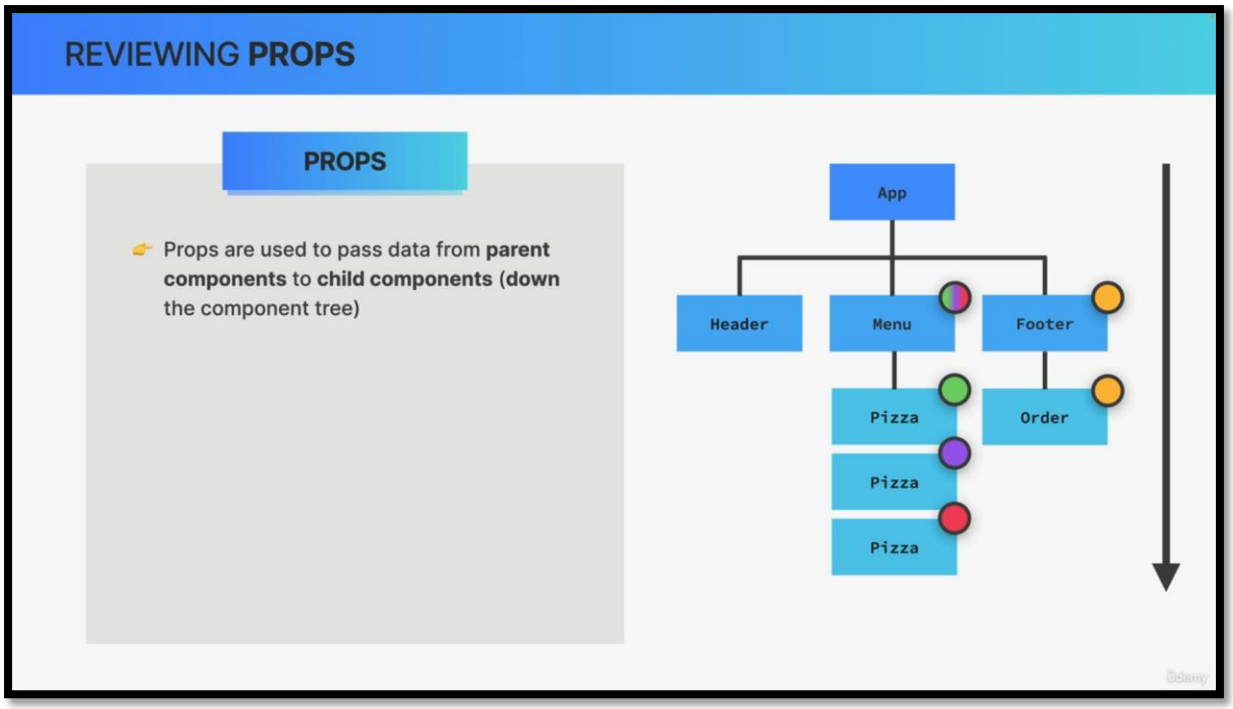
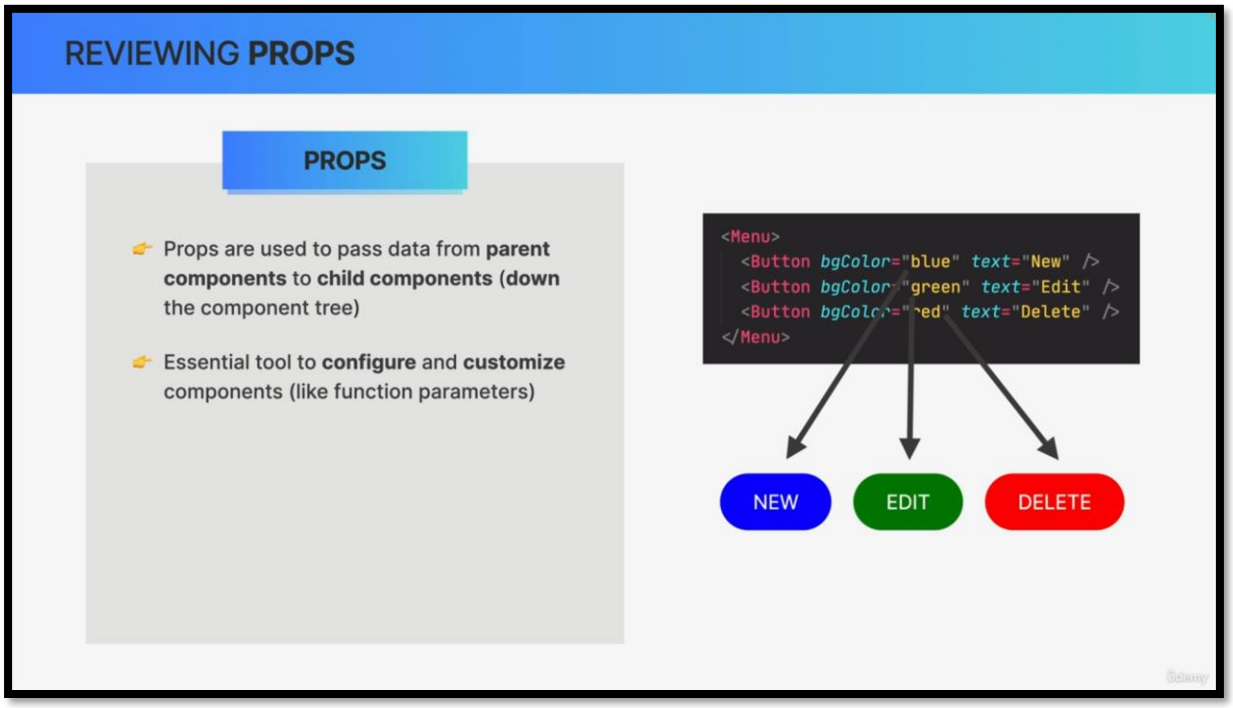
