STATE, EVENTS, AND FORMS: INTERACTIVE COMPONENTS

JONAS SCHMEDTMANN

THE ULTIMATE REACT COURSE

@JONASSCHMEDTMAN

SECTION

STATE, EVENTS, AND FORMS: INTERACTIVE COMPONENTS

LECTURE

WHAT IS STATE IN REACT?

📚 WHAT WE NEED TO LEARN

WHAT REACT DEVELOPERS NEED TO LEARN ABOUT STATE:

1. **What is state** and **why** do we need it?

This section

2. How to use state in **practice**?
   - 👉 `useState`
   - 👉 `useReducer`
   - 👉 Context API
3. **Thinking** about state
   - 👉 When to use state
   - 👉 Where to place state
   - 👉 Types of state

Rest of the course...

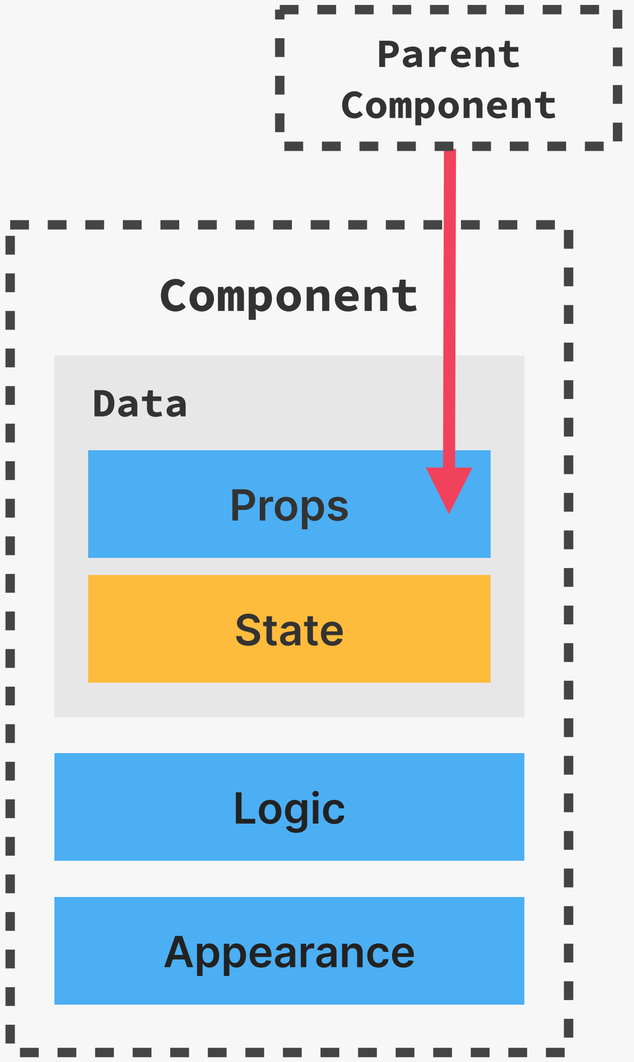
👋 **State** is the most important concept in React

(So we will keep learning about state throughout the entire course...)

WHAT IS **STATE**?

STATE

- 👉 Data that a component **can hold over time**, necessary for information that it needs to **remember** throughout the app's lifecycle
- 👉 **"Component's memory"** 

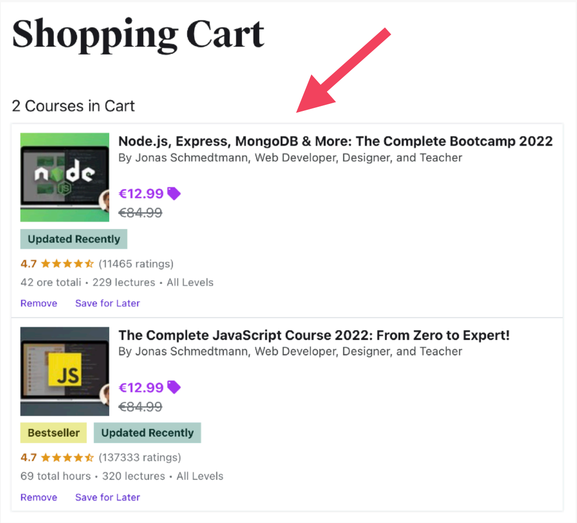


WHAT IS **STATE**?

