

REACTJS – MODULE 3

COMPONENT



AGENDA

- Introduction to Component
- Props
- State
- Difference between props and state
- Component life cycle
- Types of component
- Demo
- Activity



Introduction to component

React allows us to create independent, reusable, maintainable and testable smaller unit known as “Component”

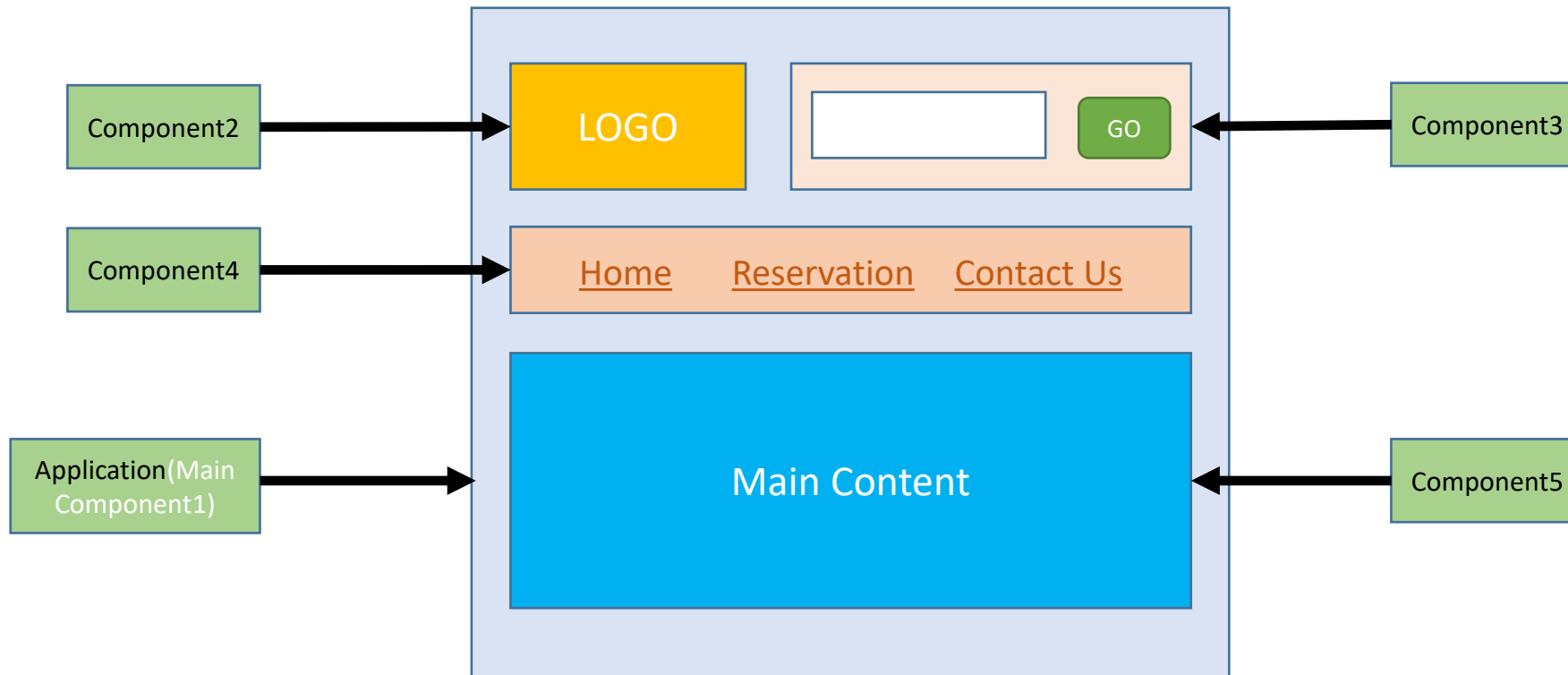
React applications are nothing but collection of components

