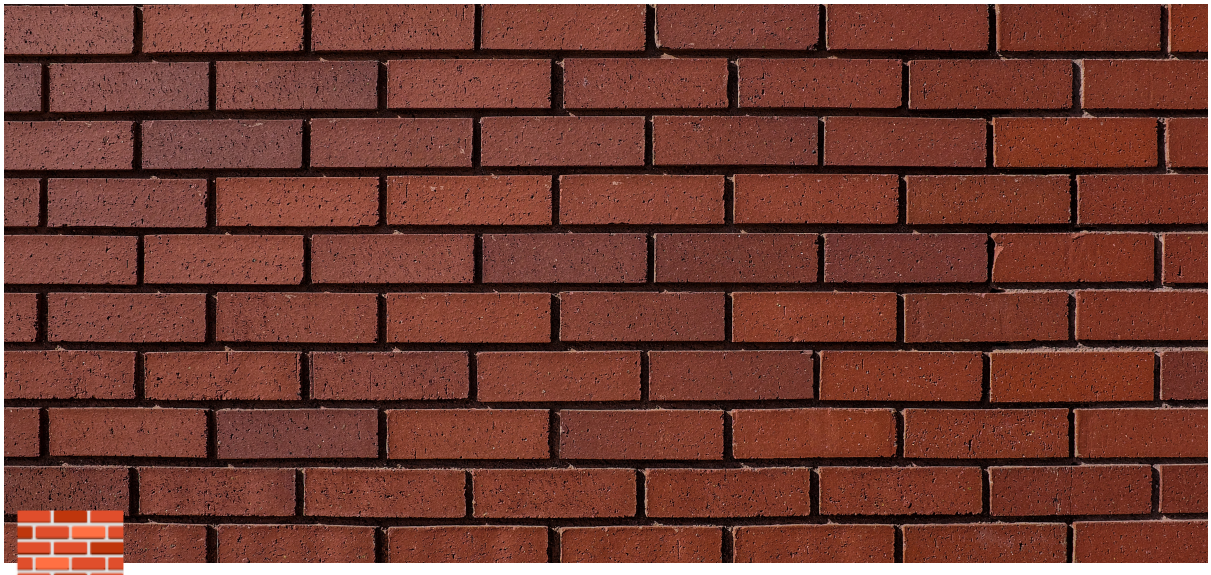


Laying the Foundation

Namaste React

https://nishant-assignment3.netlify.app/

▼ What is JSX ?

```
const element = React.createElement('h1', {}, 'Hello World!');
const root = ReactDOM.createRoot(document.querySelector('#root'));
root.render(element);
```

As the syntax of React.createElement() is longer and when we write more complex UI it becomes more tedious. To solve this issue we have JSX which is a syntactic sugar which is just like HTML but not HTML. This syntax is created by React and browser cannot understand it . We need to use a tool such a Babel to convert this JSX code to regular JavaScript code.

```
const element = <h1>Hello World!</h1>;
const root = ReactDOM.createRoot(document.querySelector('#root'));
```

```
root.render(element)
```

▼ What are the Superpowers of `JSX?`

As JSX returns an object we can do things such as:

- Assign it as a variable
- Return it from a function
- Conditionally render different elements
- JSX also helps to prevent XSS attacks. It Sanitizes the data in React.

```
// 1) Assigning as a variable

const heading = <h1 className="tilte">Heading1 </h1>


// 2) Return from a function
function getOurHeading() {

    return <h1 className="tilte">Heading1 </h1>


}

// 3) Use of Conditional rendering

function getHeading(is_heading) {
  if(is_heading) {
          return <h1 className="tilte">Heading1 </h1>

  } else {

    return <h2 className="tilte">Heading2 </h1>
}


}
```

▼ What is the Role of `type` attribute in script tag? What options can I use there?

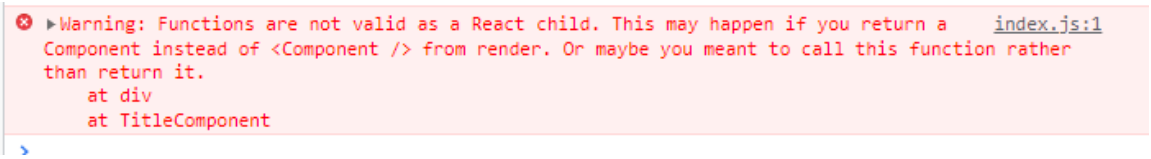
```
<script type="text/javascript"></script>
```

- `text/javascript` : It is the basic standard of writing JavaScript code inside the `<script>.`
- `text/ecmascript` : The script is followingthe ECMAScript standards.
- `module`: The script is a module that can import or export other files or modules inside it.
- `text/babel` : This value indicates that the script is a babel type and requires babel to transpile it.
- `text/typescript`: This script is written in `TypeScript`.

▼ `{TitleComponent}` vs `{<TitleComponent/>}` vs `{<TitleComponent></TitleComponent>}` in `JSX`.

- `{TitleComponent}`: Expressions in JSX are wrapped using {} . This value describes the `TitleComponent` as a javascript expression.
- `<TitleComponent/>` : This value represents a Component that is basically returning Some JSX value. A component is written inside the `{< />}` expression.
- `<TitleComponent></TitleComponent>` : `<TitleComponent />` and `<TitleComponent></TitleComponent>` are equivalent only when `< TitleComponent />` has no child components. The opening and closing tags are created to include the child components.

```
<TitleComponent>
    <FirstChildComponent />
    <SecondChildComponent />
    <ThirdChildComponent />
</TitleComponent>
```



We need to call our component using the <> or > {</>}. Else it will throw message as the above.