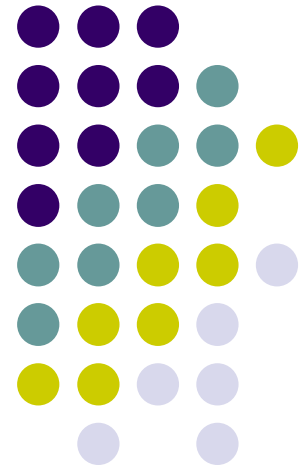


REACT



State

State

State allows components to create and manage their own data. So unlike props, components cannot pass data with state, but they can create and manage it internally.


Example

How to Use State

When you press the like button on Instagram, and you see an increment in the number of likes, this is what the state does.

```
class Test extends React.Component {  
  constructor() {  
    this.state = {  
      id: 1,  
      name: "test"  
    };  
  }  
  render() {  
    return (  
      <div>  
        <p>{this.state.id}</p>  
        <p>{this.state.name}</p>  
      </div>  
    );  
  }  
}
```

Update Component's State



```

this.state.id = "2020"; // wrong
this.setState({        // correct
  id: "2020"
});

```

What happens when state changes?

A change in the state happens based on user-input, triggering an event, and so on. Also, React components (with state) are rendered based on the data in the state. State holds the initial information.

So when state changes, React gets informed and immediately re-renders the DOM – not the whole DOM, but only the component with the updated state. This is one of the reasons why React is fast.

The `setState()` method triggers the re-rendering process for the updated parts. React gets informed, knows which part(s) to change, and does it quickly without re-rendering the whole DOM.

Pay attention to these 2 points while using state.

State shouldn't be modified directly – the `setState()` should be used

State affects the performance of your app, and therefore it shouldn't be used unnecessarily.

Props

(Stateless)

Remember

Props

Props is short for properties and they are used to pass data between React components.

React's data flow between components is uni-directional (from parent to child only).

A diagram on a dark blue background. At the top center is a code editor window with a dark gray background and a light gray border. It contains JavaScript code. To the left of the code window, a horizontal line extends from the left edge of the frame, ending in a small white circle. From this circle, a line goes down and then right, ending in another small white circle. To the right of the code window, a horizontal line extends from the right edge of the frame, ending in a small white circle. From this circle, a line goes down and then left, ending in a third small white circle. These three white circles are connected by a vertical line in the center, forming a U-shape that frames the text box below.

```
class ParentComponent extends Component {  
  render() {  
    return (  
      <ChildComponent name="First Child" />  
    );  
  }  
}  
  
const ChildComponent = (props) => {  
  return <p>{props.name}</p>;  
};
```


Firstly, we need to define/get some data from the parent component and assign it to a child component's "prop" attribute.

A component is an encapsulated piece of logic. For example, here's a component that displays a Hello, World! message:



```
function HelloWorld() {  
  return <span>Hello, World!</span>;  
}
```

Pass data with props




```
<ChildComponent name="First Child" />
```

"Name" is a defined prop here and contains text data. Then we can pass data with props like we're giving an argument to a function:



```
const ChildComponent = (props) => {  
  // statements  
};
```

And finally, we use dot notation to access the prop data and render it:



```
return <p>{props.name}</p>;
```

Difference between Props vs State

- Components receive data from **outside** with props, whereas they can create and manage their own data with state.
- Props are used to pass data, whereas state is for managing data.
- Data from props is read-only, and **cannot** be modified by a component that is receiving it from outside.
- State data can be modified by its own component, but is private (cannot be accessed from outside).
- Props can only be passed from parent component to child (unidirectional flow).
- Modifying state should happen with the `setState ()` method.



SUMMARY

- Props
- State
- React Life Cycle Intro

QUERIES

An abstract graphic featuring several overlapping circles in shades of green, blue, and yellow. A magnifying glass with a grey handle and frame is positioned over the right side of the image, focusing on the word 'QUERIES'. The background is dark blue with faint, blurred text that appears to be code or technical documentation.