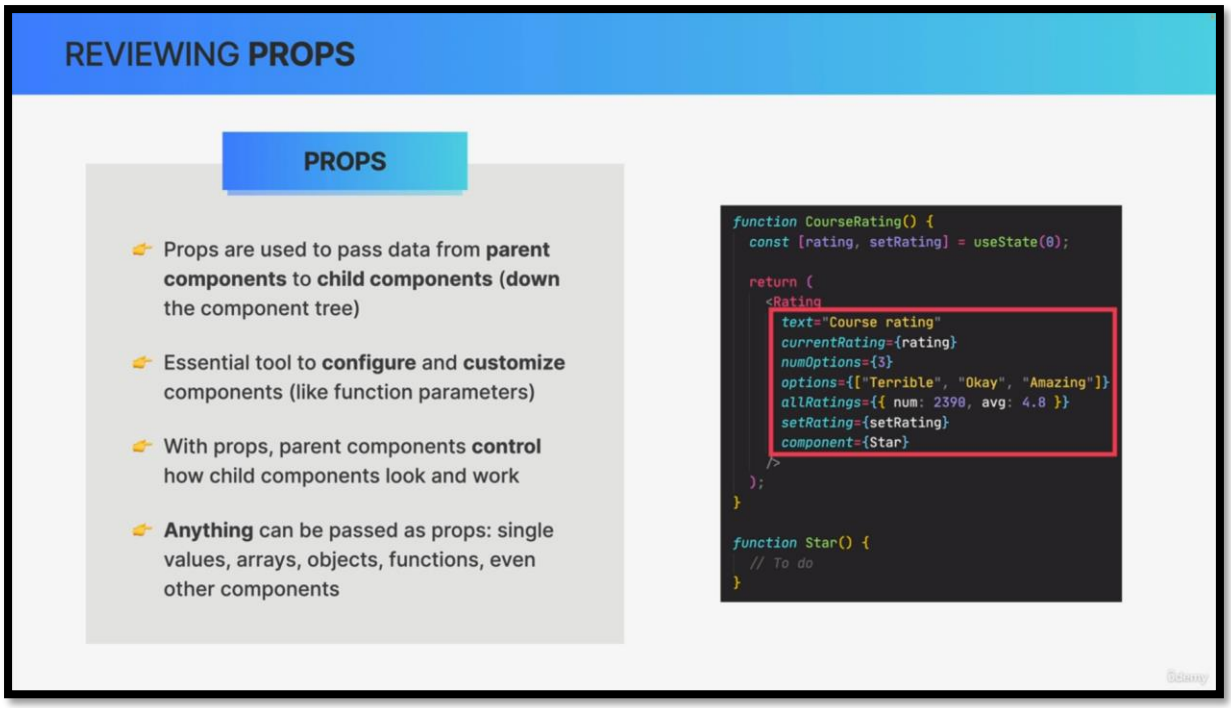


