SNS COLLEGE OF ENGINEERING

An Autonomous Institution

Coimbatore-107

19TS601-FULL STACK DEVELOPMENT

UNIT-2

REACT

2. React Render HTML - React JSX

React Render HTML

- React renders HTML to the web page by using a function called ReactDOM.render().
- The ReactDOM.render() function takes two arguments, HTML code and an HTML element.
- The purpose of the function is to display the specified HTML code inside the specified HTML element.

But render where?

- There is another folder in the root directory of your React project, named "public". In this folder, there is an index.html file.
- You'll notice a single in the body of this file. This is where our React application will be rendered.

Example

Display a paragraph inside an element with the id of "root":

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

The result is displayed in the `<div id="root">` element:

```
<body>
  <div id="root"></div>
</body>
```

The HTML Code

- The HTML code in this tutorial uses JSX which allows you to write HTML tags inside the JavaScript code:

Create a variable that contains HTML code and display it in the "root" node:

```
const myelement = (
  <table>
    <tr>
      <th>Name</th>
    </tr>
    <tr>
      <td>John</td>
    </tr>
    <tr>
      <td>Elsa</td>
    </tr>
  </table>
);

ReactDOM.render(myelement, document.getElementById('root'));
```

The Root Node

- The root node is the HTML element where you want to display the result.
- It is like a container for content managed by React.

Example

The root node can be called whatever you like:

```
<body>

  <header id="sandy"></header>

</body>
```

Display the result in the `<header id="sandy">` element:

```
ReactDOM.render(<p>Hallo</p>, document.getElementById('sandy'));
```

React JSX

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.
- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX converts HTML tags into react elements.
- JSX is an extension of the JavaScript language based on ES6, and is translated into regular JavaScript at runtime.

Example 1

JSX:

```
const myElement = <h1>I Love JSX!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Example 2

Without JSX:

```
const myElement = React.createElement('h1', {}, 'I do not use JSX!');

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Expressions in JSX

- With JSX you can write expressions inside curly braces { }.
- The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result

Example

Execute the expression `5 + 5` :

```
const myElement = <h1>React is {5 + 5} times better with JSX</h1>;
```

One Top Level Element

- The HTML code must be wrapped in ONE top level element.
- So if you like to write two paragraphs, you must put them inside a parent element, like a div element.

Example

Wrap two paragraphs inside one DIV element:

```
const myElement = (
  <div>
    <p>I am a paragraph.</p>
    <p>I am a paragraph too.</p>
  </div>
);
```

Inserting a Large Block of HTML:

To write HTML on multiple lines, put the HTML inside parentheses:

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

const myElement = (
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

- Apples
- Bananas
- Cherries

One Top Level Element

- The HTML code must be wrapped in ONE top level element.
- So if you like to write two paragraphs, you must put them inside a parent element, like a div element.

```jsx
import React from 'react';
import ReactDOM from 'react-do/client';

const myElement = (
  <div>
    <h1>I am a Header.</h1>
    <h1>I am a Header too.</h1>
  </div>
);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

I am a Header.

I am a Header too.

- JSX will throw an error if the HTML is not correct, or if the HTML misses a parent element.
- Alternatively, you can use a "fragment" to wrap multiple lines. This will prevent unnecessarily adding extra nodes to the DOM.
- A fragment looks like an empty HTML tag: <></>.

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

const myElement = (
    <>
      <p>I am a paragraph.</p>
      <p>I am a paragraph too.</p>
    </>
  );

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

I am a paragraph.

I am a paragraph too.

Elements Must be Closed

- JSX follows XML rules, and therefore HTML elements must be properly closed.
- JSX will throw an error if the HTML is not properly closed.

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

const myElement = <input type="text" />;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Attribute class = className

- The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.
- Use attribute className instead.
- JSX solved this by using className instead. When JSX is rendered, it translates className attributes into class attributes.

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

const myElement = <h1 className="myclass">Hello World</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Hello World

Conditions - if statements

- React supports if statements, but not inside JSX.
- To be able to use conditional statements in JSX, you should put the if statements outside of the JSX, or you could use a ternary expression instead:

Option 1:

- Write if statements outside of the JSX code:

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

const x = 5;
let text = "Goodbye";
if (x < 10) {
  text = "Hello";
}

const myElement = <h1>{text}</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

Hello

Option 2:

- Use ternary expressions instead:

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

const x = 5;

const myElement = <h1>{(x) < 10 ? "Hello" : "Goodbye"}</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);
```

- Note that in order to embed a JavaScript expression inside JSX, the JavaScript must be wrapped with curly braces, {}.

Thank You