

## 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



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

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

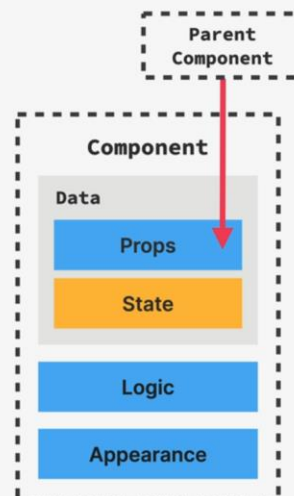
Scrimba

## 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"



Scrimba

What if a component needs to actually hold its own data and also hold it over time? What if we actually want to make our app interactive changing the UI as a result of an action?

That's where state comes into play.

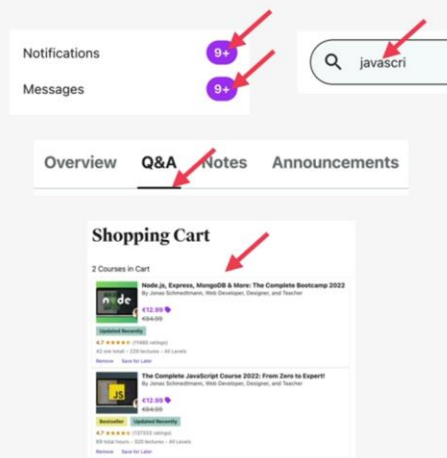
So, state is basically data that a component can hold over time and we use it for information that a component needs to remember throughout its lifecycle. Therefore, we can think of state as being the memory of a component.

## 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"



Scalmy

For example, like a notification count, the text content of an input field, or the active tab in a tab component. It can also be a bit more complex data, for example, the content of a shopping cart.

What all these pieces of state have in common is that in the application, the user can easily change these values. For example, when they read a notification, the count will go down by one, or when they click on another tab, that tab will become active. Therefore, each of these components needs to be able to hold this data over time i.e. over the lifecycle of the application.

## WHAT IS STATE?

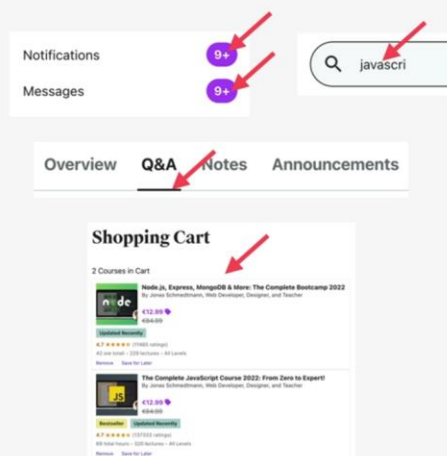
### STATE

- 👉 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)





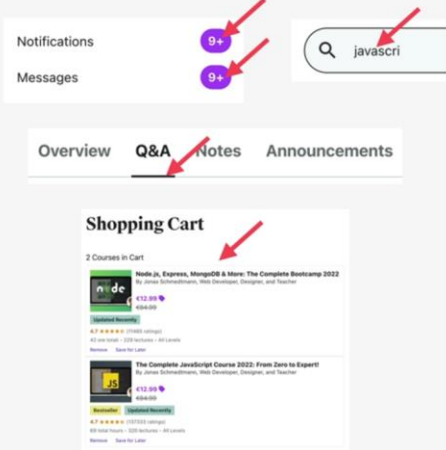
Scalmy

For that reason, each of these pieces of information is a piece of state. So, a piece of state, or a state variable is just one single actual variable in the component that we can define in our code.

## WHAT IS STATE?


### STATE

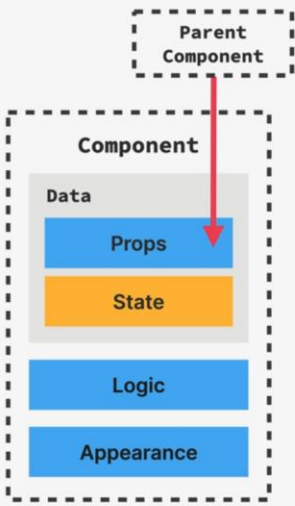
- 👉 Data that a component **can hold over time**, necessary for information that it needs to **remember** throughout the app's lifecycle
- 👉 "Component's memory" 
- 👉 "State variable" / "piece of state": A single variable in a component (component state)  
 We use these terms interchangeably



## 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**

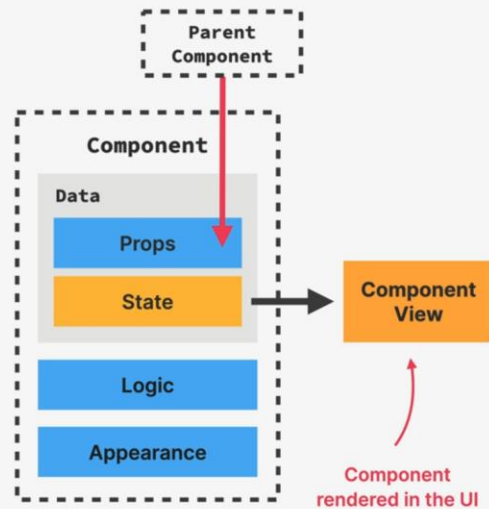


The most important aspect of state is the fact that updating state triggers React to re-render the component in the user interface.