STATE

- 👉 Data that a component **can hold over time**, necessary for information that it needs to **remember** throughout the app's lifecycle
- 👉 **"Component's memory"** 
- 👉 **"State variable" / "piece of state"**: A single variable in a component (component state)

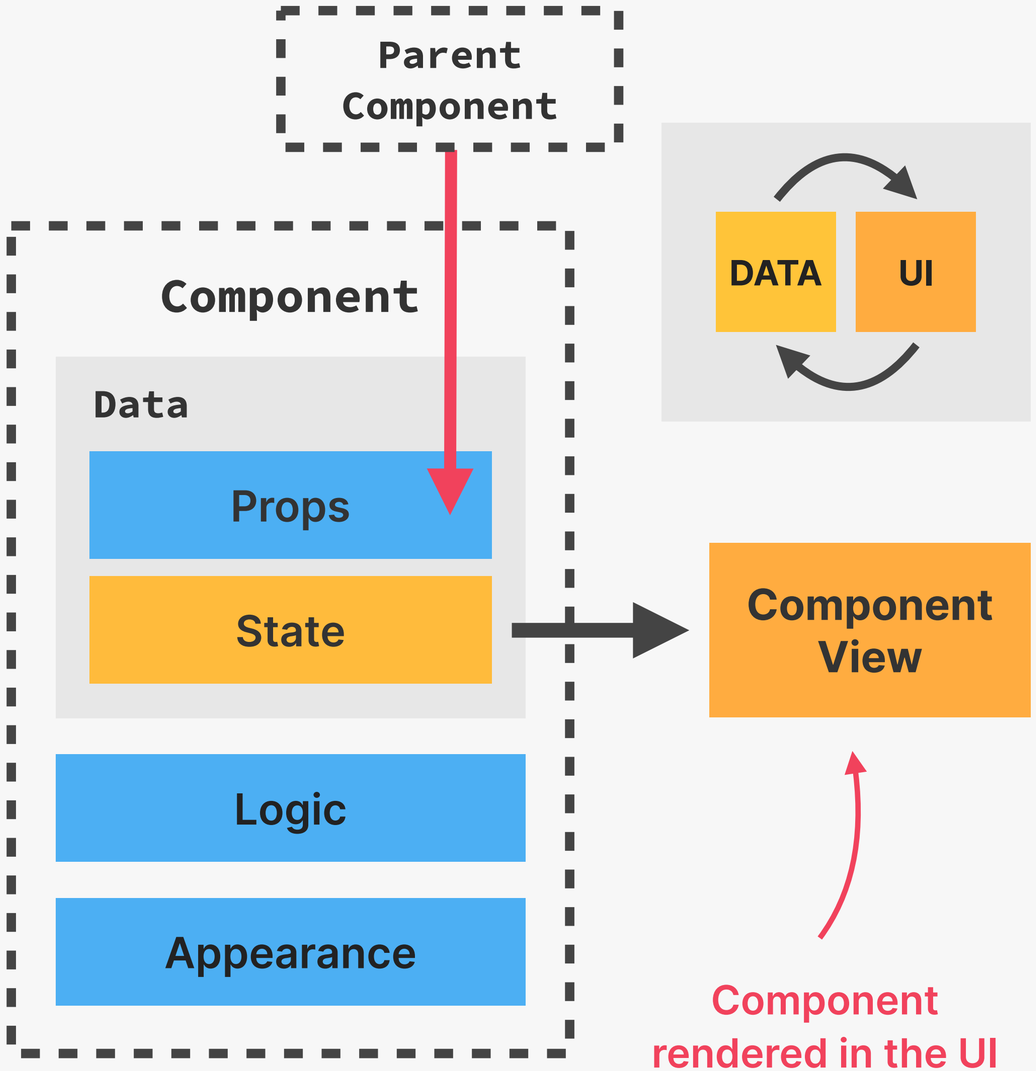
We use these terms interchangeably



WHAT IS **STATE**?

