JSX is NOT HTML

- JSX = **JavaScript XML**.
- Looks like HTML, but it is actually **JavaScript syntax**.
- Browser cannot understand JSX directly → React converts it to JavaScript.

Why JSX is not HTML?

- HTML runs directly in browser.
- JSX needs to be **compiled** by Babel → converted into plain JS.
- JSX can embed JS:

```
<h1>{name}</h1>
```

HTML vs JSX Differences

1.class → className

HTML:

```
<h1 class="title">Hello</h1>
```

JSX:

```
<h1 className="title">Hello</h1>
```

2. Self-closing Tags

HTML:

```
<img src="a.png">
```

JSX:

```
<img src="a.png" />
```

3. Attributes use camelCase

HTML:

```
<button onclick="handle()">
```

JSX:

```
<button onClick={handle}>
```

4. JSX must return a single parent element

HTML: unlimited structure

```
<h1>Hello</h1>
<p>World</p>
```

JSX: MUST wrap in one parent

❌ But **JSX me yeh allowed nahi**:

```
return (
  <h1>Hello</h1>
  <p>World</p>
)
```

React component **ek hi element return kar sakta hai**.
Isliye tumhe **ek parent container** ke andar saare elements rakhne padte hain

```
return (
  <div>
    <h1>Hello</h1>
    <p>World</p>
  </div>
);
```

📌 **Babel Document**

Is JSX a valid JavaScript?

The answer is yes and no. JSX is not a valid Javascript syntax as it's not pure HTML or pure JavaScript for a browser to understand. JS does not have built-in JSX. The JS engine does not understand JSX because the JS engine understands ECMAScript or ES6+ code

If the browser can't understand JSX how is it still working?

This is because of Parcel because "Parcel is a Beast". Before the code gets to JS Engine it is sent to Parcel and Transpiled there. Then after transpilation, the browser gets the code that it can understand.

Transpilation ⇒ Converting the code in such a format that the browsers can understand.

Parcel is like a manager who gives the responsibility of transpilation to a package called **Babel.**

Babel is a package that is a compiler/transpiler of JavaScript that is already present inside 'node-modules'. It takes JSX and converts it into the code that browsers understand, as soon as we write it and save the file. It is not created by Facebook.

Learn more about Babel on babeljs.io

JSX (transpiled by Babel) ⇒ React.createElement ⇒ ReactElement ⇒ JS Object ⇒ HTML Element(render)

Single Line and Multi Line JSX Code

Single line code:

```
const jsxHeading = <h1>Namaste React</h1>
```

Multi-line code:

If writing JSX in multiple lines then using '()' parenthesis is

mandatory. To tell Babel from where JSX is starting and ending.

```
const jsxHeading = (
<div>
<h1>Namaste React</h1>
</div>
)
```

Introducing React Components

Everything inside React is a component.

Q) What are Components?

There are 2 types of components:

1.Class-based Components - Old way of writing code, used rarely in industry

2.Functional Components - New way of writing code, most commonly used

What is a React Functional Components?

It is just a JavaScript Function that returns some JSX or a

react element.

Always name React Functional Component with Capital Letters

otherwise you will confuse it with normal function

```
const HeadingComponent1 = () => (
<h1>Namaste</h1>
)
```

```
const HeadingComponent2 = () => {
return <h1>Namaste</h1>
}
```

```
const HeadingComponent3 = () => <h1>Namaste</h1>
```

To render a functional component we call them

‘<Heading1 />’.

This is the syntax that Babel understands.

You can also call them using these ways,

‘<Title></Title>’

or

‘{Title()}’

Components Composition

A component inside a component.

Calling a component inside another component is Component Composition

```
const Title = () => <h1>Namaste React</h1>

const HeadingComponent = () => (

<div id="container">

<Title />

</div>

)
```

Code inside the ‘Title’ component will be used inside the

‘HeadingComponent’ component as the ‘Title’ component is called

inside it. It will become something like this,

```
const HeadingComponent = () => (

<div id="container">

<h1>Namaste React</h1>

</div>

)
```

How to use JavaScript code inside JSX?

Inside a React Component when **‘{}’** parenthesis is present we can

write any JavaScript expression inside it.

```
const number = 10000;

const HeadingComponent = () => (

<div id="containter">

{number}

<h1>Namaste React</h1>

</div>

)
```

How to call React Element in JSX?

We can use ‘{}’ parenthesis.

```
const elem = <span> React Element </span>

const HeadingComponent = () => (

<div id="containter">

Laying the Foundation! □Namaste-React)11{elem}

<h1>This is Namaste React</h1>

</div>

)
```

What will happen if we call 2 elements inside each other?

If we put 2 components inside each other, then it will go into an infinite loop and the stack will overflow. It will freeze your browser, so it’s not recommended to do so.

Advantages of using JSX.

1) Sanitizes the data

If someone gets access to your JS code and sends some malicious data which will then get displayed on the screen, that attack is called cross-site scripting.

It can read cookies, local storage, session storage, get cookies, get info about your device, and read data. JSx takes care of your data.

If some API passes some malicious data JSX will escape it. It prevents cross-site scripting and sanitizes the data before rendering

2) Makes code readable

JSX makes it easier to write code as we are no longer creating elements using React.createElement()

3) Makes code simple and elegant

4) Show more useful errors and warnings

5) JSX prevents code injections (attacks)