

Starting soon!

# React (pt. 3)

JFSS CS Club 2022/23  
Aritro



The background features several wavy, horizontal lines composed of small, dark blue dots. These lines are arranged in a series of parallel, undulating bands that sweep across the frame from the bottom left towards the top right, creating a sense of motion and depth against the solid blue background.

# Recap

**Answer:** Extension on JavaScript, allows HTML in JS

# What is JSX?

**Answer:** Independent and reusable code for your UI

# What are components?



```
export default function App() {  
  const [value, setValue] =  (0);  
  
  return (  
    <main>  
      <h1>Counter: {value}</h1>  
      <button onClick={() =>  (value + 1)}>  
        Increase  
      </button>  
    </main>  
  )  
}
```

# Hooks

The background features a solid blue color with several wavy, dotted lines in a lighter blue shade. These lines flow from the bottom left towards the top right, creating a sense of movement and depth.

# Hooks

- Allows us to make our components dynamic
  - Respond to changes in our code (ex. vars)
- Important hooks
  - useState (already covered)
  - **useEffect**
  - useContext (not as important)

# useEffect Hook

- Allows us to run code after UI has rendered (on screen)
- Generally runs on / after:
  - First render (when it first shows up on screen)
  - State update



# useEffect Hook

- General uses:
  - Data fetching
  - Directly changing DOM (ex. title)
  - Using timer functions (ex. setTimeout)

# Basic useEffect Usage

```
export default function App() {  
  const [value, setValue] = useState(0);  
  
  useEffect(() => {  
    console.log("Current value:", value);  
  });  
  
  return (  
    <main>  
      <h1>Counter: {value}</h1>  
      <button onClick={() => setValue(value + 1)}>  
        Increase  
      </button>  
    </main>  
  )  
}
```

function to run  
on every render /  
state change

# Dependencies

- By default, runs on every state change
  - Imagine expensive action in `useEffect` (data fetch)
    - Just want one fetch, not multiple
- Can specify when `useEffect` should run

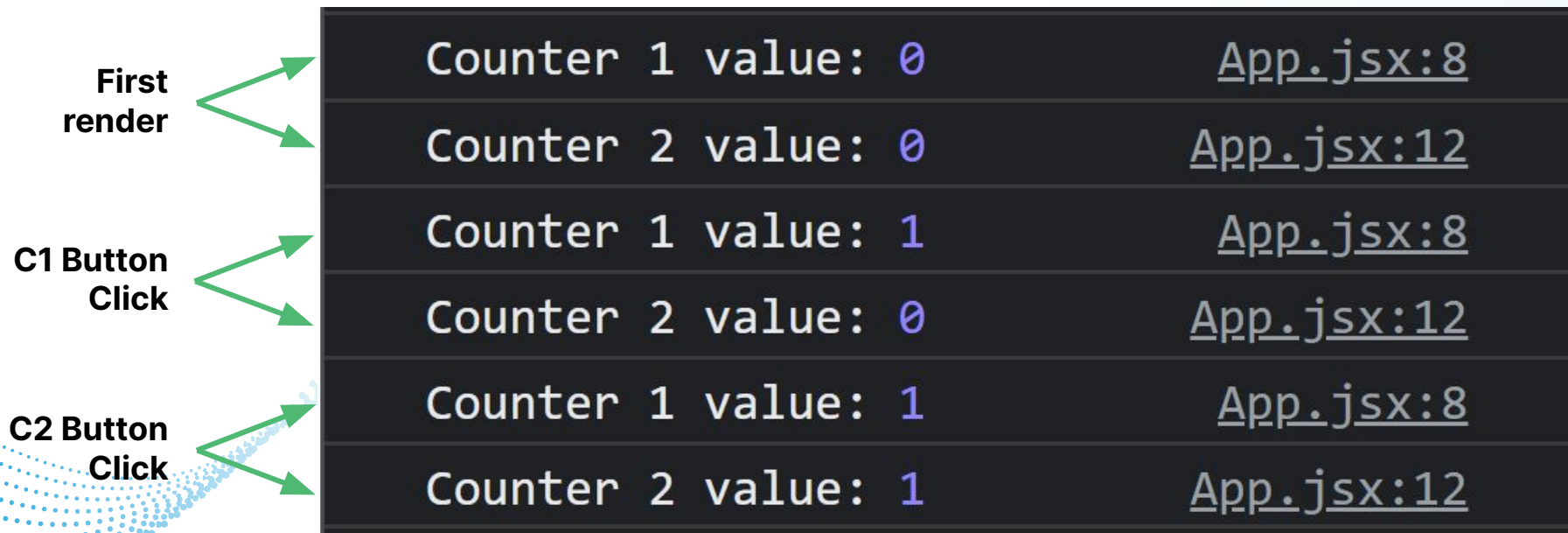
# Dependency Array

- Extra parameter in useEffect
  - `useEffect(callback, [dependencies])`
- Can add variables to it
  - When vars change, useEffect runs
  - If vars don't change, useEffect doesn't run