Component represent the view/ UI of an application

Components are considered as main building block in ReactJS

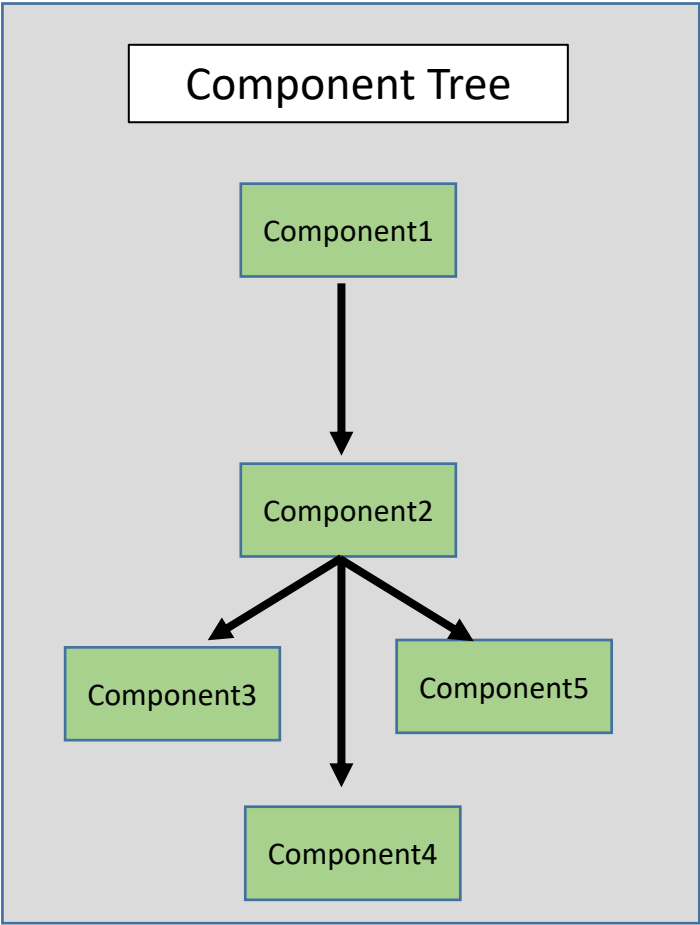
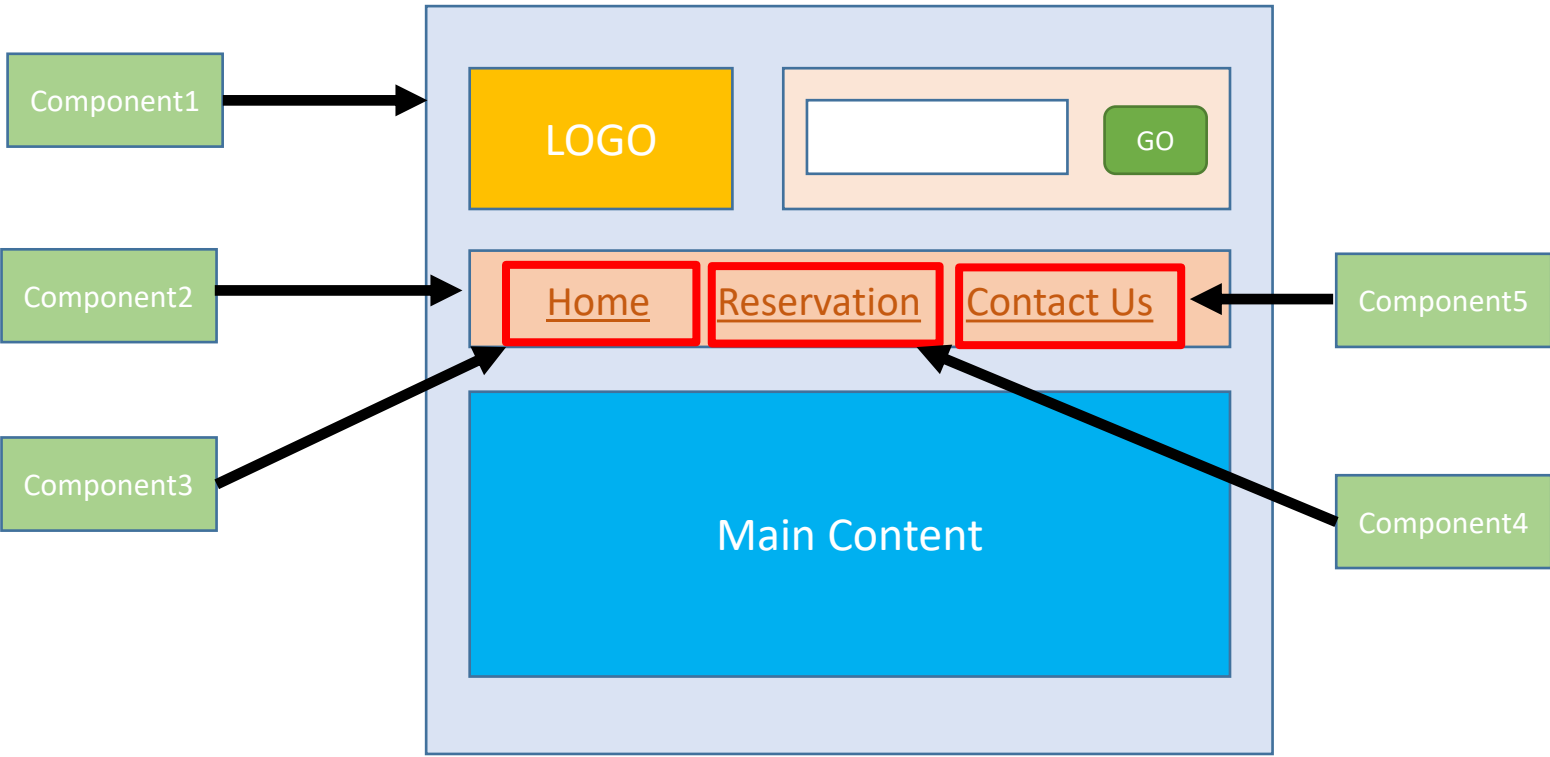
Introduction to component

The complete application when broken down into smaller parts(component) it becomes easy to maintain. This way react application can be modularized



Component Tree

When the component is rendered(displayed) inside other component, this will form component tree



How To Create component

Create Hello component(Hello.js)

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

class Hello extends Component{

  render() {
    return(
      <div>
        <h3>Hello Component</h3>
      </div>
    )
  }
}

export default Hello;
```

How To Create component

Create Hello component(Hello.js)

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

Import the definition of component from react library