STATE

- 👉 Data that a component **can hold over time**, necessary for information that it needs to **remember** throughout the app's lifecycle
- 👉 **"Component's memory"** 
- 👉 ***Component* state:** Single local component variable ("Piece of state", "state variable")
- 👉 Updating **component state** triggers React to **re-render the component**



WHAT IS **STATE**?

STATE

- 👉 Data that a component **can hold over time**, necessary for information that it needs to **remember** throughout the app's lifecycle
- 👉 **"Component's memory"** 
- 👉 ***Component* state:** Single local component variable ("Piece of state", "state variable")
- 👉 Updating **component state** triggers React to **re-render the component**

STATE ALLOWS DEVELOPERS TO:

1. Update the component's view (by re-rendering it)
2. Persist local variables between renders

👋 *State is a **tool**. Mastering state will unlock the power of React development*

JONAS SCHMEDTMANN

THE ULTIMATE REACT COURSE

@JONASSCHMEDTMAN

SECTION

STATE, EVENTS, AND FORMS: INTERACTIVE COMPONENTS

LECTURE

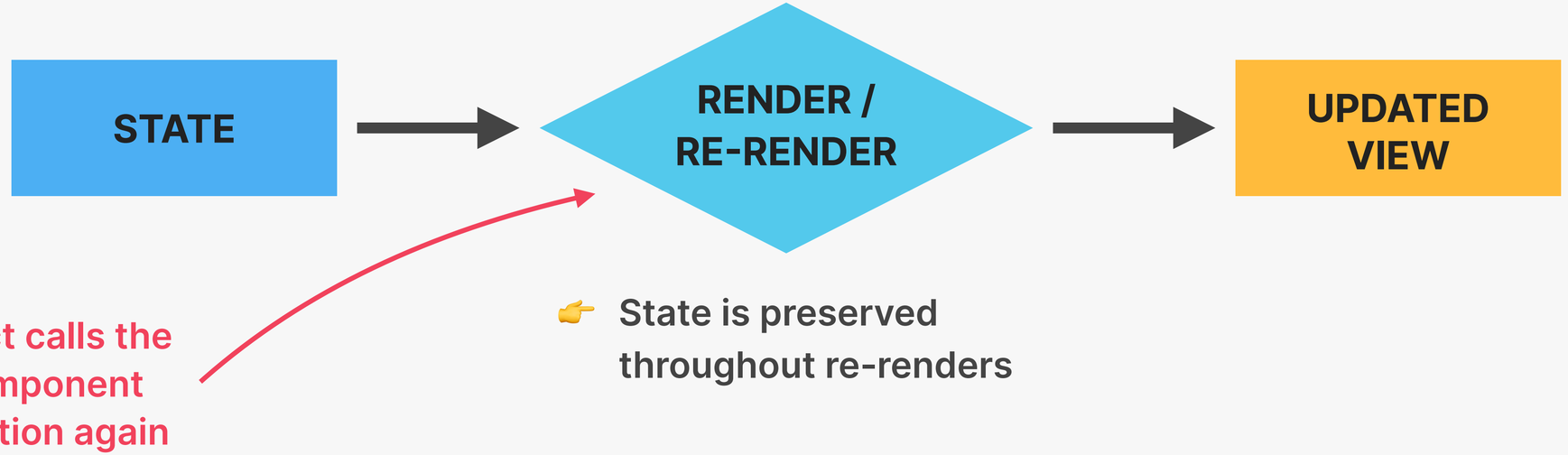
THE MECHANICS OF STATE

THE **MECHANICS** OF STATE IN REACT

We don't do direct DOM manipulations → How is a component view updated then? → *In React, a view is updated by re-rendering the component* → A component is re-rendered when its state is updated

