SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NAAC-UGC with 'A' Grade

Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY

Course Code and Name : 19TS601 FULL STACK DEVELOPMENT

Unit 2 : REACT

Topic : Dynamic Composition

What is dynamic composition?

- Dynamic composition refers to the process of creating art or graphical designs where elements are actively arranged to maintain balance and interest.
- This technique often employs contrast, movement, and varying focal points to capture attention and convey a message effectively

What is dynamic composition in React.js?

- Composition is a fundamental concept in React that allows developers to build complex and dynamic user interfaces with ease.
- By breaking down the UI into smaller, reusable components and composing them together, we can create efficient, scalable, and maintainable codebases

1. **Component Composition**
    - Combining multiple components dynamically to build a UI.
    - Example: Passing different child components as props.
2. **Higher-Order Components (HOCs)**
    - A function that takes a component and returns an enhanced version.
    - Useful for **reusability** and **code abstraction**.
3. **Render Props Pattern**
    - Passing a function as a prop to control rendering dynamically.
4. **Dynamic Import (Lazy Loading)**
    - Loading components dynamically using React.lazy().
5. **Context API for Dynamic Composition**
    - Managing shared state across multiple dynamic components.

Basic Dynamic Component Composition

```
const Card = ({ children }) => {
return ( <div className="p-4 border rounded-lg shadow-md">
{children}
 </div> );
 };
const App = () => {
return (
<div className="flex flex-col gap-4">
 <Card>
<h2 className="text-lg font-bold">Title 1</h2> <p>Content for the first
card.</p>
</Card>
 <Card>
<h2 className="text-lg font-bold">Title 2</h2> <button className="bg-
blue-500 text-white px-4 py-2 rounded">Click Me</button>
</Card>
 </div> );
 };
 export default App;
```

- Here, <Card> is dynamically composed with different children elements

Using Props for Dynamic Rendering

```
const Button = ({ type }) => {
 const styles = type === "primary" ? "bg-blue-500" : "bg-gray-500"; return
<button className={
`${styles} text-white px-4 py-2 rounded`}>Click Me</button>; };
const App = () => {
 return ( <div> <Button type="primary" /> <Button type="secondary" />
</div> );
 };
```

- export default App;
- **Here, the <Button> component dynamically changes based on the type prop.**

React State

React components has a built-in state object.
The state object is where you store property values that belong to the component.
When the state object changes, the component re-renders.

Creating the `state` Object

- The state object is initialized in the constructor

Specify the state object in the constructor method:

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {brand: "Ford"};
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    );
  }
}
```

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>    );
  }}
```

