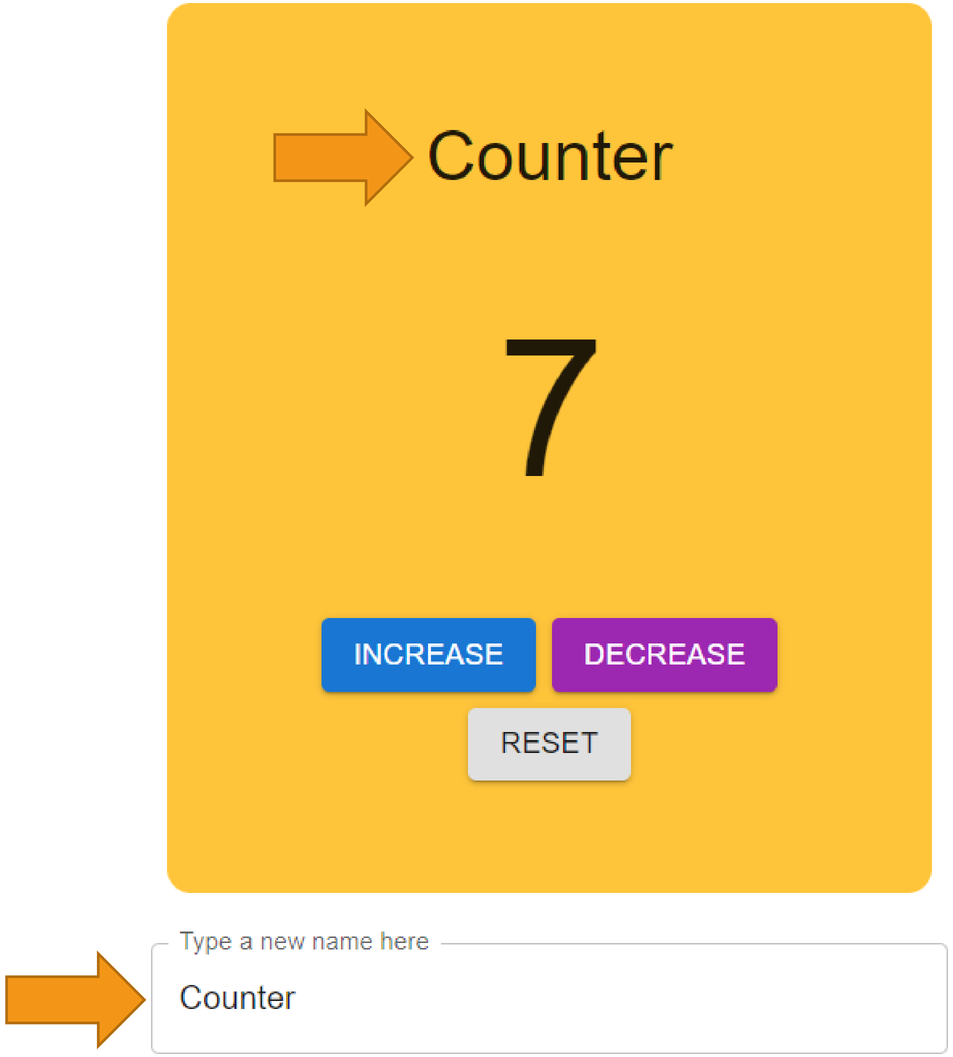


```
                                                     App.jsx:13
  SyntheticBaseEvent {_reactName: 'onChange', _targetInst: null, type: 'change', nat
▼ iveEvent: InputEvent, target:
  input#:r1:.MuiInputBase-input.MuiOutlinedInput
  , …} i
    bubbles: true
    cancelable: false
    currentTarget: null
    defaultPrevented: false
    eventPhase: 3
  ▶ isDefaultPrevented: ƒ functionThatReturns
  ▶ isPropagationStopped: ƒ functionThatRetur
    isTrusted: true
  ▶ nativeEvent: InputEvent {isTrusted: true,
  ▼ target: input#:r1:.MuiInputBase-input.Mui
      value: "Counter"
    ▶ __reactEvents$tw8p2kgijy: Set(1) {'inva
```

