Chapter 03 - Laying the Foundation

Topics

- JSX
- React.createElement vs JSX
- Benefits of JSX
- Behind the Scenes of JSX
- Babel & parcel role in JSX
- Components
- Functional Components
    - Composing Components

Assignment

- What is JSX?

  JSX stands for **JavaScript XML**.
  It lets you write HTML-like syntax inside JavaScript. It's not HTML, but it looks like it — and makes writing UI in React much easier.

- Superpowers of JSX

- **Readable** – Looks like HTML, so easier to understand.
- **Sanitizes data** – Protects against code injection and XSS(cross site scripting) attacks.
- **Elegant** – Shorter and cleaner than `React.createElement()`.
- **Better Errors** – Shows more helpful errors and warnings.
- **Supports JavaScript inside** – You can embed `{variables}` and expressions directly.

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

  The type attribute defines what kind of script you're writing.

  Common values:

  - text/javascript: Default for normal JavaScript.
  - module: For modern JavaScript using import/export.

  You use type="module" when using ES6 modules in your browser.

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

Syntax	Meaning
{TitleComponent}	Just refers to the component function itself (won't render it).
{<TitleComponent />}	Renders the component (recommended way).
{<TitleComponent></TitleComponent>}	Also renders the component, same as above.

Coding Assignment:

- Create a Nested header Element using React.createElement(h1,h2,h3 inside a div with class “title”)

```
const header = React.createElement(
  "div",
  { className: "title" },
  [
    React.createElement("h1", {}, "This is H1"),
    React.createElement("h2", {}, "This is H2"),
    React.createElement("h3", {}, "This is H3")
  ]
);

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(header);
```

  - Create the same element using JSX

```
const jsxHeader = (
  <div className="title">
    <h1>This is H1</h1>
    <h2>This is H2</h2>
    <h3>This is H3</h3>
  </div>
);
```

  - Create a functional component of the same with JSX

```
const Header = () => (
  <div className="title">
    <h1>This is H1</h1>
    <h2>This is H2</h2>
    <h3>This is H3</h3>
  </div>
);
```

  - Pass attributes into the tag in JSX

```
const Header = () => (
  <div className="title" id="main-header">
    <h1 style={{ color: "red" }}>This is H1</h1>
    <h2 title="subtitle">This is H2</h2>
    <h3 className="heading3">This is H3</h3>
  </div>
);
```

- Composition of Component(Add a component inside another)

```
const Title = () => <h1>This is the Title</h1>;

const Header = () => (
  <div>
    <Title />
    <h2>This is a subtitle</h2>
  </div>
);
```

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

```
const TitleComponent = () => <h1>Hello JSX!</h1>;

// These are valid renderings:
{<TitleComponent />}
{<TitleComponent></TitleComponent>}

// This won't render, just references the function:
{TitleComponent}
```

- Create a Header Component from scratch using Functional Components with JSX
  - Add a Logo on left
  - Add a search bar in middle
  - Add User icon on right
  - Add CSS to make it look nice

```
const Header = () => (
  <div className="header">
    <img className="logo" src="logo.png" alt="Logo" />
    <input type="text" placeholder="Search..." className="search" />
    <img className="user-icon" src="user.png" alt="User" />
  </div>
);
```

```css
.header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px;
  border-bottom: 1px solid #ddd;
}

.logo {
  height: 50px;
}

.search {
  width: 300px;
  padding: 8px;
  font-size: 16px;
}

.user-icon {
  height: 40px;
  border-radius: 50%;
}
```

References

- Babel: https://babeljs.io/
- Attribute Type: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/script#attr-type

- JS Modules: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules
- Babel Playground: https://babeljs.io/repl#
- React without JSX: https://reactjs.org/docs/react-without-jsx.html