Using the `state` Object

The state object anywhere in the component by using the this.state.*propertyname* syntax:

```
import React from 'react';
import ReactDOM from 'react-dom/client';
 class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
 render() {
   return (
     <div>
       <h1>My {this.state.brand}</h1>
       <p>
         It is a {this.state.color}
         {this.state.model}
         from {this.state.year}.
       </p>
     </div>
   );
  }
}

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

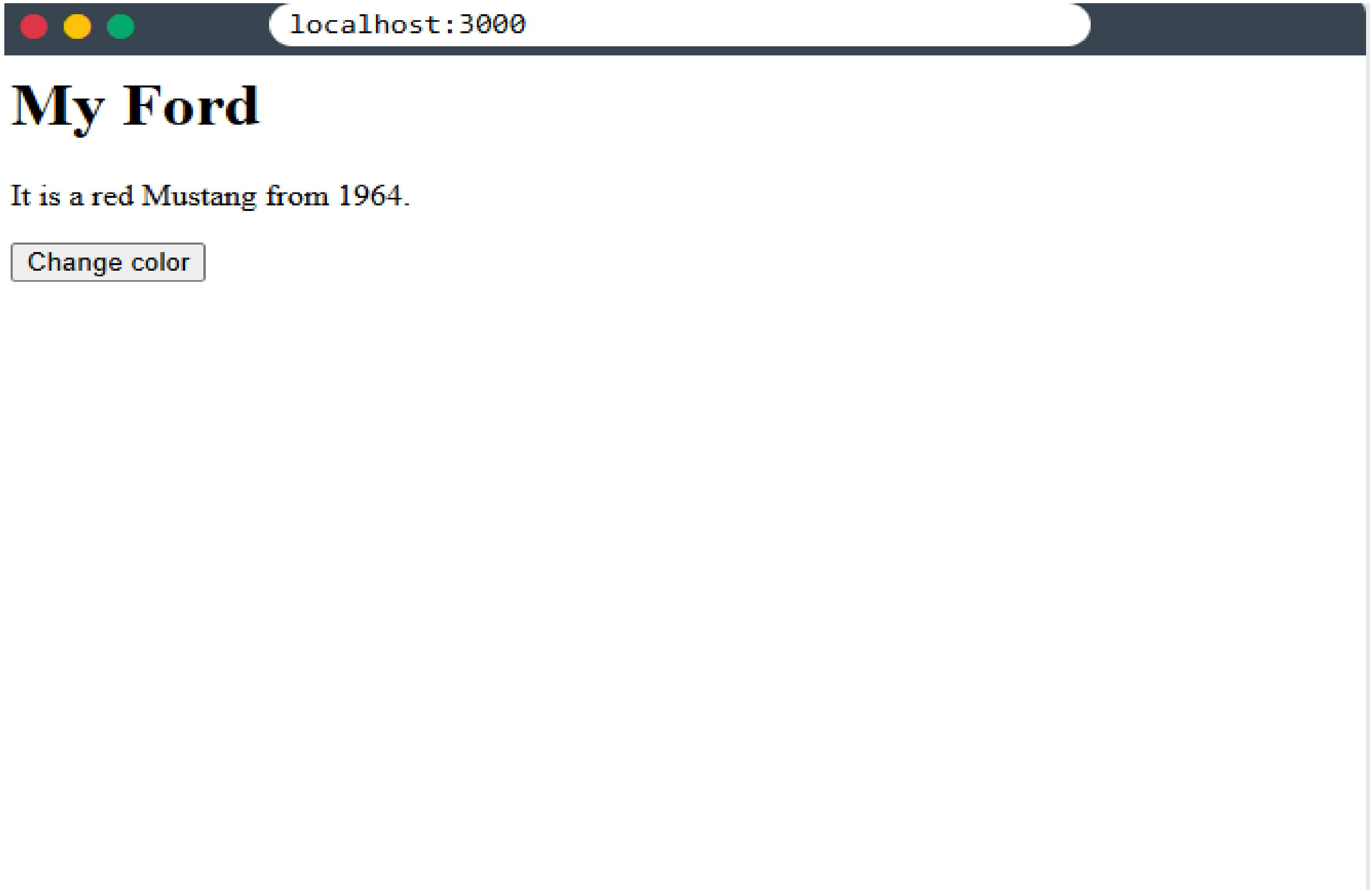
Output:

```
My Ford
It is a red Mustang from 1964.
```

Changing the state Object

- To change a value in the state object, use the this.setState() method.
- When a value in the state object changes, the component will re-render, meaning that the output will change according to the new value(s).
- Add a button with an onClick event that will change the color property

```
import React from 'react';
import ReactDOM from 'react-dom/client';
 class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Ford",
      model: "Mustang",
      color: "red",
      year: 1964
    };
  }
  changeColor = () => {
    this.setState({color: "blue"});
  }
render() {
  return (
    <div>
      <h1>My {this.state.brand}</h1>
      <p>
        It is a {this.state.color}
        {this.state.model}
        from {this.state.year}.
      </p>
      <button
        type="button"
        onClick={this.changeColor}
      >Change color</button>
    </div>
  );
 }
}
```



localhost:3000

My Ford

It is a red Mustang from 1964.

Always use the setState() method to change the state object, it will ensure that the component knows its been updated and calls the render() method (and all the other lifecycle methods).

1.What is dynamic composition?

Text Book:

1.Pro MERN Stack, Full Stack Web App Development with Mongo, Express, React, and Node, Vasan Subramanian, A Press Publisher, 2019.

Reference:

David Flanagan, “Java Script: The Definitive Guide”, O’Reilly Media, Inc, 7 th Edition, 2020

2. Matt Frisbie, “Professional JavaScript for Web Developers” Wiley Publishing, Inc, 4th Edition, ISBN: 978-1-119-36656-0, 2019

Thank You!
dreamstime