Nitesh Singh

Practical Guide on How

JSX

Works Under The Hood

We will build this simple UI

CSS will be same for all

style.css

```css
* {
  margin: 0;
  padding: 0;
}

.container {
  height: 100vh;
  display: flex;
  gap: 5px;
  justify-content: center;
  align-items: center;
  padding: 0 10px 0 10px;
  flex-direction: column;
}
p {
  font-size: 25px;
}

button {
  border: none;
  padding: 5px;
  border-radius: 5px;
}
```

Plain HTML CSS JS

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Reactive</title>
    <link rel="stylesheet" href="reactive_style.css">
</head>

<body>
    <div id="root">

        <div class="container">
            <p id="p">1</p>
            <button id="btn">Click Me</button>
        </div>

    </div>
</body>

<script>
    let p = document.getElementById("p")
    let btn = document.getElementById("btn")
    p.innerText = 0;
    p.className = "";

    let button = document.createElement("button");
    button.innerText = "Click Me";

    btn.addEventListener("click", () => {
        p.innerText++;
    });

</script>

</html>
```

adding code manually

Using Plain JS

index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Reactive</title>
    <link rel="stylesheet" href="reactive_style.css">
  </head>

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

  <script>
    const root = document.getElementById("root");

    let app = document.createElement("div");
    app.className = "container";

    let p = document.createElement("p");
    p.innerText = 0;
    p.className = "";

    let button = document.createElement("button");
    button.innerText = "Click Me";

    button.addEventListener("click", () => {
      p.innerText++;
    });

    app.appendChild(p);
    app.appendChild(button);

    root.appendChild(app);
  </script>
</html>
```

code will be add insde this

using JS to add elements

Using String Template Plain JS

index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Reactive</title>
    <link rel="stylesheet" href="reactive_style.css" />
  </head>
  <body>
    <div id="root"></div>
  </body>

  <script>
    const root = document.getElementById("root");
    let val = 1;

    function inc() {
      val++;
      render();
    }

    function render() {
      let app = `
                <div class="container">
                    <p>${val}</p>
                    <button type="button" onclick="inc()">Click Me</button>
                </div>
            `;
      root.innerHTML = app;
    }

    render();
  </script>
</html>
```

using string template

How JSX Works

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Reactive</title>
    <link rel="stylesheet" href="reactive_style.css" />
</head>

<body>
    <div id="root"></div>
</body>
// script here
</html>
```

index.html

```html
<script>
        const root = document.getElementById("root");
        let val = 1;

        function inc() {
            val++;
            render();
        }

        function destructure(props) {
            let data = "";
            for (const key of Object.keys(props)) {
                data += `${key}='${props[key]}' `;
            }
            return data.trim();
        }

        function traverse(childrens) {
            let code = "";
            if (!Array.isArray(childrens)) {
                return '';
            }
            code += `
        ${childrens.map((item) => {
                return `
            <${item.type} ${destructure(item.props)}>
                ${typeof item.children == "object" ? traverse(item.children) : item.children}
            </${item.type}>`
            }).join('')}
    `;

            return code;
        }
```

these function parse JSX tree

Here, I have shown a demo only. The actual code will be different and include many more things.

Nitesh Singh

Source Code

https://github.com/writer-nitesh/jsx-practical-guide