# Without Dependencies

```
export default function App() {  
  const [counter1, setCounter1] = useState(0);  
  const [counter2, setCounter2] = useState(0);  
  
  useEffect(() => {  
    console.log("Counter 1 value:", counter1);  
  });  
  
  useEffect(() => {  
    console.log("Counter 2 value:", counter2);  
  });  
  
  return (  
    <main>  
      <h1>Counter 1: {counter1}</h1>  
      <h1>Counter 2: {counter2}</h1>  
  
      <button onClick={() => setCounter1(counter1 + 1)}>  
        Increase (C1)  
      </button>  
  
      <button onClick={() => setCounter2(counter2 + 1)}>  
        Increase (C2)  
      </button>  
    </main>  
  )  
}
```

# Without Dependencies



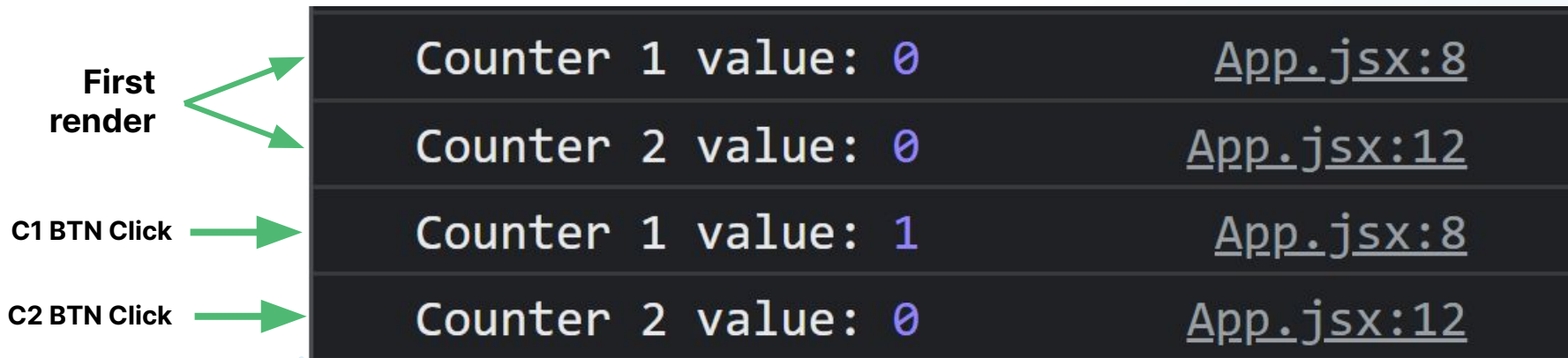
useEffect for other counter running when it shouldn't, not performant

# With Dependencies

```
export default function App() {  
  const [counter1, setCounter1] = useState(0);  
  const [counter2, setCounter2] = useState(0);  
  
  useEffect(() => {  
    console.log("Counter 1 value:", counter1);  
  }, [counter1]);  
  
  useEffect(() => {  
    console.log("Counter 2 value:", counter2);  
  }, [counter2]);  
  
  return (  
    <main>  
      <h1>Counter 1: {counter1}</h1>  
      <h1>Counter 2: {counter2}</h1>  
  
      <button onClick={() => setCounter1(counter1 + 1)}>  
        Increase (C1)  
      </button>  
  
      <button onClick={() => setCounter2(counter2 + 1)}>  
        Increase (C2)  
      </button>  
    </main>  
  )  
}
```

Appropriate variable  
in each  
dependency array

# Without Dependencies



Only runs when specific state changes



# Trying useEffect

The background is a solid blue color. It features several decorative wave-like patterns made of small white dots. These patterns are arranged in a series of overlapping, undulating lines that flow from the bottom left towards the top right, creating a sense of movement and depth.



**Free templates for all your presentation needs**



For PowerPoint and  
Google Slides



100% free for personal  
or commercial use



Ready to use,  
professional and  
customizable



Blow your audience  
away with attractive  
visuals