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>
    </>
  );
}
```

Notes on Virtual DOM

Purpose: The Virtual DOM (VDOM) is a concept used in React and other modern front-end frameworks to improve the efficiency of updating the view in response to changes in application state. It is an abstraction of the actual DOM and provides a lightweight copy that can be updated and manipulated more efficiently.

Key Concepts:

1. **Virtual DOM (VDOM):**
   - A virtual representation of the actual DOM.
   - Exists entirely in memory.
   - Used to optimize and manage updates to the real DOM.
2. **Real DOM:**
   - The actual Document Object Model that represents the structure of a web document.

- Manipulating the real DOM can be slow and inefficient, especially for complex applications with frequent updates.

3. **Reconciliation:**
   - The process of comparing the current VDOM with the previous VDOM to determine the minimal set of changes needed to update the real DOM.
   - React uses a diffing algorithm to identify these changes and apply them efficiently.

How the Virtual DOM Works:

1. **Rendering:**
   - React components render to create a VDOM tree.
   - The initial render creates the entire VDOM and the real DOM.
2. **Updating:**
   - When the state of a component changes, React creates a new VDOM tree.
   - The new VDOM is compared to the previous VDOM to identify the differences (diffing).
   - React computes the most efficient way to update the real DOM based on these differences (reconciliation).
3. **Applying Changes:**
   - Only the changes (differences) are applied to the real DOM.
   - This minimizes the number of DOM manipulations and optimizes performance.

Advantages of the Virtual DOM:

1. **Performance:**
   - Minimizes the number of direct DOM manipulations, which are typically slow.
   - Efficiently updates only the parts of the DOM that have changed.
2. **Abstraction:**
   - Provides a simpler programming model by abstracting away the complexity of direct DOM manipulation.
   - Developers can focus on building components without worrying about DOM optimization.
3. **Consistency:**
   - Ensures a consistent and predictable rendering process.
   - Helps prevent issues related to direct DOM manipulation and side effects.

```
function timer() {
    let javascriptElement = `
    <div>
       <h2>JavaScript</h2>
       <h1>${new Date().toLocaleTimeString()}</h1>
       </div>
    `;
    document.getElementById("javascript-container").innerHTML =
     javascriptElement;
```

```
  let reactElement = React.createElement(
    "div",
    null,
    React.createElement("h2", null, "React"),
    React.
    createElement("h1", null, new Date().toLocaleTimeString())
  );

  ReactDOM.render(
    reactElement,
    document.getElementById("react-container")
  );
}
setInterval(timer, 1000);
```

Virtual DOM Interview Questions and Answers

Question 1: What is the Virtual DOM?

Answer: The Virtual DOM (VDOM) is a concept where a virtual representation of the actual DOM is kept in memory and synced with the real DOM by a library such as React. It provides a lightweight copy of the DOM that can be updated and manipulated more efficiently.

Question 2: How does the Virtual DOM improve performance in React applications?

Answer: The Virtual DOM improves performance by minimizing the number of direct manipulations to the real DOM, which are typically slow. It does this by:

1. Keeping a virtual representation of the UI.
2. Comparing the current and previous VDOM to identify differences (diffing).
3. Applying only the necessary changes to the real DOM (reconciliation), rather than re-rendering the entire UI.

Question 3: Explain the process of reconciliation in React.

Answer: Reconciliation is the process React uses to update the real DOM based on changes in the Virtual DOM. It involves:

1. Creating a new VDOM when the state or props of a component change.
2. Comparing the new VDOM with the previous VDOM to find the differences (diffing).
3. Generating a minimal set of operations to update the real DOM to match the new VDOM.
4. Applying these operations to the real DOM efficiently.

Question 4: What is the difference between the Virtual DOM and the real DOM?

Answer: The Virtual DOM is an in-memory representation of the real DOM. It is a lightweight copy that can be updated quickly without causing performance issues. The real DOM is the actual Document Object Model that represents the structure of a web document and interacts with the browser. Direct manipulation of the real DOM can be slow and inefficient, especially for complex or frequently updated UIs.

Question 6: What are the benefits of using the Virtual DOM in React?

Answer: The benefits of using the Virtual DOM in React include:

1. Improved performance: Minimizes direct manipulations of the real DOM, which are typically slow.
2. Efficient updates: Only updates the parts of the DOM that have changed, rather than re-rendering the entire UI.
3. Simplified programming model: Abstracts away the complexity of direct DOM manipulation, allowing developers to focus on building components.
4. Consistent rendering: Ensures a consistent and predictable rendering process, reducing the risk of bugs related to direct DOM manipulation.

Question 7: What is the diffing algorithm in React?

Answer: The diffing algorithm in React is a process used to compare the current VDOM with the previous VDOM to identify changes. React uses a heuristic algorithm that assumes two elements of different types will produce different trees. This algorithm is designed to be efficient and fast, allowing React to quickly identify and apply the minimal set of changes needed to update the real DOM.

Question 8: How does React handle component updates with the Virtual DOM?

Answer: When a component's state or props change, React follows these steps:

1. A new VDOM is created based on the updated state or props.
2. The new VDOM is compared with the previous VDOM to identify differences.
3. React determines the minimal set of changes needed to update the real DOM.
4. React updates the real DOM accordingly, applying only the necessary changes.

Question 9: Why might direct manipulation of the real DOM be considered inefficient?

Answer: Direct manipulation of the real DOM is considered inefficient because:

1. The real DOM operations are typically slow, especially for complex and frequently updated UIs.

2. Each DOM operation can trigger reflows and repaints in the browser, which are performance-intensive tasks.
3. Direct manipulation can lead to inconsistencies and bugs, as developers need to manually manage and optimize DOM updates.

Question 10: How does the Virtual DOM contribute to a better developer experience in React?

Answer: The Virtual DOM contributes to a better developer experience in React by:

1. Abstracting away the complexity of direct DOM manipulation, allowing developers to focus on building components.
2. Providing a declarative programming model where developers describe what the UI should look like, and React takes care of the updates.
3. Ensuring efficient and predictable updates to the UI, reducing the risk of bugs and performance issues.
4. Enabling a consistent and intuitive way to manage state and props, making it easier to reason about the application's behavior.