SNS COLLEGE OF ENGINEERING

An Autonomous Institution

Coimbatore-107

19TS601-FULL STACK DEVELOPMENT

UNIT-2

REACT

4. Class components

React Class Components

- Before React 16.8, Class components were the only way to track state and lifecycle on a React component. Function components were considered "state-less".
- With the addition of Hooks, Function components are now almost equivalent to Class components. The differences are so minor that you will probably never need to use a Class component in React.

- Even though Function components are preferred, there are no current plans on removing Class components from React.

React Components

- Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function.
- Components come in two types, Class components and Function components, in this chapter you will learn about Class components.

Create a Class Component

- When creating a React component, the component's name must start with an upper case letter.
- The component has to include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.
- The component also requires a render() method, this method returns HTML.

Example

Create a Class component called `Car`

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

- Now your React application has a component called Car, which returns a <h2> element.
- To use this component in your application, use similar syntax as normal HTML: <Car />

- Display the Car component in the "root" element:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}

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

Hi, I am a Car!

Component Constructor

- If there is a constructor() function in your component, this function will be called when the component gets initiated.
- The constructor function is where you initiate the component's properties.
- In React, component properties should be kept in an object called state.

- The constructor function is also where you honor the inheritance of the parent component by including the super() statement, which executes the parent component's constructor function, and your component has access to all the functions of the parent component (React.Component).

Example

Create a constructor function in the Car component, and add a color property:

```
class Car extends React.Component {
  constructor() {
    super();
    this.state = {color: "red"};
  }
  render() {
    return <h2>I am a Car!</h2>;
  }
}
```

- Use the color property in the render() function:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  constructor() {
    super();
    this.state = {color: "red"};
  }
  render() {
    return <h2>I am a {this.state.color} Car!</h2>;
  }
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```


I am a red Car!

Props

- Another way of handling component properties is by using props.
- Props are like function arguments, and you send them into the component as attributes.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  render() {
    return <h2>I am a {this.props.color} Car!</h2>;
  }
}

ReactDOM.render(<Car color="red"/>, document.getElementById('root'));
```

I am a red Car!

Props in the Constructor

- If your component has a constructor function, the props should always be passed to the constructor and also to the React.Component via the super() method.

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  constructor(props) {
    super(props);
  }
  render() {
    return <h2>I am a {this.props.model}!</h2>;
  }
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car model="Mustang"/>);
```

I am a Mustang!

Components in Components

- Components inside other components:

Example

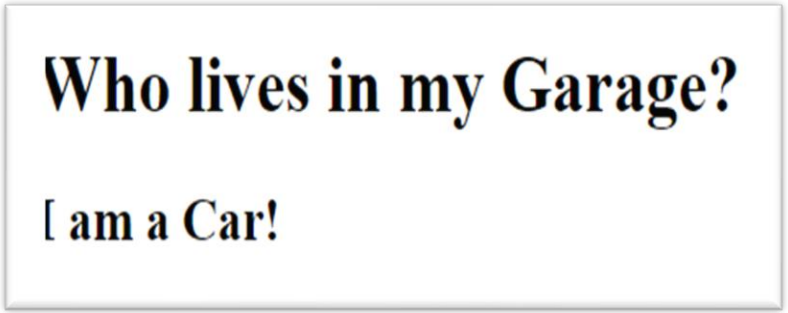
- Use the Car component inside the Garage component:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

class Car extends React.Component {
  render() {
    return <h2>I am a Car!</h2>;
  }
}

class Garage extends React.Component {
  render() {
    return (
      <div>
      <h1>Who lives in my Garage?</h1>
      <Car />
      </div>
    );
  }
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```



Components in Files

- React is all about re-using code, and it can be smart to insert some of your components in separate files.
- To do that, create a new file with a .js file extension and put the code inside it:
- Note that the file must start by importing React (as before), and it has to end with the statement export default Car;.

Example

This is the new file, we named it `Car.js` :

```
import React from 'react';

class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}

export default Car;
```

- To be able to use the Car component, you have to import the file in your application.

Example

- Now we import the Car.js file in the application, and we can use the Car component as if it was created here.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Car from './Car.js';

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

Hi, I am a Car!

Thank You