We use props in React to pass data from parent components to child components.

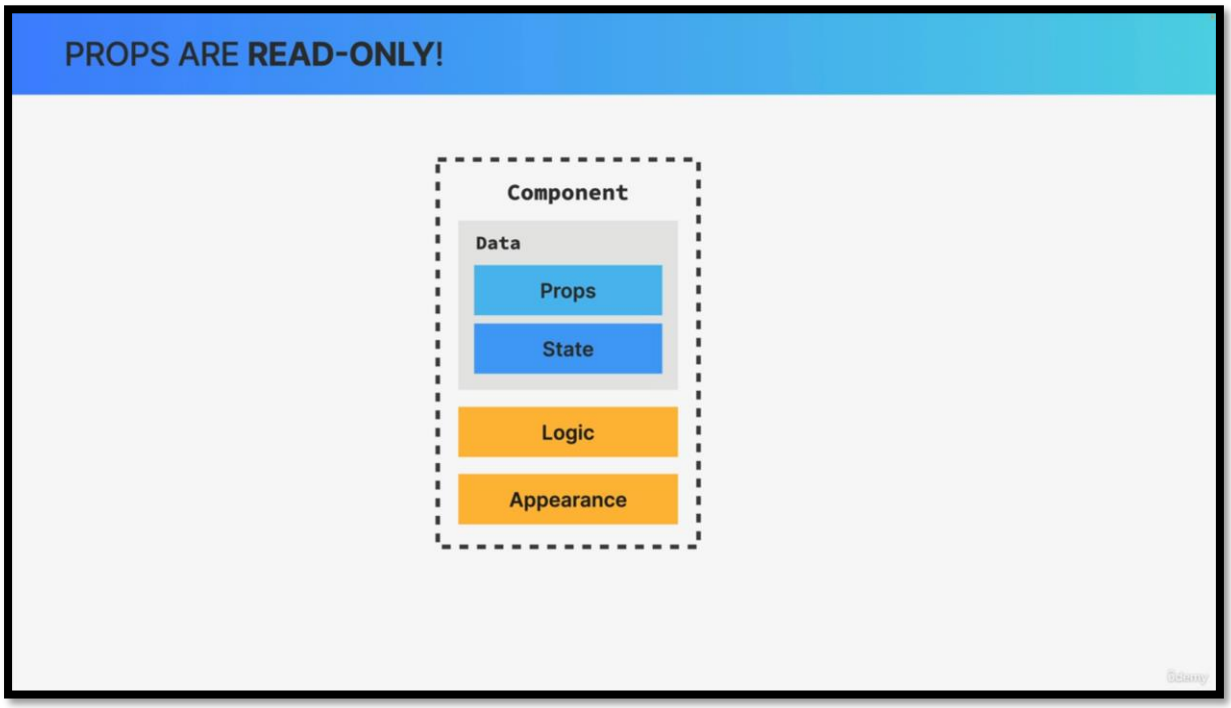


Essentially to pass information down the component tree. This means that we use props to communicate between parent and child components. Therefore, props are an essential React tool to configure and also to customize components.