```
class Hello extends Component{
```

Hello is a custom component extended from Component

```
  render() {  
    return(  
      <div>  
        <h3>Hello Component</h3>  
      </div>  
    )  
  }  
}
```

Render is a method which can return a single element(here it is <div> element)

```
export default Hello;
```

Export Hello component so that it can be imported when required

How To Create component

App component(App.js)

```
import React, { Component } from 'react';
import Hello from './Hello';

class App extends Component {
  render() {
    return (
      <div>
        <h2>App Component</h2>
        <Hello/>
      </div>
    );
  }
}

export default App;
```


How To Create component

App component(App.js)

```
import React, { Component } from 'react';  
import Hello from './Hello';
```

Import Hello component

```
class App extends Component {  
  render() {  
    return (  
      <div>  
        <h2>App Component</h2>  
        <Hello/>  
      </div>  
    );  
  }  
}
```

The UI of Hello component is captured

```
export default App;
```

How To Create component

Index component(index.js)

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

ReactDOM.render(<App />, document.getElementById('root'));
registerServiceWorker();
```

How To Create component

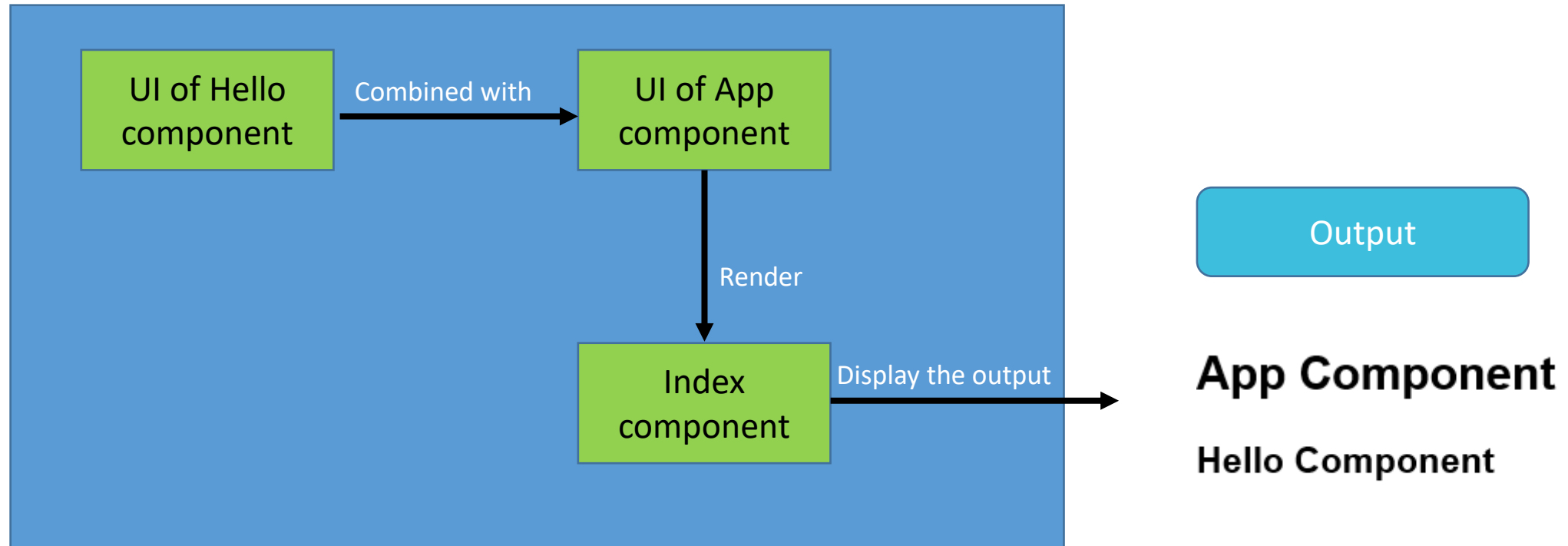
Index component(index.js)

```
import React from 'react';  
import ReactDOM from 'react-dom';  
import './index.css';  
import App from './App';  
import registerServiceWorker from './registerServiceWorker';  
  
ReactDOM.render(<App />, document.getElementById('root'));  
registerServiceWorker();
```

This is required to use DOM specific methods

App component is rendered inside root element

How To Create component



Props in Component

Props are considered as arguments/parameter to a component

Using props we can also customize the component

Components can be reused by passing different Props

Props are defined by parent component, once defined with the value it cannot be changed(Props are read-only).