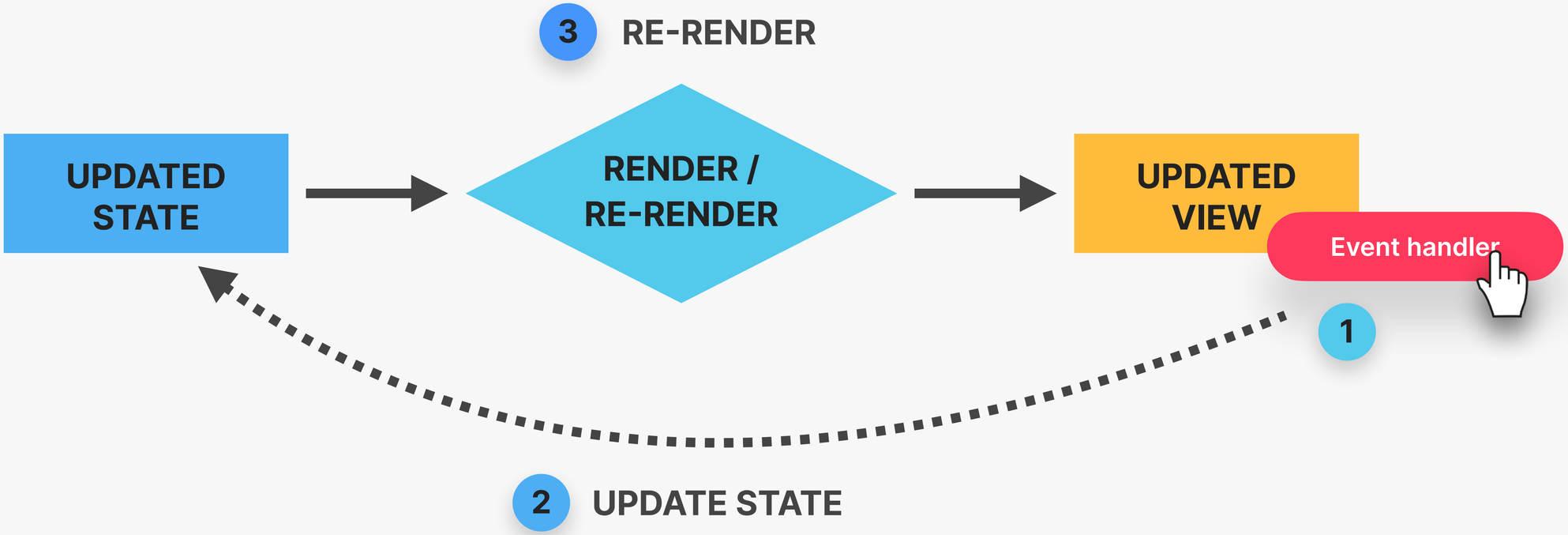
Because React is **declarative**

Important React principle



THE **MECHANICS** OF STATE IN REACT

We don't do direct DOM manipulations → How is a component view updated then? → *In React, a view is updated by re-rendering the component* → A component is re-rendered when its state is updated

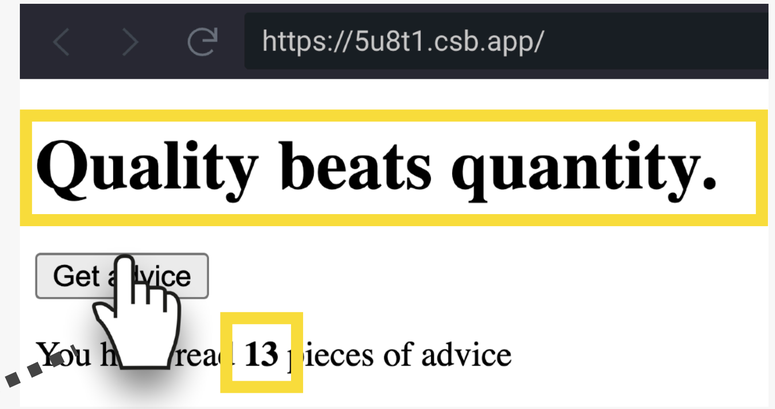


THE **MECHANICS** OF STATE IN REACT

We don't do direct DOM manipulations → How is a component view updated then? → *In React, a view is updated by re-rendering the component* → A component is re-rendered when its state is updated → **So to update a view, we update state**

```
const [advice, setAdvice] =
  useState("Quality beats quantity.");
const [countAdvice, setCountAdvice] =
  useState(13);
```

