1. What is JSX?

Ans:-

It is called as syntax expression to JavaScript. JSX may remind you of HTML markup, but it comes with full power of JavaScript. JSX ultimately produces React Elements which then get converted to JavaScript object and then to HTML(DOM).

1. What are the super powers of JSX?

Ans:-

Below we are declaring one variable and then using it inside JSX by wrapping in curly braces.

Const name = ”Srushti”;

Const element = <h1>Hello, {name} </h1>;

1. You can put any valid JS syntax inside the curly braces in JSX.
2. 2. JSX is an expression too. That means after compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects. Thus, you can use JSX inside if,for loops.

function getGreeting(user) {

if (user)

{ return <h1>Hello, {formatName(user)}!</h1>;

}

return <h1>Hello, Stranger.</h1>;

}

1. JSX prevents injection attacks. It is safe to embed user inputs in JSX.

Eg, cont title=”someMaliciousContent”

Const element =<h1>{title}</h1>

By default,react DOM escapes any values embedded in JSX before rendering them. Thus it ensures that you can never inject anything that is not written in your application.

Everything is converted to string before it renders. This avoids XSS attacks (Cross-site scripting).

1. Ultimately JSX represents objects/react elements
2. You can specify attributes with JSX with “ “ or { }

Eg: const element = <img src=”https://www.google.com-logo.jpeg></img> OR

const element = <img src={ <https://www.google.com-logo.jpeg> }></img>

1. You can specify more children in JSX

Eg: const element = (

<div>

<h1>Heading 1<h1>

<h2>Heading 2</h2>

</div>

);

1. {TitleComponent} vs {<TitleComponenet/>} vs {<TitleComponenet><TitleComponent/>}

Ans:-

First one is the way of rendering react element which gets converted to JavaScript variable behind the scenes. Second and Third is the way of rendering functional components in react.

1. Role of type attribute in script tag? What options can I use there?

Ans:-

Type will decided which type of script you are rendering in browser.

Type I have three values:

Attribute is not set (default), an empty string, or a JavaScript MIME type. Indicates that the script is a "classic script", containing JavaScript code. Authors are encouraged to omit the attribute if the script refers to JavaScript code rather than specify a MIME type.

1. Type= ”module”

This value causes the code to be treated as a JavaScript module. The processing of the script contents is deferred. The charset and defer attributes have no effect. For information on using module, see our [JavaScript modules](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules) guide. Unlike classic scripts, module scripts require the use of the CORS protocol for cross-origin fetching.

1. Type= ”importmap”

This value indicates that the body of the element contains an import map. The import map is a JSON object that developers can use to control how the browser resolves module specifiers when importing [JavaScript modules](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules#importing_modules_using_import_maps).

1. Type= ”blocking”

This attribute explicitly indicates that certain operations should be blocked on the fetching of the script.