Props in Component

App.js

```
import React, { Component } from 'react';
import Hello from './Hello';

class App extends Component {
  render() {
    return (
      <div>
        <h2>App Component</h2>
        <Hello name="Accenture"/>
      </div>
    );
  }
}

export default App;
```

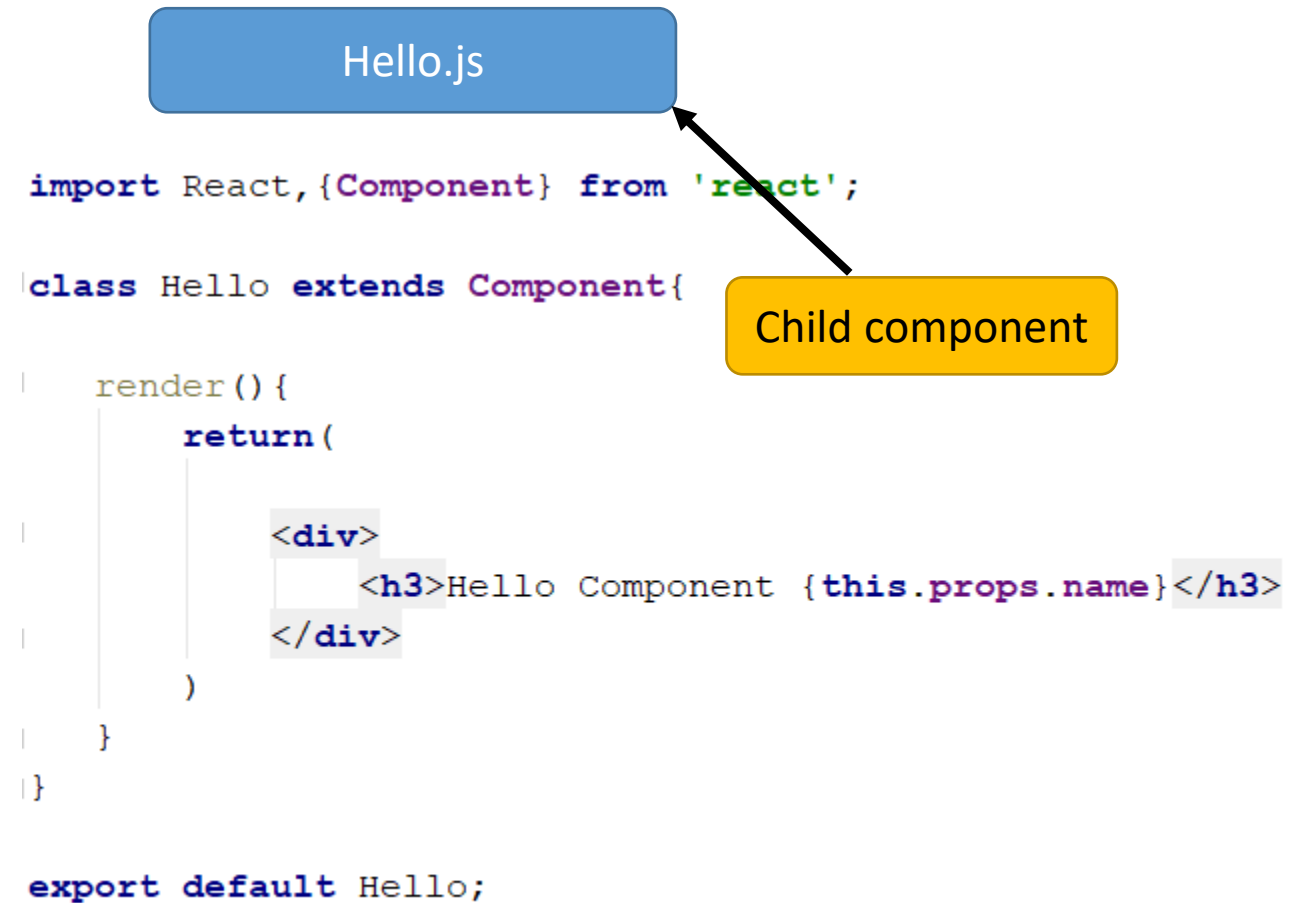
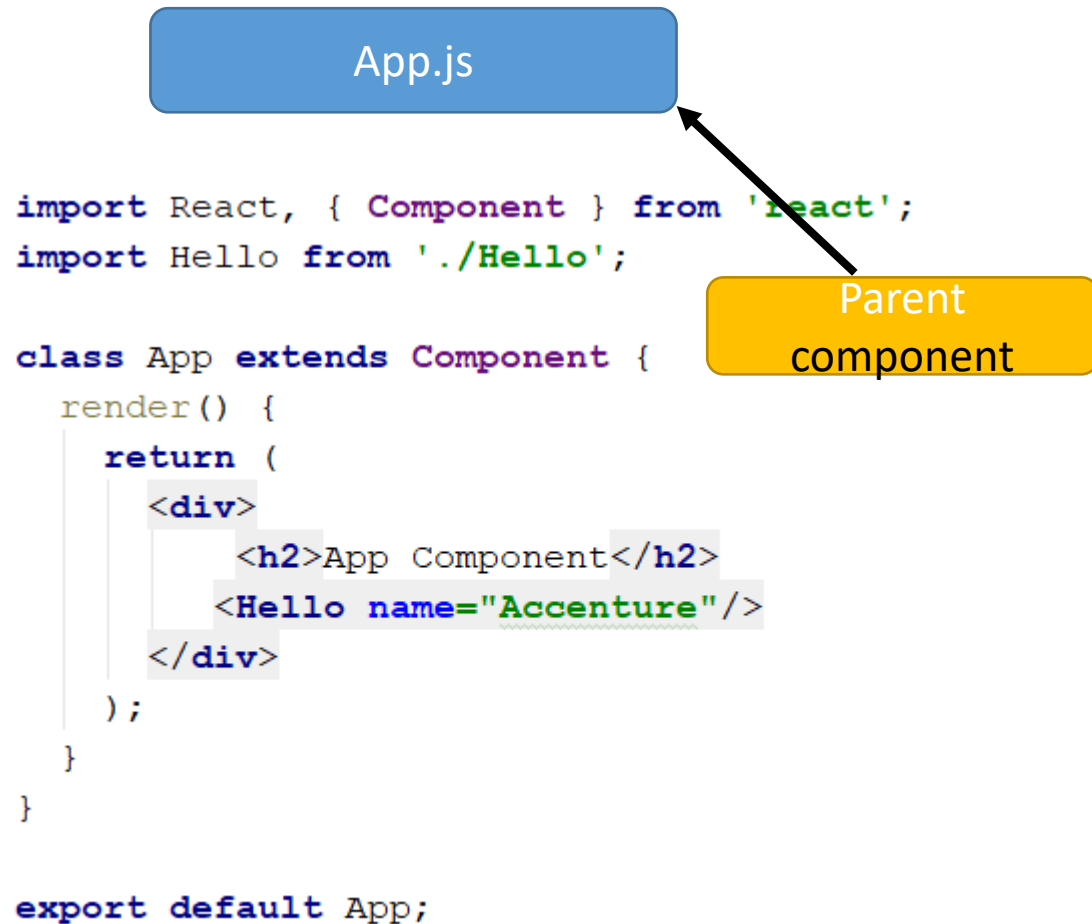
Hello.js