RE-RENDER



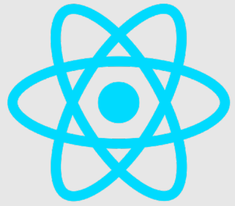
UPDATE STATE

```
setAdvice(data.slip.advice);
setCountAdvice((count) => count + 1);
```

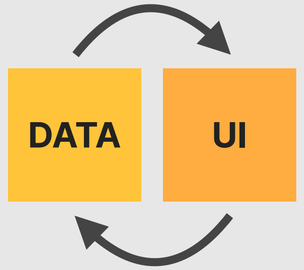
THE **MECHANICS** OF STATE IN REACT

We don't do direct DOM manipulations → How is a component view updated then? → ***In React, a view is updated by re-rendering the component*** → A component is re-rendered when its state is updated → **So to update a view, we update state**

👉 *React is called "React" because...*



REACT *REACTS* TO STATE CHANGES BY RE-RENDERING THE UI



JONAS SCHMEDTMANN

THE ULTIMATE REACT COURSE

@JONASSCHMEDTMAN

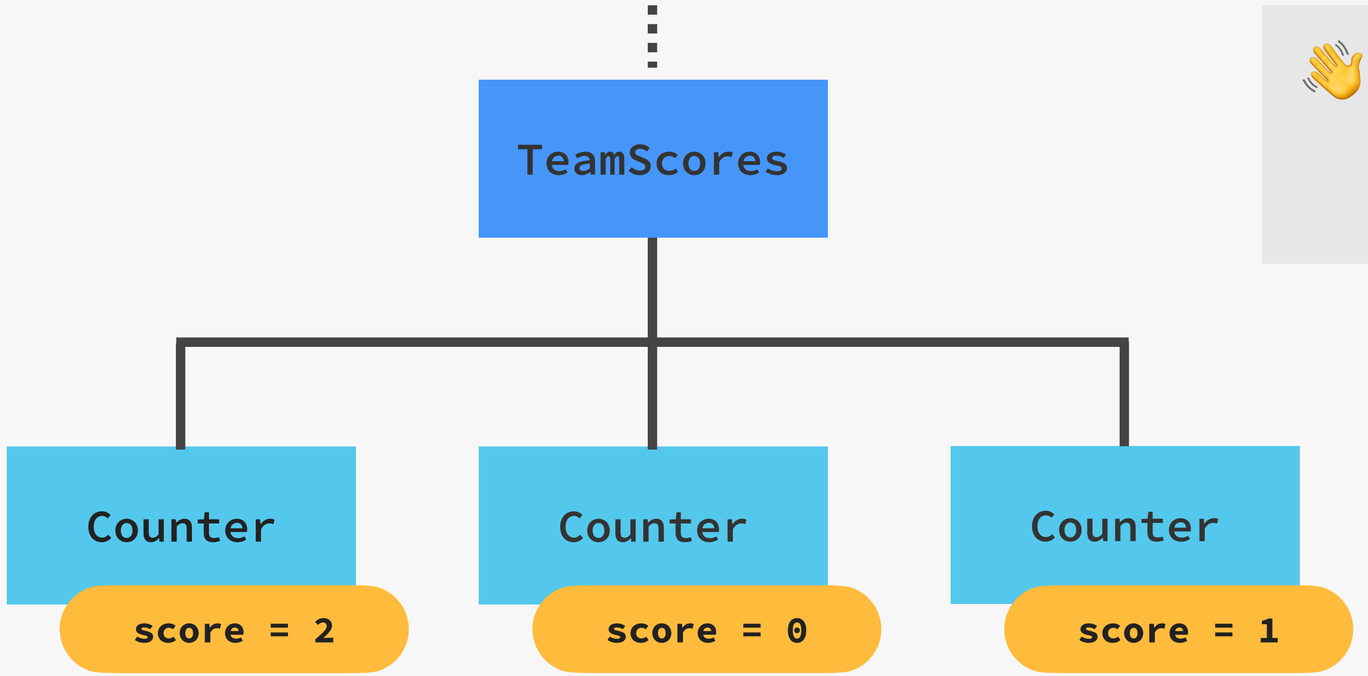
SECTION

STATE, EVENTS, AND FORMS:
INTERACTIVE COMPONENTS

LECTURE

MORE THOUGHTS ABOUT STATE
+ STATE GUIDELINES

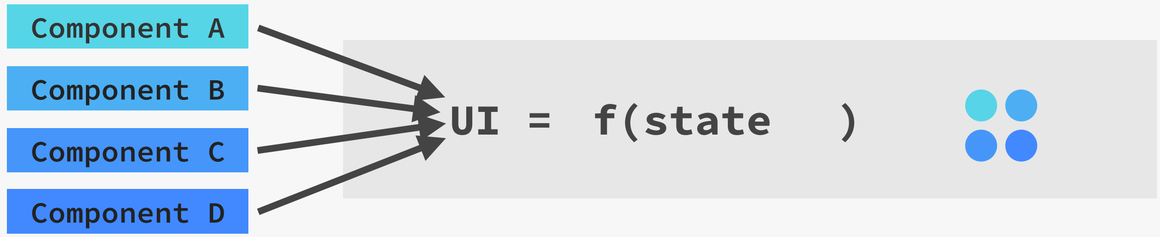
ONE COMPONENT, ONE **STATE**

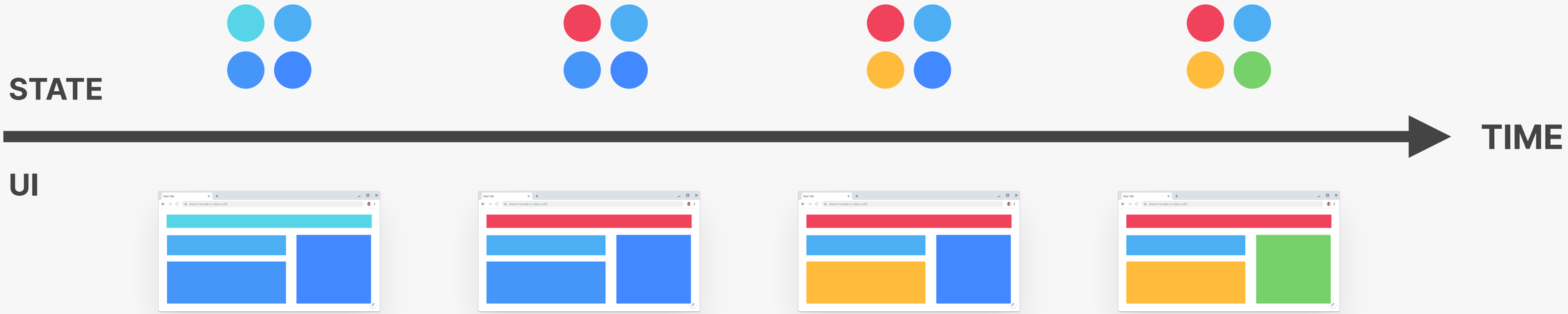


👋 Each component has and manages **its own state**, no matter how many times we render the same component

Multiple *instances* of the `Counter` component

UI AS A **FUNCTION** OF STATE





DECLARATIVE, *REVISITED*

- 👉 With state, we view UI as a **reflection of data changing over time**
- 👉 We ***describe*** **that reflection** of data using state, event handlers, and JSX

🎯 IN **PRACTICAL** TERMS...

PRACTICAL GUIDELINES ABOUT STATE

- 👉 Use a state variable for any data that the component should keep track of ("remember") over time. **This is data that will change at some point**. In Vanilla JS, that's a `let` variable, or an `[]` or `{}`

- 👉 Whenever you want something in the component to be **dynamic**, create a piece of state related to that "thing", and update the state when the "thing" should change (aka "be dynamic")

  - 👉 ***Example:*** *A modal window can be open or closed. So we create a state variable* `isOpen` *that tracks whether the modal is open or not. On* `isOpen = true` *we display the window, on* `isOpen = false` *we hide it.*

- 👉 If you want to change the way a component looks, or the data it displays, **update its state.** This usually happens in an **event handler** function.