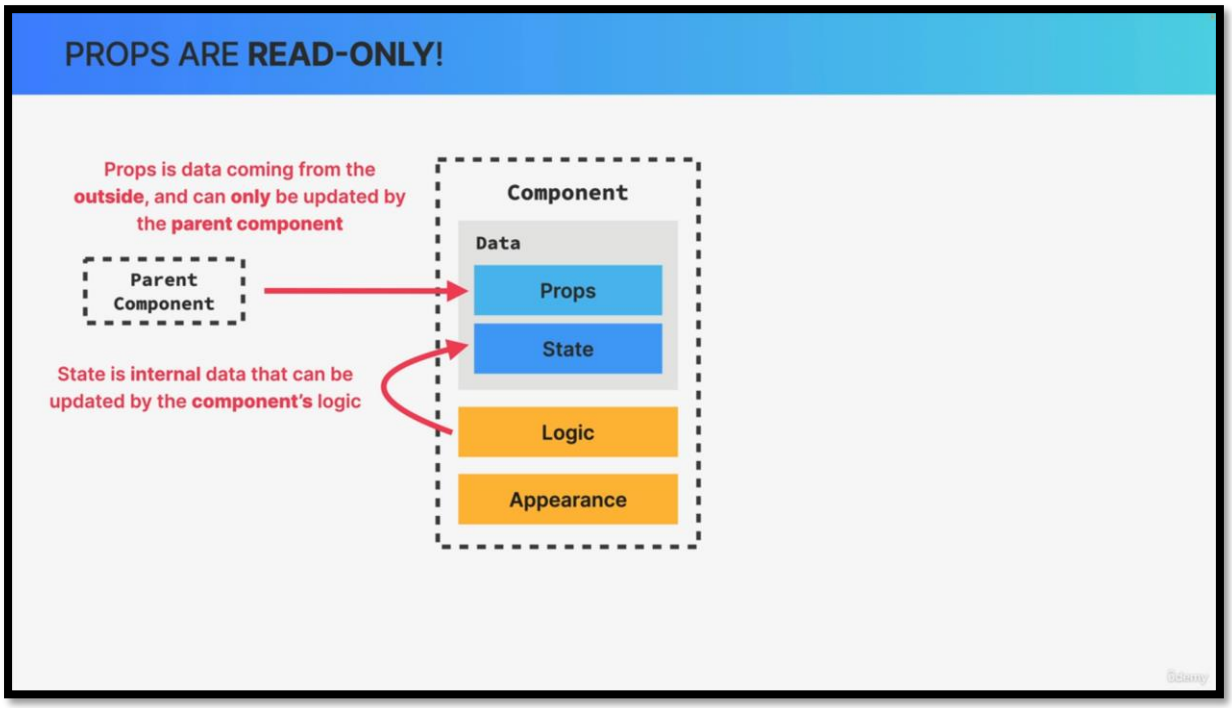


So, we can imagine props as settings that we can use to make a parent component control how its child component should look like and how it should work.

Props are just like arguments passed to regular JavaScript functions. We can pass anything into JavaScript functions. The same is true for props. We can pass any type of value as a prop like single values, array objects, functions and even other React components.