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

class Hello extends Component {
  render() {
    return (
      <div>
        <h3>Hello Component {this.props.name}</h3>
      </div>
    );
  }
}

export default Hello;
```

Props in Component



Props in Component

App.js

```
import React, { Component } from 'react';
import Hello from './Hello';

class App extends Component {
  render() {
    return (
      <div>
        <h2>App Component</h2>
        <Hello name="Accenture"/>
      </div>
    );
  }
}

export default App;
```

Hello.js

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

class Hello extends Component{
  // ...
  render() {
    return (
      <h3>Hello Component</h3>
    );
  }
}

export default Hello;
```

Here name is a parameter it is passed to Hello component with the value "Accenture"

`{this.props.name}</h3>`

Props(properties) are accessed using `this.props.propname`

Props in Component

Output

App Component

Hello Component Accenture

State in Component

States are considered as changing data of a component

States can be initialized only inside the constructor of a component

Using `setState` one can modify the state of a component

State in Component

Hello.js

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

class Hello extends Component{

  constructor() {
    super();
    this.state={
      noOfEmps:2000
    }
  }

  render() {
    return(

      <div>
        <h3>Hello Component {this.props.name}</h3>
        <h3>{this.props.name} has {this.state.noOfEmps} Employees</h3>
      </div>
    )
  }
}

export default Hello;
```

State in Component

Hello.js

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

class Hello extends Component{

  constructor() {
    super();
    this.state={
      noOfEmps:2000
    }
  }

  render() {
    return(
      <div>
        <h3>Hello Component {this.props.name}</h3>
        <h3>{this.props.name} has {this.state.noOfEmps} Employees</h3>
      </div>
    )
  }
}

export default Hello;
```

State of a component is initialized inside constructor. Here "noOfEmps" is the state of Hello component

This is the way to access state of a component

State in Component

Output

App Component

Hello Component Accenture

Accenture has 2000 Employees

State in Component – Modify State

Hello.js

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

class Hello extends Component {

  constructor() {
    super();
    this.state = {
      noOfEmps: 2000
    }
  }

  modifyState() {
    this.setState({
      noOfEmps: 3000
    })
  }

  render() {
    return (
      <div>
        <h3>Hello Component {this.props.name}</h3>
        <h3>{this.props.name} has {this.state.noOfEmps} Employees</h3>
        <button onClick={() => this.modifyState()}>Click to Modify State</button>
      </div>
    )
  }
}

