Assignment 3 - Laying the Foundation

1. **What is JSX ?**

   JSX is a syntax extension to JavaScript that allows developers to write HTML-like syntax in their JavaScript code.
   JSX allows developers to write HTML-like/xml like code within JavaScript. It provides a more declarative and intuitive way to define the structure and content of user interface components. JSX code looks like a combination of HTML and JavaScript, making it easier to visualize and build complex UIs.
   Under the hood, JSX gets transpiled (converted) into standard JavaScript by tools like Babel before being executed by the browser.
   It also allows us to use the full power of JavaScript within our UI code, such as using expressions and variables within our JSX code.

   JSX becomes a React.createElement after the transpilation which is a Javascript object. This gets converted to HTML when we render the code in React.

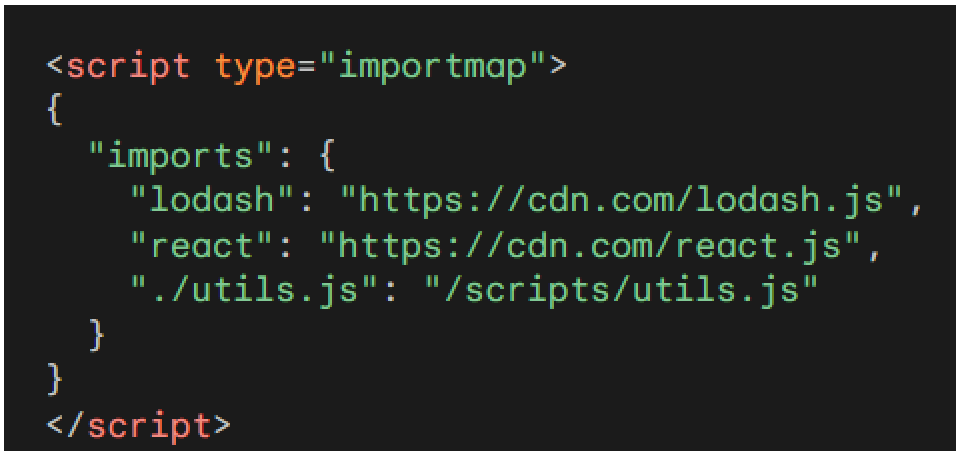
2. **Superpowers of JSX ?**

- Makes it easier to write/read and add HTML in React.
- Makes code more readable
- JSX allows us to embed JavaScript expressions within curly braces {}. This enables dynamic content rendering, conditional rendering, and the use of variables and functions directly within JSX.
- Static Type Checking: When using TypeScript or Flow with React, JSX provides additional benefits like static type checking, which can catch type-related errors during development, leading to more robust and bug-free applications.
- JSX encourages a component-based architecture, where UI elements are broken down into smaller, self-contained components. This modularity enhances code reusability, maintainability, and scalability.

3. **Role of type attribute in the script tag? What values can I use there?**

- **text/javascript:** This is the default value and indicates that the content inside the script tag is JavaScript code.
- **text/ecmascript:** This value indicates that the content inside the script tag is ECMAScript, which is the official name for the JavaScript language specification.

- **application/javascript:** This value also indicates that the content is JavaScript code. It is similar to text/javascript, but it is preferred when serving JavaScript as a separate file through HTTP.
- **application/ecmascript:** This value indicates that the content inside the script tag is ECMAScript.
  module: This value is used when you are using JavaScript modules. It is used to indicate that the script is a module and should be treated as such by the browser.
- **text/jsx:** This value is used when you are using JSX (JavaScript XML) in your script. JSX is typically used with React to define components.
- **Importmap**: This value indicates that the script contains an import map. An import map is a JSON object that allows developers to control how the browser resolves module specifiers when importing JavaScript modules.



```
<script type="importmap">
{
  "imports": {
    "lodash": "https://cdn.com/lodash.js",
    "react": "https://cdn.com/react.js",
    "./utils.js": "/scripts/utils.js"
  }
}
</script>
```

4. **{TitleComponent} vs {<TitleComponent />} vs {<TitleComponent> </TitleComponent>} in JSX.**

- **{TitleComponent}:** This value describes the TitleComponent as a javascript expression or a variable. The {} can embed a javascript expression or a variable inside it.
- **<TitleComponent/> :** This value represents a Component that is basically returning Some JSX value. In simple terms TitleComponent is a component that is 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.