## 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**




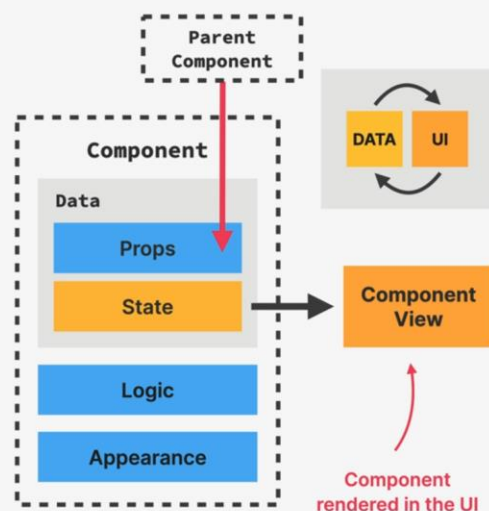
So, it will create a new updated view for that component. A component's view is basically just the component visually rendered on the user interface.

When one single component is rendered, we call that a view. So, all the views combined together then make up the final user interface.

## WHAT IS STATE?

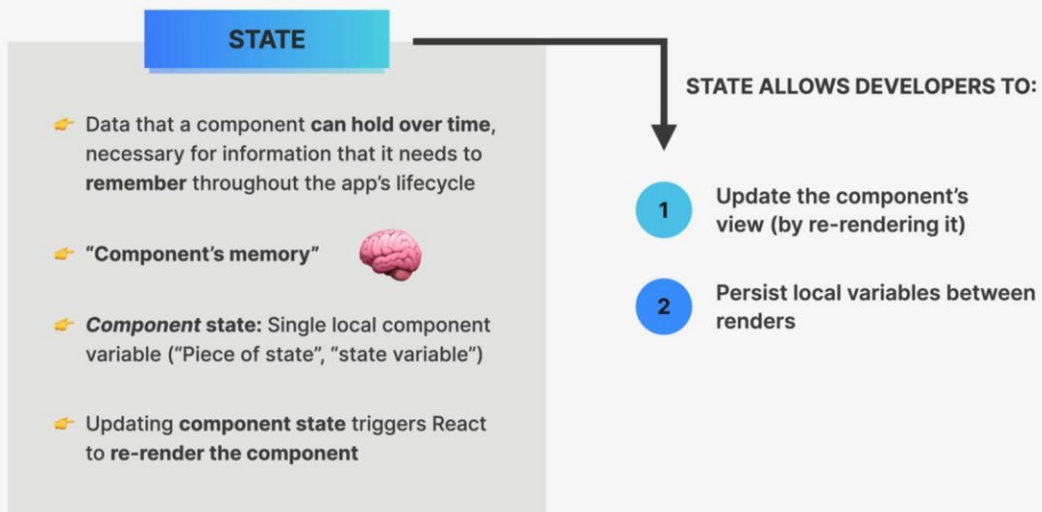
### STATE

- ✚ 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**



React automatically keeps data in sync with the UI. So, state is how React keeps the user interface in sync with data. We change the state, we change the UI.

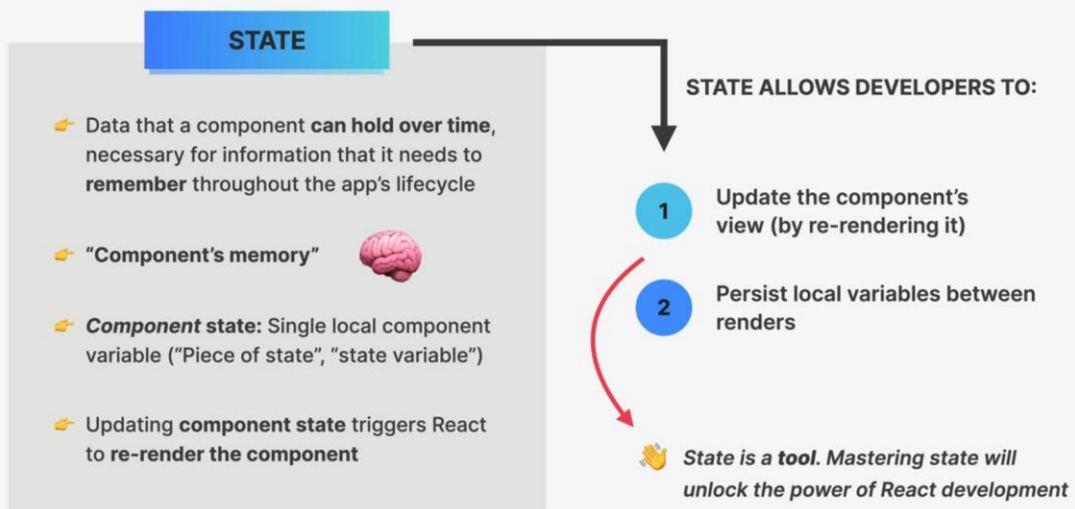
## WHAT IS STATE?



State allows developers to do two important things.

1. First, state allows us to update the component's view by re-rendering the component. So, it gives us a way to change part of the UI.
2. Second, state allows developers to persist local variables between multiple renders and re-renders.

## WHAT IS STATE?



State is the most powerful tool that we have in the world of React. So, understanding how a state works, what it does and understanding the mechanics of state will unlock the power of React development for you.