Hema Coding School

YouTube Link: https://www.youtube.com/@HemaCodingSchool

Introduction

In this example, we create a reusable React component that displays an image along with a name. This component is styled using inline CSS and is rendered multiple times within a parent component.

Key Concepts Covered

1. **Creating Functional Components**
2. **Using Props**
3. **Styling Components**
4. **Rendering Multiple Instances**

Detailed Explanation

1. **Creating Functional Components:**
   - A functional component in React is simply a JavaScript function that returns JSX.

```
      let CreateImg = (props) => {
  // Component logic and JSX here
};
```

Using Props:

- Props (short for properties) allow you to pass data from a parent component to a child component.

```
    let CreateImg = (props) => {
  return (
    <div>
      <div>{props.name}</div>
      <img src={props.img} alt="TATA" width="300px" height="300px" />
    </div>
  );
};
```

In this example, props.name and props.img are used to dynamically set the content of the component.

Styling Components:

- Inline styles in React are specified as objects.

```

let imageStyle = {
  color: 'red',
  padding: '20px',
  boxShadow: '0px 0px 10px blue'
};
```

This style object is applied to the component using the style attribute.

Rendering Multiple Instances:

- You can reuse the CreateImg component multiple times within a parent component.

```
 let App = <div style={{display:'flex', justifyContent:'space-evenly' }}>
<CreateImg name= "TATA"
img="https://assets.gqindia.com/photos/645e034efc79052643f24e8e/16:9/w_1920,c_limit/Ratan-Tata.jpg"/>
<CreateImg name= "TATA"
img="https://assets.gqindia.com/photos/645e034efc79052643f24e8e/16:9/w_1920,c_limit/Ratan-Tata.jpg"/>
<CreateImg name= "TATA"
img="https://assets.gqindia.com/photos/645e034efc79052643f24e8e/16:9/w_1920,c_limit/Ratan-Tata.jpg"/>
         </div>
```

Each CreateImg component is passed different props for name and img.

Rules of JSX

1. **JSX Must Have One Parent Element:**
   - JSX expressions must have one parent element. If you need to return multiple elements, wrap them in a single parent element like a div or a React Fragment.

```
return (
  <div>
    <h1>Hello</h1>
    <p>World</p>
  </div>
);
```

JSX Tags Must Be Closed:

- All JSX tags must be properly closed, either with a closing tag or self-closing if they don't have children.

```
<img src="image.jpg" alt="description" />
```

JSX Attributes Use CamelCase:

- In JSX, attributes are written in camelCase instead of lowercase.

```
<div className="container"></div>
<input defaultValue="text" />
```

JavaScript Expressions in JSX:

- You can use JavaScript expressions inside JSX by enclosing them in curly braces {}.

```
const name = "Mahesh";
return <h1>Hello, {name}</h1>;
```

CSS in JSX:

- Inline styles in JSX are written as objects with camelCased properties.

```
const style = { color: 'blue', fontSize: '14px' };
return <h1 style={style}>Hello, World!</h1>;
```

Comments in JSX:

- Comments in JSX are written inside curly braces.

```
return (
  <div>
    {/* This is a comment */}
    <h1>Hello, World!</h1>
  </div>
);
```

Rules of Components

1. **Component Names Must Be Capitalized:**

React components must start with a capital letter. This helps React distinguish between custom components and HTML elements.

Components Must Return JSX:

- Functional components must return a JSX element or null.

```
let MyComponent =()=> {
  return <h1>Hello, World!</h1>;
}
```

Props in Components:

- Props are read-only properties passed from a parent component to a child component. They allow customization and data flow between components.

```
let Greeting = (props)=> {
  return <h1>Hello, {props.name}</h1>;
}
```

Interview Question:

What is JSX?

- **Answer:** JSX stands for JavaScript XML. It is a syntax extension for JavaScript that allows you to write HTML-like code within JavaScript. JSX makes it easier to create React elements because it is more intuitive and easier to read than traditional JavaScript. Under the hood, JSX is transformed into React.createElement() calls, which return plain JavaScript objects representing virtual DOM elements.

How do you create a functional component in React?

- **Answer:** A functional component in React is a JavaScript function that returns JSX. Here's an

```
let Greeting =()=> {
  return <h1>Hello, World!</h1>;
}
```

Functional components can receive props as an argument and return JSX to describe what should appear on the screen.

What are props in React?

- **Answer:** Props (short for properties) are read-only attributes that are passed from a parent component to a child component. They allow for customization and configuration of child components. Props are used to pass data and event handlers down to child components.

What is the difference between a functional component and a class component in React?

- **Answer:** Functional components are simpler and are defined as plain JavaScript functions. They can use hooks to manage state and side effects. Class components are ES6 classes that extend React.Component and have access to lifecycle methods and state.

```
// Functional component
let Greeting = ()=> {
  return <h1>Hello, World!</h1>;
}

// Class component
class Greeting extends React.Component {
  render() {
    return <h1>Hello, World!</h1>;
  }
}
```

How do you apply inline styles in JSX?

- **Answer:** Inline styles in JSX are specified as objects with camelCased properties. Here's an example:

```
const style = { color: 'blue', fontSize: '14px' };
let StyledComponent =()=> {
  return <h1 style={style}>Styled Text</h1>;
}
```

What are React Fragments and why are they useful?

- **Answer:** React Fragments allow you to group multiple elements without adding extra nodes to the DOM. This is useful for returning multiple elements from a component without introducing unnecessary wrapper elements.

```
let FragmentExample =()=> {
  return (
    <>
      <h1>Title</h1>
      <p>Description</p>
    </>
  );
}
```