JSX (JavaScript XML)

HTML inside JavaScript.

JavaScript inside HTML.

React look like HTML but behave like JavaScript.

JSX is not HTML — it’s syntactic sugar over React.createElement()

JSX (JavaScript XML) is a syntax extension for JavaScript used in React.

It lets you write HTML-like code inside JavaScript.

Embedding Expressions

```
const name = "Manas Kumar Lal";
const element = <h1>Hello, {name}!</h1>;
```

Valid Expressions:

- ✓ Variables
- ✓ Function calls
- ✓ Ternary expressions
- ✓ Mathematical operations

Note:

JSX and template literals both follow this rule: Only expressions can go inside {} or ${} and not statements.

Statements like if-else, for, etc. are not allowed.

JSX with Inline Styles

JSX style is an object, not a string

```
const element = (
  <div style={{
    color: "red",
    fontSize: "20px"
  }}>Hello Manas Kumar Lal</div>
)
```

```
const element = (
  <div style={{
    "color": "red",
    "font-size": "40px"
  }}>Hello Manas Kumar Lal</div>
)
```

Conditional Rendering in JSX

Using Ternary Operator

```
const isLoggedIn = true;

<p>{isLoggedIn ? "Welcome back!" : "Please log in."}</p>;
```

Short-circuit rendering

```
const isAdmin = true;

const element = isAdmin && <p>Admin Panel</p>;
```

JSX with Loops (Arrays)

JSX doesn’t support for, but we can use .map().

```
const items = ["Manas", "Muskan", "Mehek"];

<ul>
  {items.map((item, index) => (
    <li key={index}>{item}</li>
  ))}
</ul>
```

JSX is an Expression

Assign it to variables

Pass as props

Return from functions

```
function callMe(otherJSX) {
  return <h1>Suno, {otherJSX}</h1>;
}
```

Behind the Scenes

Compiled to React.createElement() behind the scenes.

JSX:

```
const element = <h1>Hello, School4U!</h1>;
```

Compiled:

```
const element = React.createElement("h1", null, "Hello, School4U!");
```

React.createElement(type, props, ...children)

Practice:

Convert these JSX into createElement syntax:

```
const element = <h1>Hello, Muskan!</h1>;
```

```
const element = <h1 className="heading">Hello, Muskan!</h1>;
```

```
const element = <a href="https://www.school4u.in" className="link">School4U</a>;
```

```
const element = <h1 className="heading">Hello, {name}!</h1>;
```

```
const element = <div>Hello, {alpha}. You are {age} years old.</div>
const element = (
  <div>
    <h1>Hello, Mhek</h1>
  </div>
);
```

```
const element = (
  <di className="card">
    <h1>Hello, {user.name}</h1>
    <p>You are {user.age + 1} years old next year.</p>
  </di>
);
```

```
const element = (
  <div
    className='alpha'
    style={{
      color: "red",
      fontSize: "20px"
    }}
  >
    Hello
  </div>
)
```

JSX Rules:

1. Must return single parent element. Wrap with div (<div></div>) or fragments (<></>)

2. Use "className" and "htmlFor" props instead of "class" and "for" attribute. (class & for are reserved keywords in JS)

3. Self-closing tags are used for elements without children.

4. Cannot use if-else directly – use ternary or logical &&

5. JSX supports only expressions inside {} – not statements.