

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

```
* {
  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 x

```
<!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 x

```
<!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>
```

code will be add inside this

```
<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>
```

using JS to add elements

# Using String Template Plain JS

index.html x

```
<!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 x

```
<!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 x

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

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

these function parse JSX tree

```
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;
}
```

```
function render() {
  // JSX JSON tree
  let JSXJson = [{
    type: "div",
    props: { class: "container" },
    children: [
      {
        type: "p",
        props: { "aria-label": "test" },
        children: val, // Update directly with val
      },
      {
        type: "button",
        props: { type: "button", onclick: inc },
        children: "Click Me",
      },
    ]
  }];
}
```

JSX tree structure

it's how JSX stored under the hood

```
// Rendering JSX JSON
const code = JSXJson.map(item => {
  return `<${item.type} ${destructure(item.props)}> ${typeof item.children == "object" ?
  traverse(item.children) : item.children}</${item.type}>`;
}).join('');
root.innerHTML = code;
}
```

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

adding code in html element

equivalent to

ReactDOM.createRoot(document.getElementById('root')!).render(<Code />)

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>