Data Binding



```
import React, {useState} from "react";
import Counter from "./Counter";

// App Component
export default function App() {
  const [count, setCount] = useState(7);
  const [name, setName] = useState("Counter App")

  const handleChangeName = (event) => {
    console.log(event);
    setName(event.target.value);
  };

  return (
    <div>
      <Counter name={name} count={count}
         increaseCount={increaseCount}
         descreaseCount={decreaseCount}
         resetToDefault={resetToDefault}
      />
      <input
        onChange={handleChangeName}
        placeholder="Type a new name here"
      />
    </div>
  );
}
```



Type a new name here

Counter

Summary

- Components are the building blocks of UI in a React app
- States hold the data and can be manipulated by using React Hooks
- Virtual DOM is faster than real DOM
- JSX is a mix of HTML, CSS, and JS
- Event handlers provide interaction in React app
- Props are states and methods passed to child components
- Model cannot be manipulated via view directly

React Lecture 1

✓Components

✓States

✓JSX, React Element, and Virtual DOM

✓Event handlers

✓Props

✓Data binding

What’s Next?

Next week

- React Lecture 2 on April 17 at 12:00
- React Hands-on 1 on April 19 at 14:00

Upcoming week

- React Hands-on 2 on April 24 at 12:00

React Lecture 2

- Component lifecycle
- React Router
- Redux
- Discussion
- Installation Guide
- Project Demo

Example Source Code – CodeSandbox Links

- Components
- States
- JSX, React Element, and Virtual DOM
- Event Handlers
- Event Handlers – Passing Arguments
- Event Handlers – Event Param/Properties/Object
- Props
- Props – Map method
- Data Binding

References

State

- https://daveceddia.com/why-not-modify-react-state-directly/
- https://stackoverflow.com/questions/37755997/why-cant-i-directly-modify-a-components-state-really
- https://www.javatpoint.com/react-state

React Element & JavaScript XML (JSX)

- https://www.javatpoint.com/react-fragments
- https://stackoverflow.com/questions/47761894/why-are-fragments-in-react-16-better-than-container-divs
- https://babeljs.io/docs/en/
- https://babeljs.io/repl

Event handlers

- https://www.freecodecamp.org/news/javascript-events-explained-in-simple-english/
- https://gist.github.com/fongandrew/f28245920a41788e084d77877e65f22f
- https://reactjs.org/docs/events.html
- https://reactjs.org/docs/handling-events.html
- https://www.w3schools.com/react/react_events.asp

References

- https://dev.to/nagwan/react-synthetic-events-34e5
- https://stackoverflow.com/questions/42597602/react-onclick-pass-event-with-parameter
- https://stackoverflow.com/questions/32782922/what-do-multiple-arrow-functions-mean-in-javascript
- https://medium.com/byte-sized-react/what-is-this-in-react-25c62c31480#:~:text=The%20'this'%20keyword%20typically%20references,or%20context%20of%20its%20use.
- https://stackoverflow.com/questions/38046970/react-component-this-is-not-defined-when-handlers-are-called
- https://gist.github.com/dfoverdx/2582340cab70cff83634c8d56b4417cd