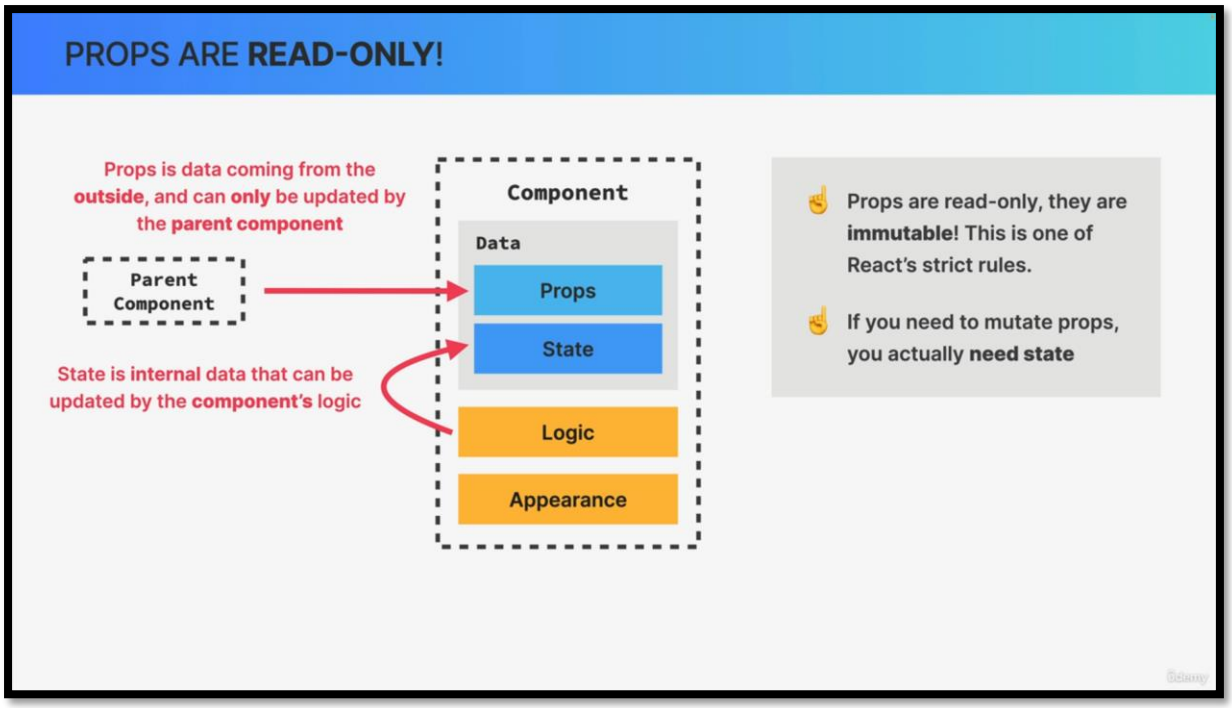


React renders a component based on its current data and that UI will always be kept in sync with that data.



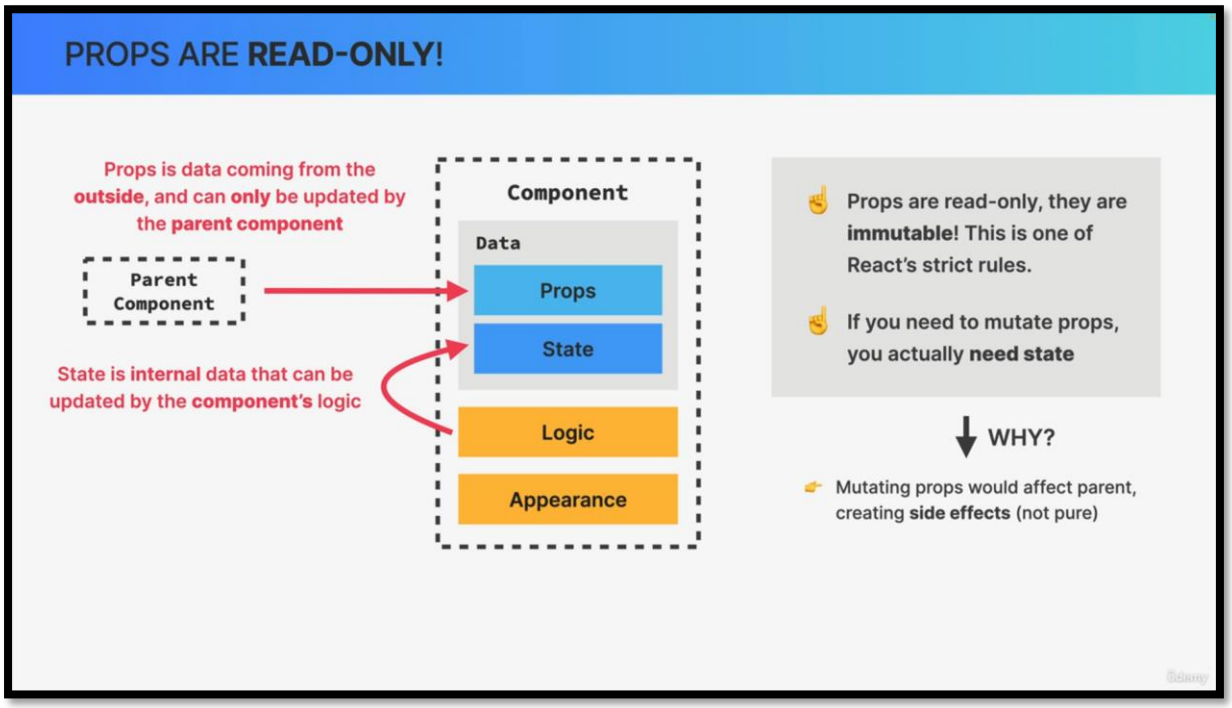
So, this data that React users to render a component is made out of props and state. State is basically internal component data that can be updated by the component's logic. While props on the other hand is data that is coming from the parent component i.e. from the outside basically.

It's the parent component who owns that data and so therefore it cannot be modified by the child component. Instead, props can only be updated by the parent component itself.



This brings us to one of the few strict rules that React gives us which is that props are immutable.

So, props cannot be changed, they are read-only. If at any point you feel like you need to mutate props what you need is state because state is for data that changes over time.

