**ReactJS Expressions**

JSX supports expression in pure JavaScript syntax. Expression has to be enclosed inside the curly braces, **{** **}**. Expression can contain all variables available in the context, where the JSX is defined. Let us create simple JSX with expression.

*Example:*

*ReactExpression.js*

import React from "react";

function ReactExpression() {

  const time = new Date().toLocaleString();

  return (

    <div>

      <h3>React Expression</h3>

      {/\* ReactExpression {time} \*/}

      <h4>Current Time is: {time}</h4>

    </div>

  );

}

export default ReactExpression;

*App.js*

import "./App.css";

import ReactExpression from "./Components/ReactExpression";

function App() {

  return (

    <div className="App">

      <ReactExpression />

    </div>

  );

}

export default App;

***ReactJS Attributes***

JSX supports HTML like attributes. All HTML tags and its attributes are supported. Attributes has to be specified using *camelCase* convention instead of normal HTML attribute name. For example, class attribute in HTML has to be defined as className. The following are few other examples –

* *htmlFor* instead of *for*
* *tabIndex* instead of *tabindex*
* *onClick* instead of *onclick*

*Example:*

*ReactLoginForm.js*

import React from "react";

import { useState } from "react";

const ReactLoginForm = () => {

  const [name, setName] = useState("");

  const [password, setPassword] = useState("");

  const [confPassword, setConfPassword] = useState("");

  const handleName = (event) => {

    setName(event.target.value);

  };

  const handlePassword = (event) => {

    setPassword(event.target.value);

  };

  const handleConfPassword = (event) => {

    setConfPassword(event.target.value);

  };

  const HandleSubmit = (event) => {

    if (password !== confPassword) {

      alert("Passwords Didn't Match!");

    }

    alert(

      `The Form Submitted with Name : ${name}, Age: ${age}, Email: ${email}`

    );

    event.preventDefault();

  };

  return (

    <div>

      <div className="container">

        <form onSubmit={HandleSubmit}>

          <table>

            <tr>

              <td className="text-start">Name: </td>

              <td>

                <input

                  className="form-control form-control-sm mt-0"

                  type="text"

                  value={name}

                  onChange={handleName}

                />

              </td>

            </tr>

            <tr>

              <td className="text-start">Password: </td>

              <td>

                <input

                  className="form-control form-control-sm mt-0"

                  type="password"

                  value={password}

                  onChange={handlePassword}

                />

              </td>

            </tr>

            <tr>

              <td className="text-start">Confirm Password: </td>

              <td>

                <input

                  className="form-control form-control-sm mt-0"

                  type="password"

                  value={confPassword}

                  onChange={handleConfPassword}

                />

              </td>

            </tr>

            <tr>

              <td colSpan="1"></td>

              <td>

                <button className="btn btn-outline-primary" type="submit">

                  Submit

                </button>

              </td>

            </tr>

          </table>

        </form>

      </div>

    </div>

  );

};

export default ReactLoginForm;

**JSX Basics**

As we have already seen that, all of the React components have a render function. The render function specifies the HTML output of a React component. JSX(JavaScript Extension), is a React extension which allows writing JavaScript code that looks like HTML. In other words, JSX is an HTML-like syntax used by React that extends ECMAScript so that HTML-like syntax can co-exist with JavaScript/React code. The syntax is used by preprocessors (i.e., transpilers like babel) to transform HTML-like syntax into standard JavaScript objects that a JavaScript engine will parse.

JSX provides you to write HTML/XML-like structures (e.g., DOM-like tree structures) in the same file where you write JavaScript code, then preprocessor will transform these expressions into actual JavaScript code. Just like XML/HTML, JSX tags have a tag name, attributes, and children.

***Why use JSX?***

* It is faster than regular JavaScript because it performs optimization while translating the code to JavaScript.
* Instead of separating technologies by putting markup and logic in separate files, React uses components that contain both.
* It is type-safe, and most of the errors can be found at compilation time.
* It makes easier to create templates.

*Example:*

  return (

    <div>

      <h3>React JSX Basics</h3>

      <h4 style={{ color: "red" }}>Current Time is {getCurrentTime()}</h4>

    </div>

  );

***Difference between JavaScript and JSX***

|  |  |
| --- | --- |
| *JavaScript* | *JSX* |
| JavaScript is an ECMAScript following scripting language which allows developers to make their web pages more interactive. Every time a webpage loads something, or a slideshow plays, or an animated picture is displayed, JavaScript is involved behind it. Basically it adds life and creativity into a website, making it more engaging. | JSX is just a syntax extension of JavaScript which allows users to write React components; React is an open source frontend JavaScript library for building complex UIs; it allows the programmers to create complex UIs from smaller components.  JSX or JavaScript XML combines HTML and JavaScript, making the code easier to read and understandable for the user. JSX combines interactivity with markup rather than separating the two. It makes it easier to visualize DOM. In JSX we can directly write HTML tags inside JavaScript code. |

**Rendering HTML**

React renders HTML to the web page by using a function called *ReactDOM.render().* This function takes two arguments, HTML code and an HTML element.The purpose of the function is to display the specified HTML code inside the specified HTML element.

The Html is rendered in the root directory of your React project, named "public". In this folder, there is an index.html file.

ReactDOM.render(<p>Hello</p>, document.getElementById('root'));

You'll notice a single <div> in the body of this file. This is where our React application will be rendered.

<body>

  <div id="root"></div>

</body>