export default Hello;
```

State in Component – Modify State

Hello.js

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

class Hello extends Component {

  constructor() {
    super();
    this.state = {
      noOfEmps: 2000
    }
  }

  modifyState() {
    this.setState({
      noOfEmps: 3000
    })
  }

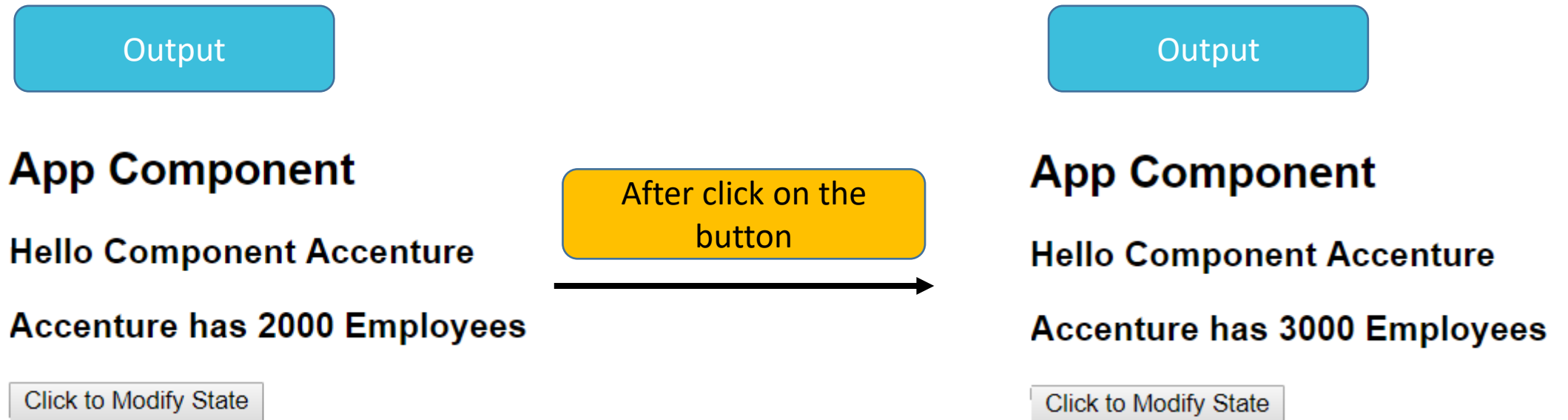
  render() {
    return (
      <div>
        <h3>Hello Component {this.props.name}</h3>
        <h3>{this.props.name} has {this.state.noOfEmps} Employees</h3>
        <button onClick={() => this.modifyState()}>Click to Modify State</button>
      </div>
    )
  }
}

export default Hello;
```

useState is used to modify the state of a component.

When the button is clicked modifyState method is called which will change the state

State in Component – Modify State



Difference between Props and State

Props

Passed as a parameter from parent component to child component

Fixed value, once defined cannot be changed

Accessed using
`{this.props.propname}`

State

Defined inside a component. Defines the state of the component

Data(State) can be changed using `setState`

Accessed using
`{this.state.statename}`

Component Life Cycle

Every component in react will have life cycle events which are executed based on the requirement

These events are basically categorized into three categories and each category will have related events

Mounting/Initialization

Events

getInitialState()
getDefaultProps()
componentWillMount()
render()
componentDidMount()

Updating

Events

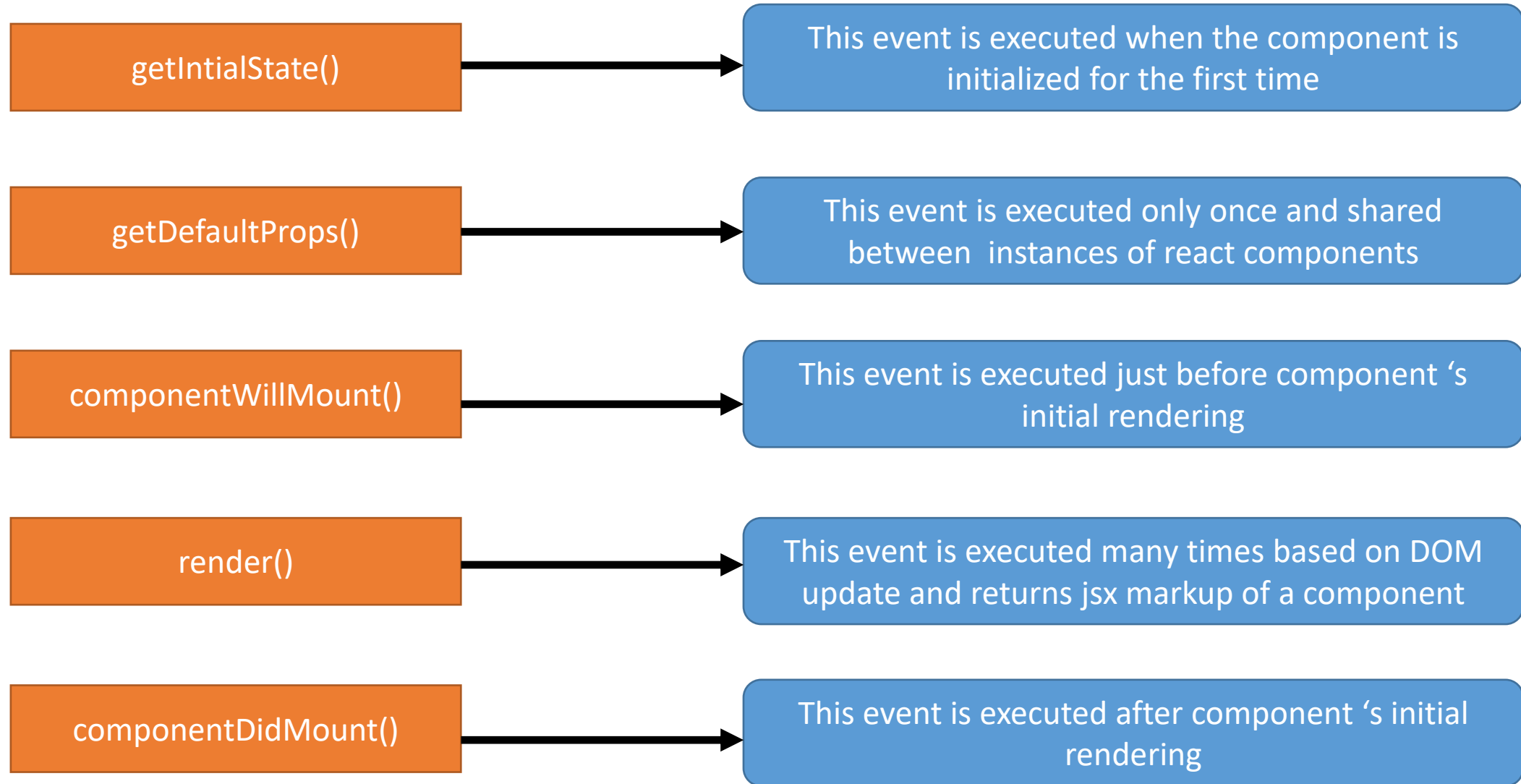
componentWillReceiveProps()
shouldComponentUpdate()
componentWillUpdate()
render()
componentDidUpdate()

