**React JSX**

We all know that all React components have a **render function**. The **render function** specifies the HTML output of the React component. JSX (JavaScript Extension) is a React extension which allows us to write 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 object that a JavaScript engine can 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 preprocessors will transform these expressions into actual Javascript code. Just like HTML/XML , JSX tags have a tag name , attributes and children.

Example

Here , we will write JSX syntax in JSX file and see the corresponding JavaScript code which transforms by preprocessors (babel)

JSX code

<div>Hello React </div>

Corresponding Output

React.createElement(“div”,null,”Hello React”);

The above line creates a **react element**  and passing **three arguments** inside where the first is the **name of the element** which is div , second is the **attributes** passed by the div tag , and the last is the **content** you pass which is “Hello React”.

Why use JSX ?

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

Nested Elements in JSX

To use more than one element , you need to wrap it with one container element. Here , we use div as container element which has three nested elements inside it.

CODE

App.jsx

import React , {Component} from ‘react’

class App extends Component{

render(){

return(

<div>

<h1>My first Project</h1>

<h2>Practice </h2>

<p>This is my first code </p>

</div>

);

}

}

export default App;

OUTPUT

My first Project

Practice

This is my first code

JSX Attributes

JSX uses attributes same as regular HTML. JSX uses camelCase naming convention for attributes rather than standard naming convention of HTML such as class in HTML becomes **className** in JSX because the class is the reserved keyword in JavaScript. We can also use our own custom attributes in JSX. For custom attributes , we need to use data-prefix. In the below example , we have used a custom attribute **data-demoAttribute** as an attribute for the <p> tag.

Example

CODE

import React , {Component} from ‘react’;

class App extends Component{

render(){

return(

<div>

<h1>React </h1>

<h2>My first project</h2>

<p data-demoAttribute = “demo”>This is my first project</p>

</div>

);

}

}

export default App;

In JSX we can specify attribute values in two ways :

1. As String Literals : We can specify the values of attributes in double quotes.

var element = <h2 className = “firstAttribute”>Hello React </h2>

Example

import React , {Component} from ‘react’;

class App extends Component{

render(){

return(

<div>

<h1 className =”hello”> React </h1>

<p data-demoAttribute = “demo”>This is my first React Project </p>

</div>

);

}

}

export default App;

OUTPUT

React

This is my first React Project

1. As Expressions : We can specify the values of attributes as expressions using curly braces {}.

var element = <h2 className= {varName}> React </h2>

Example

import React , {Component} from ‘react’

class App extends Component {

render(){

return(

<div>

<h1 className = “hello”>{25+20}</h1>

</div>

);

}

}

export default App;

OUTPUT

45

JSX Comments

JSX allows us to use comments that begin with /\* and ends with \*/ and wrapping them in curly braces {} just like in the case of JSX expressions.

Eg

CODE

import React , {Component} from ‘react’

class App extends Component {

render(){

return(

<div>

<h1 className = “hello”>{25+20}</h1>

{/\* This is a comment in JSX \*/}

</div>

);

}

}

export default App;

JSX styling

React always recommends to use **inline** styles. To set inline styles , you need to use camelCase syntax. React automatically allows appending px (pixels) after the number value on specific elements.

Eg

CODE

import React , {Component} from ‘react’

class App extends Component {

render(){

var myStyle = {

fontSize : 80,

fontFamily : ‘Courier’,

color: ‘#003300’

}

return(

<div>

<h1 style = {myStyle}>www.javatpoint.com</h1>

</div>

);

}

}

export default App;

Eg

import React , {Component} from ‘react’

class App extends Component {

render(){

var i = 5;

return(

<div>

<h1>{i == 1 ? ‘True’ : ‘False’} </h1>

</div>

);

}

}

export default App;

OUTPUT

false