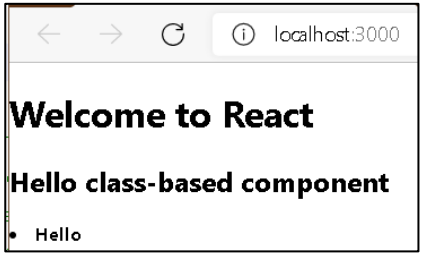
LIFECYCLE EVENTS

Lifecycle Events

- React class components can have hooks for several lifecycle events
  - During the lifetime of a component, there's a series of events that gets called, and to each event you can hook and provide custom functionality.
  - React lifecycle methods are a series of events that happen from the birth of a React component to its death.
  - There are 4 phases in a React component lifecycle:
    - initial
    - Mounting
    - Updating
    - Unmounting

```
class HelloComponent extends Component{
  constructor(){
      super();
       this.state = {message: "hello" }
  }
  render(){
     return (<h2>{this.state.message}</h2>)
  }
}
```



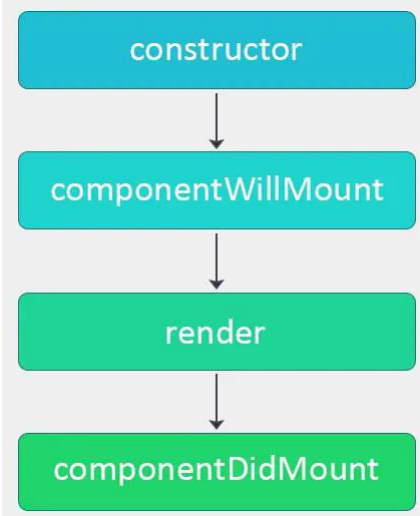
Lifecycle methods

- React lifecycle methods:
  - Each React lifecycle phase has a number of lifecycle methods that you can override to run code at specified times during the process.
  - These are popularly known as component lifecycle methods.

- Initialisation phase:
  - Constructor(props): This is a special function that is called when new components are created. In this, we initialize the state or props of the component.

Lifecycle methods

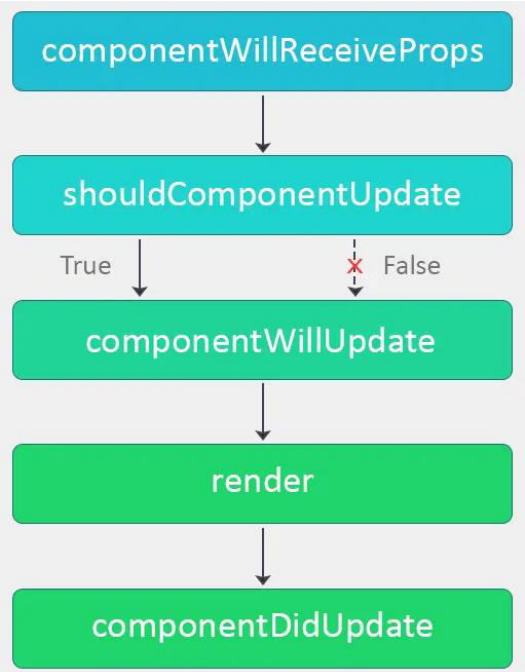
- The mounting phase refers to the phase during which a component is created and inserted to the DOM.
- The following methods are called in order.
  - **ComponentWillMount**(): This function is called immediately before mounting occurs. It is called right before the first rendering is performed.
  - **render**(): You have this method for all the components created. It returns the Html node.
  - **componentDidMount**(): This method is called right after the react component is mounted on DOM or you can say that it is called right after render method executed for the first time
  - Here we can make API call, foreg, interact with dom or perform any ajax call to load data.



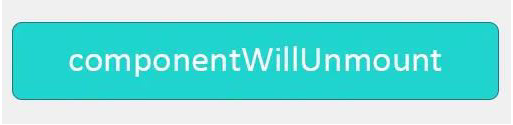

```
class HelloComponent extends Component{
    componentDidMount(){
        console.log("Mounted component")
    }
    render(){
        return <h1>Hello</h1>
    }
}
```

Lifecycle methods

- Update: In this state, the dom is interacted by a user and updated. For example, you enter something in the textbox, so the state properties are updated.
  - The component is re-rendered whenever a change is made to react component's state or props, you can simply say that the component is updated.
- Following are the methods available in update state:
  - shouldComponentUpdate() : called when the component is updated.
  - componentDidUpdate() : after the component is updated.



- UnMounting: this state comes into the picture when the Component is not required or removed.
- Following are the methods available in unmount state:
  - ComponentWillUnmount(): called when the Component is removed or destroyed.



```
import React, {Component} from 'react';

class LifecycleComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {name: ''};

    this.UpdateName = this.UpdateName.bind(this);
    this.testclick = this.testclick.bind(this);
  }

  UpdateName(event) {
    this.setState({name: event.target.value});
  }

  testclick(event) {
    alert("The name entered is: "+ this.state.name);
  }

  componentDidMount() {
    console.log('Mounting State : calling method componentDidMount');
  }

  shouldComponentUpdate() {
    console.log('Update  State : calling method shouldComponentUpdate');
    return true;
  }
```

```
componentDidUpdate() {
    console.log('Update  State : calling method
componentDidUpdate')
  }
  componentWillUnmount() {
    console.log('UnMounting State : calling method
componentWillUnmount');
  }

  render() {
    return (
      <div>
         Enter Your Name:<input type="text"
value={this.state.name} onChange={this.UpdateName} /><br/>
        <h2>{this.state.name}</h2>
        <input type="button"
          value="Click Here"
          onClick={this.testclick} />
      </div>
    );
  }
}

export default LifecycleComponent;
```


```
Mounting State : calling method componentDidMount
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
Update  State : calling method shouldComponentUpdate
Update  State : calling method componentDidUpdate
```

[Lifecycle Methods (kirupa.com)]