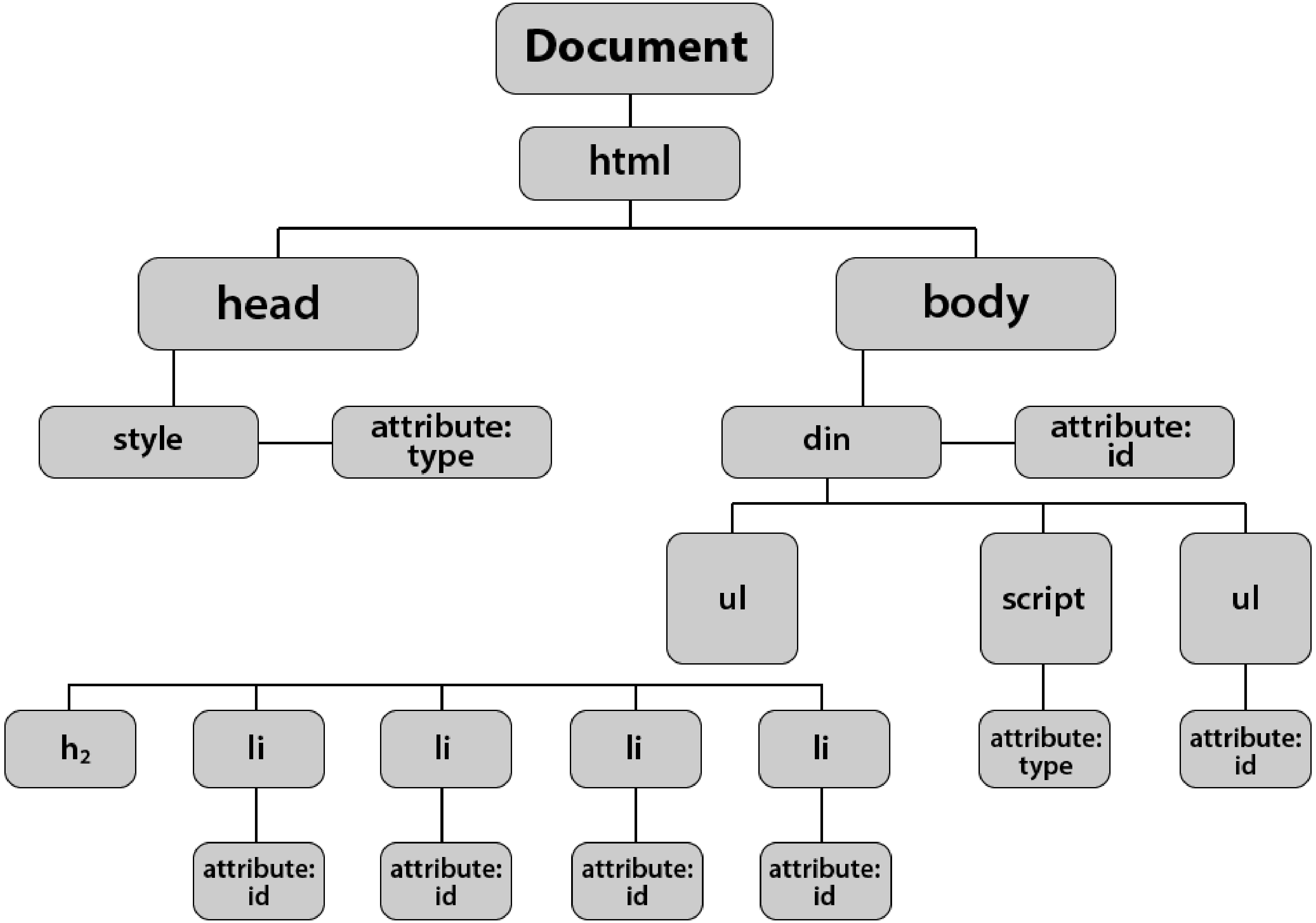
Props

- https://www.javatpoint.com/react-props
- https://reactjs.org/docs/components-and-props.html
- https://www.w3schools.com/react/react_props.asp
- https://ui.dev/react-router-v4-pass-props-to-components/

Data binding

- https://stackoverflow.com/questions/34519889/can-anyone-explain-the-difference-between-reacts-one-way-data-binding-and-angula

Appendix: DOM Tree



Source: https://medium.com/@josephchavez_33756/the-dom-tree-no-its-not-an-actual-tree-566cff758672