Introduction

In today's class, we covered the basics of creating and rendering React elements using JSX. JSX, or JavaScript XML, is a syntax extension for JavaScript that allows you to write HTML directly within React. This makes it easier to visualize the UI components you are building.

Key Concepts Covered

1. **Creating JSX Elements**
2. **Reusing Components**
3. **Rendering Components**

Detailed Explanation

1. **Creating JSX Elements:**
    - JSX allows you to create elements that look like HTML but have the full power of JavaScript.

```
let h2Jsx = (
  <>
    <h2 style={{ color: 'blue' }}>Hello React js class 01 jsx</h2>
    <div>This is JSX</div>
  </>
);
```

This snippet creates a fragment containing an h2 element with a blue color and a div element with some text.

Reusing Components:

- You can define reusable components in React using functions.

```
let reuseElement = () => {
  let h2Jsx = (
    <>
      <h2 style={{ color: 'blue' }}>Hello React js class 01 jsx</h2>
      <div>This is JSX</div>
    </>
  );
  return h2Jsx;
};
```

The reuseElement function returns a JSX fragment that can be used multiple times.

Rendering Components:

- You can use ReactDOM.render to render your React elements into the DOM.

```
let mainElement = () => {
  let createH2 = (
    <>
      <div>{reuseElement()}</div>
      <div>{reuseElement()}</div>
      <div>{reuseElement()}</div>
      <div>{reuseElement()}</div>
      <div>{reuseElement()}</div>
    </>
  );

  ReactDOM.render(createH2, document.getElementById('root'));
};
mainElement();
```

The mainElement function uses the reuseElement function to create multiple h2 elements and renders them inside the root div.

The ReactDOM.render function takes two arguments: the JSX to render and the DOM element to render it into.

Practical Application

- This approach helps in creating reusable UI components, which is a fundamental concept in React.
- Reusability improves code maintainability and readability.

Interview Questions:

1. **What is JSX?**
    - Explain JSX and how it differs from regular JavaScript.
    - Example: JSX allows you to write HTML elements in JavaScript and place them in the DOM without using functions like createElement or appendChild.
2. **What is the purpose of ReactDOM.render?**
    - Describe what ReactDOM.render does and its role in a React application.
    - Example: ReactDOM.render renders a React element into the specified DOM container and returns a reference to the component.
3. **How do you create a functional component in React?**
    - Define what a functional component is and provide an example.
    - Example: A functional component is a JavaScript function that returns JSX.
4. **How can you apply styles to a JSX element?**
    - Explain different methods to style JSX elements.
    - Example: Inline styles, CSS classes, and CSS-in-JS libraries.

5. **What are the advantages of using JSX?**
    - Discuss the benefits of using JSX in a React application.
    - Example: JSX increases readability, allows for easier debugging, and integrates with JavaScript seamlessly.
6. **What is the significance of the key prop in React?**
    - Explain why keys are important in lists of elements.
    - Example: Keys help React identify which items have changed, are added, or are removed.
7. **How can you reuse components in React?**
    - Describe different methods for creating and reusing components.
    - Example: Using functional components, higher-order components, and composition.
8. **What is a React Fragment and when would you use it?**
    - Explain the purpose of React Fragments and provide an example.
    - Example: React Fragments allow you to group a list of children without adding extra nodes to the DOM.

9. Write a simple React component using JSX that displays "Hello, World!"

- Provide a basic example of a functional component.

```
function HelloWorld() {
  return <h1>Hello, World!</h1>;
}
```