**Laying the foundation**

Q1> What is JSX?

Ans>

const element = <h3>Namaste React</h3>;

1-This is called JSX. It’s neither HTML nor string in Javascript.

2-JSX is HTML like syntax but not HTML. If we console.log element it will give “Uncaught SyntaxError: Unexpected token '<'”

because browser does not understand angle bracket. It is **BABEL** who convert this code in older code which is compatible for browser.

BABEL coverts above code like this -

const heading1 = React.createElement ("h3",

{id:"title",

key:"h3"}

,

"Namaste React"

);

this will give Object which is render later as HTML and inject into DOM by ReatDOM. That's how JSX works

3- It was created as a syntactic sugar for React.createElement(). It is an extension of JavaScript that allows developers to write HTML right within JavaScript.

Q2> Difference between JSX and HTML?

Ans> 1- You need to return a single parent element in JSX for that we use fragment

2- You can implement JS expression directly in JSX- by using curly braces {}

exp-

const Article = () => {

return (

<>

<div>

<p>Hey Dear</p>

<p>What is up?</p>

{ new Date().toDateString() }

<br />

<br />

{200 + 15} is seven in word

<br />

</div>

</>

);

};

export default Article;

3- All Tags Self-close in JSX otherwise it will give error

exp-

<img src=" />

4- ClassName and HTML For, not class and for in JSX

exp-

<label htmlFor="name">Name</label>

<div className="container">

5- Write all HTML Attributes in camelCase in JSX

exp-

<button onClick={sayHI}>ALert Hi</button>

<label htmlFor="name">Name</label>

6- Write Inline Styles as Objects in JSX

exp-

const inlineStyle = {

color: "#2ecc71",

fontSize: "26px",

};

Q3> Superpowers of JSX?

Ans>

1. JSX Represents Objects as Babel compiles JSX down to React.createElement() calls.
2. JSX Prevents Injection Attacks

By default, React DOM escapes any values embedded in JSX before rendering them. Thus it ensures that you can never inject anything that’s not explicitly written in your application. Everything is converted to a string before being rendered. This helps prevent XSS (cross-site-scripting) attack

1. Specifying Children with JSX

JSX tags may contain children:

const element = (

<div>

<h1>Hello!</h1>

<h2>Good to see you here.</h2>

</div>

);

1. JSX is an Expression Too

function getGreeting(user) {

if (user) {

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

}

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

}

1. We can embed expressions in JSX

function formatName(user) {

return user.firstName + ' ' + user.lastName;

}

const user = {

firstName: 'Sam',

lastName: 'Jolly'

};

const element = (

<h1>

Hello, {formatName(user)}!

</h1>

);

Q4> Role of Type in Script tag. What options can I use there?

Ans>

This attribute indicates the type of script represented. The value of this attribute will be one of the following:

A> **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.

<script type="module" src="main.js"></script>

B> **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 module

<script type="importmap">

// JSON object defining import

</script

Q5> Difference among {Titlecomponent} vs {<Titlecomponent/>} vs <Titlecomponent><Titlecomponent/>}

Ans> {Titlecomponent} = add JS Expression inside JSX

{<Titlecomponent/>} = add component inside another component (composition of component)

{<Titlecomponent><Titlecomponent/>} is same like {<Titlecomponent/>}. It just a different syntax.