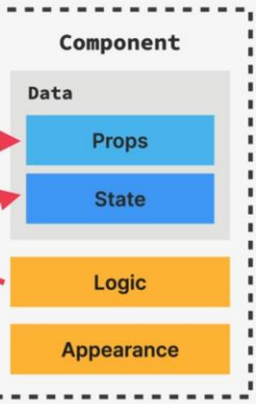


PROPS ARE **READ-ONLY!**

Props is data coming from the **outside**, and can **only** be updated by the **parent component**

Parent Component

State is internal data that can be updated by the **component's** logic



☝ Props are read-only, they are **immutable**! This is one of React's strict rules.

☝ If you need to mutate props, you actually **need state**

WHY?

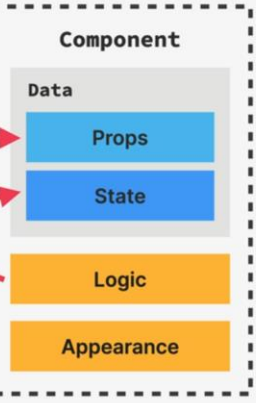
- Mutating props would affect parent, creating **side effects** (not pure)
- Components have to be **pure functions** in terms of props and state
- This allows React to optimize apps, avoid bugs, make apps predictable

PROPS ARE **READ-ONLY!**

Props is data coming from the **outside**, and can **only** be updated by the **parent component**

Parent Component

State is internal data that can be updated by the **component's** logic



☝ Props are read-only, they are **immutable**! This is one of React's strict rules.

☝ If you need to mutate props, you actually **need state**

WHY?

- Mutating props would affect parent, creating **side effects** (not pure)
- Components have to be **pure functions** in terms of props and state
- This allows React to optimize apps, avoid bugs, make apps predictable


```
let x = 7;

function Component(){
  x = 23
  return <h1>Number {x}</h1>
}
```

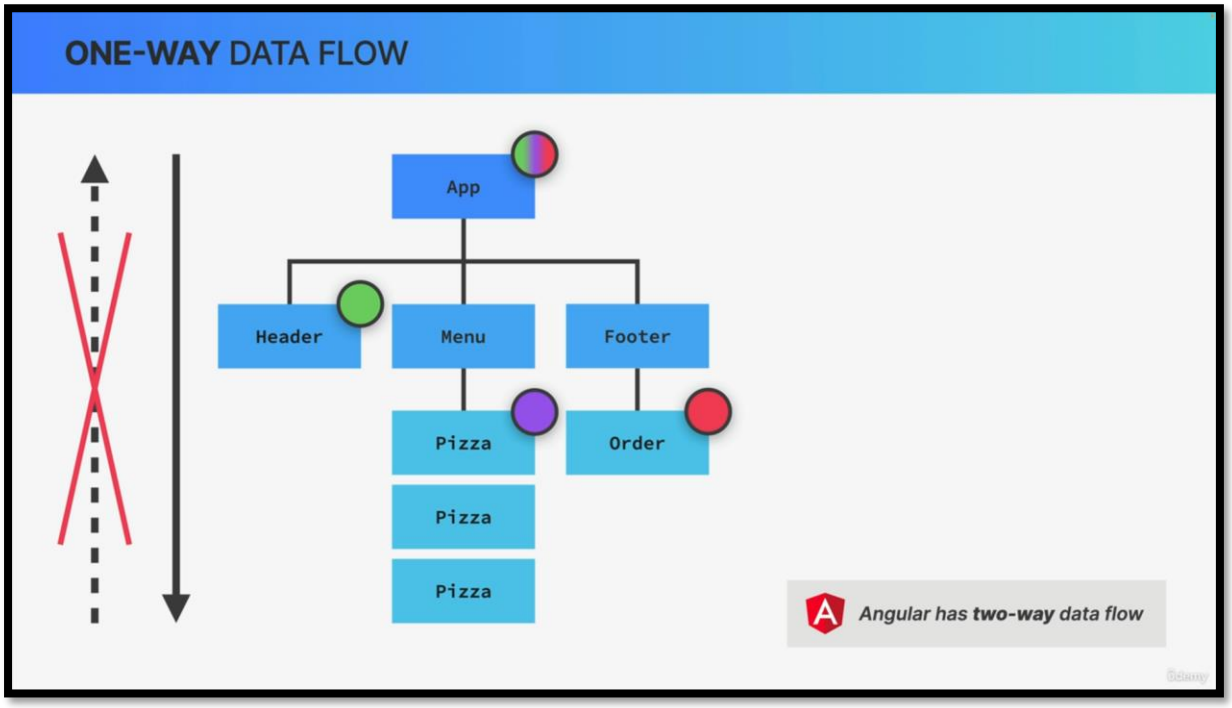
Don't do this!