Unmounting

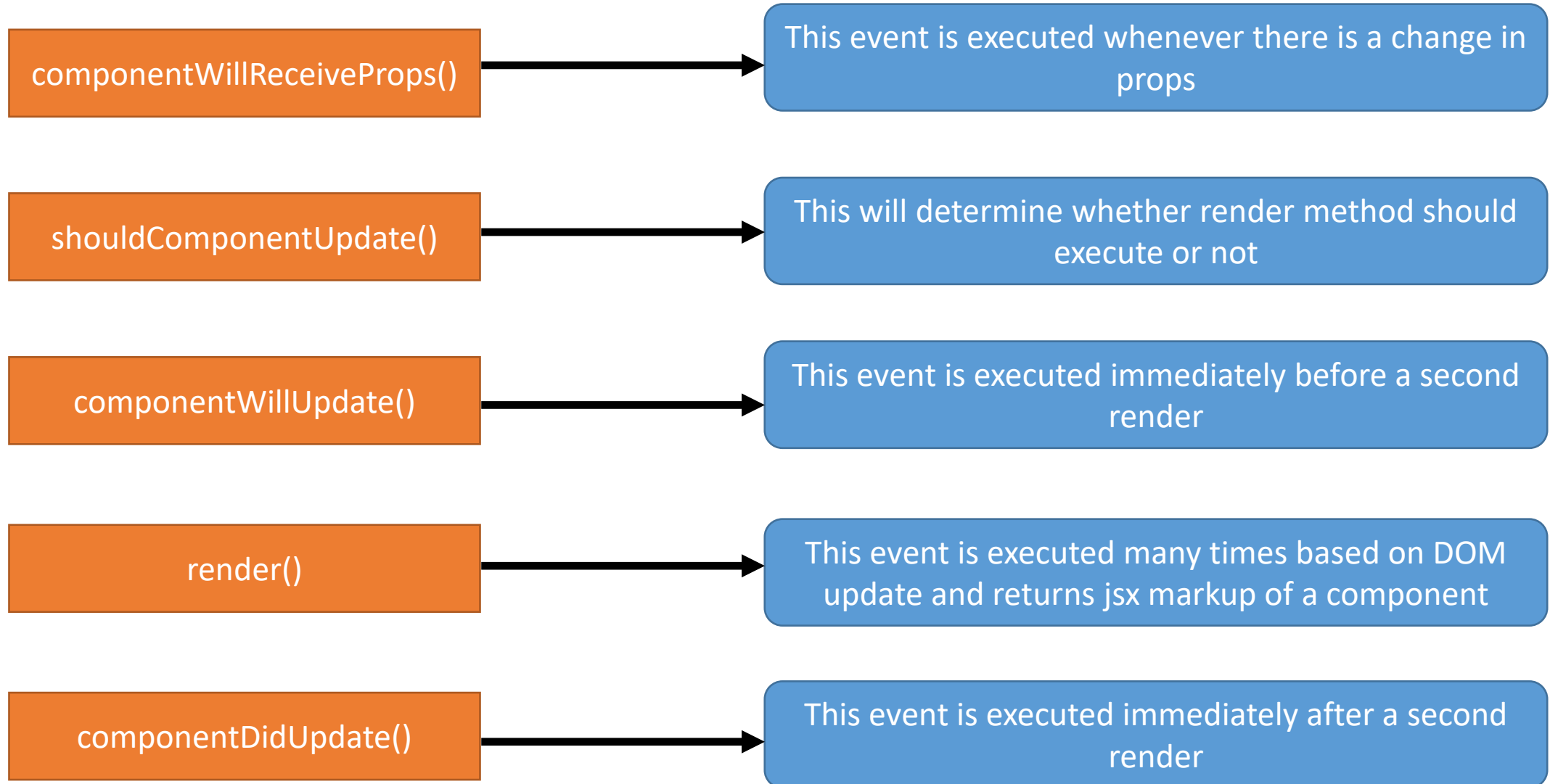
Events

componentWillUnmount()

