1. What is JSX?

JSX stands for **JavaScript XML**. It allows developers to write HTML-like syntax directly within JavaScript code, making it easier and more intuitive to create React elements. JSX provides a more readable and concise way to define UI components compared to using React.createElement(), which can become verbose and hard to manage in complex UIs. Although it looks like HTML, **JSX is not HTML** — it's syntactic sugar that gets transpiled into React.createElement() by transpilers like Babel. Ultimately, JSX returns JavaScript objects, just like React.createElement() does.

2. **Superpowers of JSX:**

- **Transpiles to** React.createElement()
  JSX is syntactic sugar for React.createElement(). Under the hood, JSX is converted into JavaScript objects with a special $$typeof property set to React.element, and the type corresponds to the HTML tag or React component being rendered.
- **Simplified Syntax**
  JSX provides an HTML-like syntax that is easier and more intuitive to write compared to raw JavaScript function calls.
- **Improved Readability and Maintainability**
  The structure and formatting of JSX closely resemble actual HTML, making it easier for developers to understand, debug, and maintain the UI code.
- **Component Composition**
  JSX supports composing UI by combining smaller components into larger ones, encouraging modular and reusable code.
- **Supports Nesting**
  Components and elements can be easily nested inside one another, which is essential for building hierarchical UI structures.
- **Scales Well for Complex UIs**
  JSX's expressive nature allows developers to build and manage complex user interfaces efficiently.

3. **What is the role of the type attribute in the <script> tag? What options can I use there?**
   The type attribute in the <script> tag specifies the MIME type (media type) of the script, telling the browser how to interpret the code inside the tag (or the file referenced by the src attribute).

   Common values for the type attribute:

- **text/javascript** *(default)*
  This is the default value and can be omitted. It indicates that the script is standard JavaScript.

- **module**
  This specifies that the script is a JavaScript **module**, allowing you to use import and export statements. Modules are automatically deferred and scoped.
  <script type="module" src="app.js"></script>

- **application/json**
  Used when embedding JSON data within a <script> tag. The contents won't be executed as code — it's usually accessed by JavaScript for configuration or templating purposes.
  <script type="application/json" id="config-data">
  { "theme": "dark", "lang": "en" }
  </script>

- **Custom types**
  You can use custom type values for things like templating engines (e.g., Handlebars, JSX, or GraphQL). The browser will ignore the content, and JavaScript can read and process it manually.
  <script type="text/x-handlebars-template" id="template"></script>

- **Summary:**
  The type attribute helps the browser know how to handle the content of the <script> tag. It's essential when using **modules**, embedding **JSON**, or working with **template languages** or **custom data**.

4. **What is a MIME type?**

MIME stands for **Multipurpose Internet Mail Extensions**. Despite the name, MIME types are widely used on the web to indicate the **type of content** being handled, not just in email.

A **MIME type** (also called a **media type**) tells the browser or any client what kind of data is being sent or received, so it knows how to process or display it.

5. {TitleComponent} vs {< TitleComponent />} vs {< TitleComponent ></ TitleComponent >} in JSX

Syntax	**Meaning**	**Use Case**
{TitleComponent}	Reference to the component (function/class)	Pass component as a prop, or dynamically render
{<TitleComponent />}	Render the component	Standard rendering of components
{<TitleComponent></TitleComponent>}	Same as above, but with long-form tag	Needed when passing children