Why are props immutable in React?

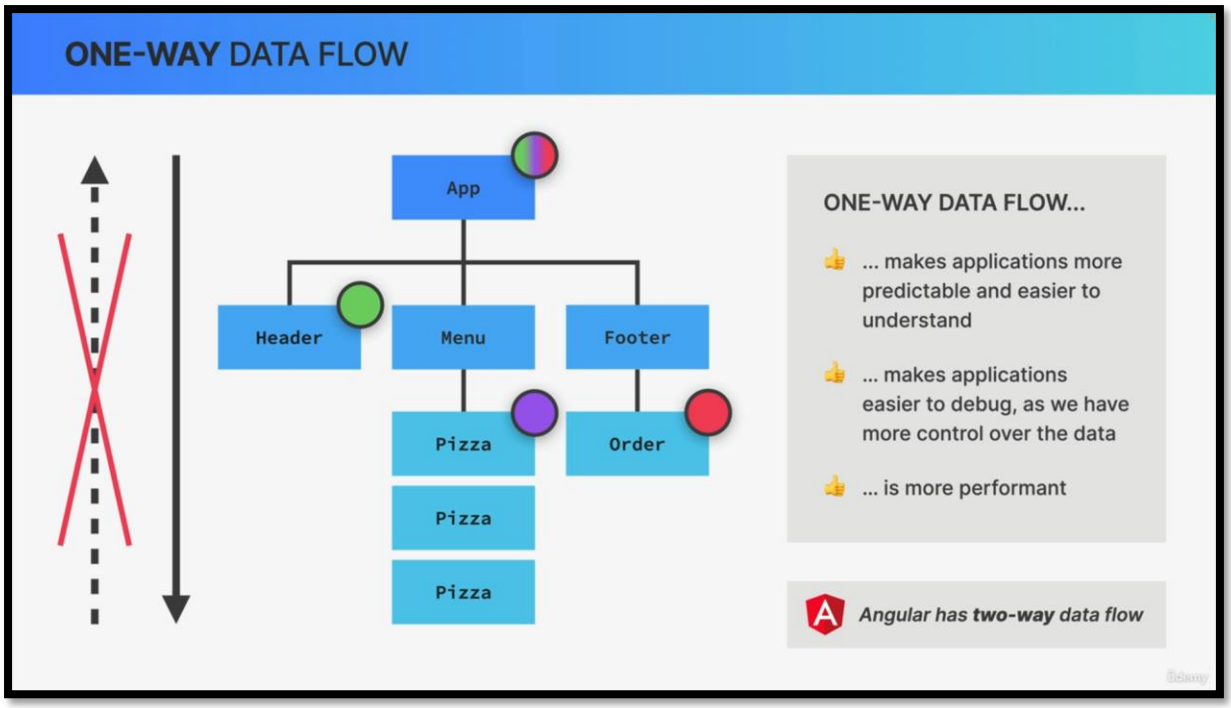
1. Props are just an object. Therefore, if you change the props object in your component you would also affect the parent component because that's just how objects work in JavaScript. So, when you copy an object and mutate the copy, the original object will also be mutated.

2. If you change an object that is located outside of the component function, that function has then created a side effect. React, however, is all about pure functions i.e. functions without side effects at least when it's about a components data.

3. Components have to be pure in terms of their props and state, because this allows React to optimize your application and it avoids some strange bugs that can appear when you manipulate external data.



React uses a so-called one-way data flow. One-way data flow means that in React applications, data can only be passed from parent to child components, which happens by using props. Data can flow from parents to children but never the opposite way.



There are multiple reasons why React uses a one-way data flow like this.

1. The first is that it makes applications way more predictable and way easier to understand for developers because it is just a lot easier to understand where the data is coming from if it only flows in one direction.

2. It makes applications way easier to debug again because we have way more control over the data and we understand exactly how that data flows around.

3. Two-way data binding is usually less efficient. So, it's less performant to implement.