- 👉 When building a component, imagine its view as a **reflection of state changing over time**

- 👉 For data that should not trigger component re-renders, **don't use state**. Use a regular variable instead. This is a common **beginner mistake**.

JONAS SCHMEDTMANN

THE ULTIMATE REACT COURSE

@JONASSCHMEDTMAN

SECTION

STATE, EVENTS, AND FORMS: INTERACTIVE COMPONENTS

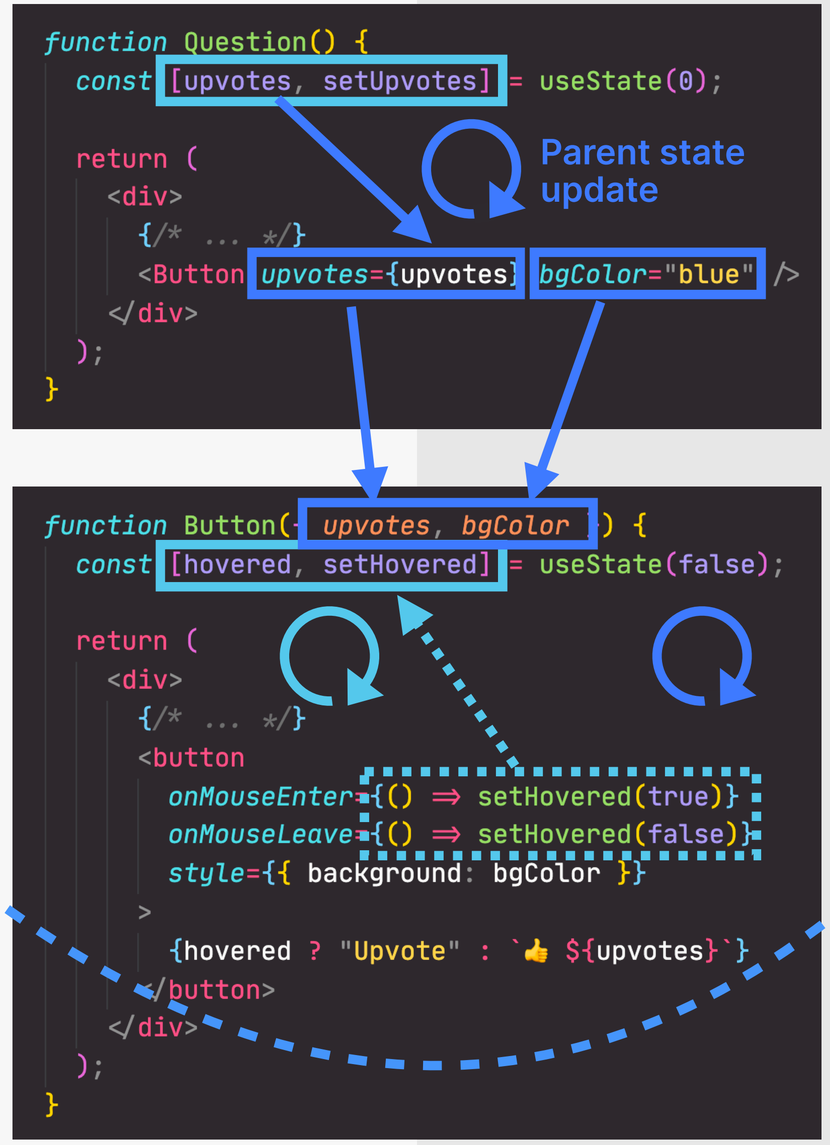
LECTURE

STATE VS. PROPS

STATE **VS.** PROPS

STATE

- 👉 **Internal** data, owned by component
- 👉 Component “memory”
- 👉 Can be updated by the component itself
- 👉 Updating state causes component to re-render
- 👉 Used to make components interactive



PROPS

- 👉 **External** data, owned by parent component
- 👉 Similar to function parameters
- 👉 Read-only
- 👉 **Receiving new props causes component to re-render.** Usually when the parent’s state has been updated
- 👉 Used by parent to configure child component (“settings”)