Class 3: Assignment

▼ What is JSX?

- Using `React.createElement` is a long process and it becomes tedious when we are writing complex UI components. Hence, JSX was introduced.
- It is an HTML-like syntax extension to JavaScript.
- `const greeting = <h1> Hello World </h1>;`
- JSX uses `react.createElement` behind the scenes to create React elements.
- The browser does not understand JSX and hence it needs Babel to convert the code to be compatible with older browsers.

▼ Superpowers of JSX.

▼ Embedding expressions in JSX

We declare a variable called `name` and then use it inside JSX by wrapping it in curly braces.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

- You can put any valid JavaScript expression inside the curly braces in JSX.
- For example, `2 + 2`, `user.firstName`, or `formatName(user)` are all valid JavaScript expressions.

▼ JSX is an expression

- After compilation, JSX expressions become regular JavaScript function calls and evaluate to JavaScript objects.
- This means that you can use JSX inside of `if` statements and `for` loops, assign it to variables, accept it as arguments, and return it from functions:

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

▼ Specifying children with JSX

JSX tags may contain children. And a key should be used when there is more than 1 child.

```
const element = (
  <div>
    <h1>Hello!</h1>
    <h2>Good to see you here.</h2>
  </div>
);
```

▼ JSX prevents injection attacks.

How?

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) attacks.

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

- The `<script>` HTML element is used to embed executable code or data; this is typically used to embed or refer to JavaScript code.
- The `type` attribute of the `<script>` element indicates the *type* of script represented by the element: a classic script, a JavaScript module, an import map, or a data block.
- The value of this attribute will be one of the following:

1. **Attribute is not set (default), an empty string, or a JavaScript MIME type.** Indicates that the script is a "classic script", containing JavaScript code.

2. `module` - This value causes the code to be treated as a JavaScript module.
3. `importmap` - This value indicates that the body of the element contains an import map.

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

- `{TitleComponent}` to render a JSX element
- `{<TitleComponent/>}` and `{<TitleComponent></TitleComponent>}` to render a functional component.