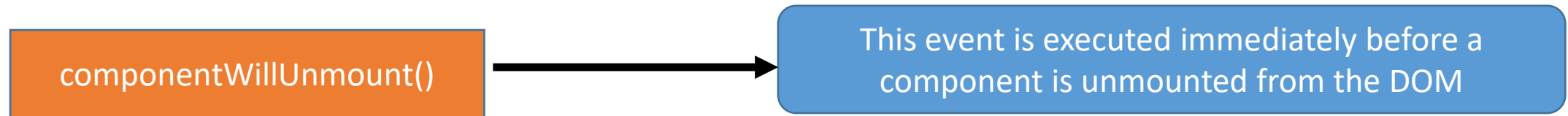
Component Life Cycle – Mounting/Initialization



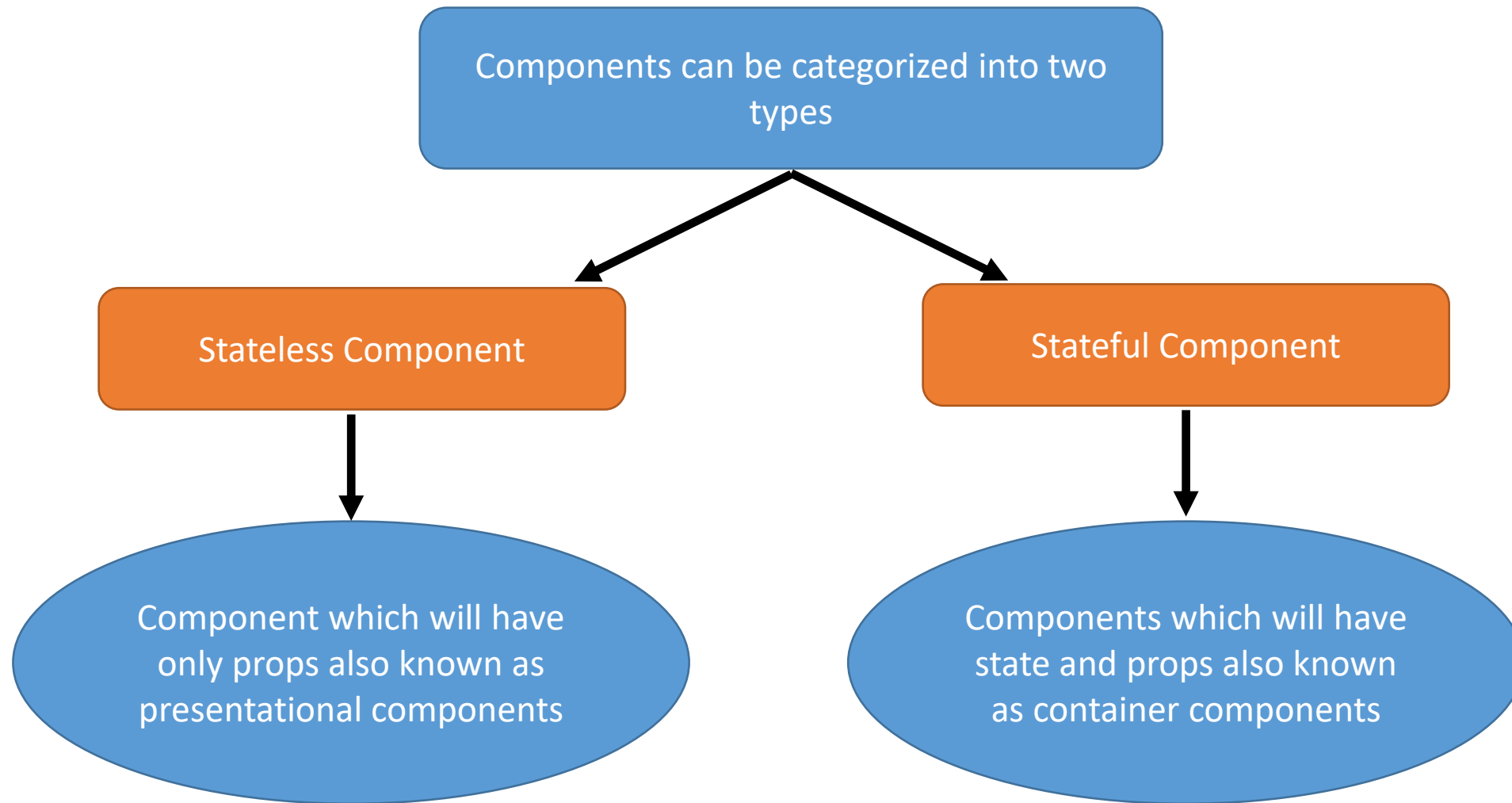
Component Life Cycle – Updating



Component Life Cycle – Unmounting



Component Types



Activity

1. Create Student component
2. Display roll number and first name of a student using props of a component
3. Display marks of a student using state of a component
4. Create a button which should increase the marks of a student by 5
5. Stop the increment of a marks, once the marks reach to 100 and also show the alert with the message “student marks cannot be more than 100”

Expected Activity output

Initial view

Student details are Roll Number =101 and Name =Mark

Click to increase Marks by 5

Student Marks 35

After multiple clicks on
increase marks button

Student details are Roll Number =101 and Name =Mark

Click to increase Marks by 5

Student Marks 100

Display error

localhost:3000 says:

Student marks cannot be more than 100

OK

MODULE SUMMARY

- What is component
- Role of component in react application
- How components created
- Props in component
- States in component



THANK YOU

