**React JSX**

React differs from other libraries because it adopts a whole new language called JSX to illustrate what the visuals will look like.

What is JSX?

* JSX stands for JavaScript XML.
* JSX is a language that allows you to mix JavaScript and HTML-like tags to define user interface (UI) elements and their functionality.
* JSX is an extension for JavaScript that allows for writing what looks like HTML and XML in our JavaScript.
* JSX is a shorthand for calling **React.createElement()** to create elements and components.
* JSX looks like HTML and offers a friendly syntax for creating DOM elements.
* JSX accepts any valid JavaScript expression inside curly braces.

Create React Element Using JSX

Open a new command prompt, navigate to the folder where you want to create your React project and run the following command:

create-react-app example3

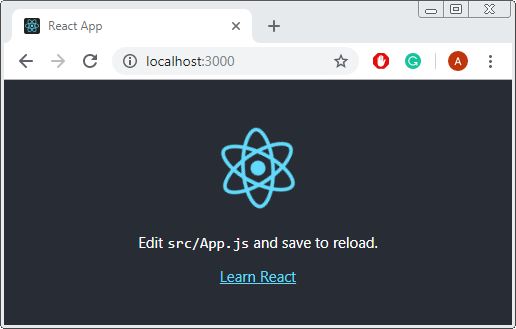
When the folder is created, navigate to it by typing:

cd example3

Next, type:

npm start

The above command launches the server, earlier the initial preparation for the project is finished, a new browser window will visible and display the URL **http://localhost:3000**and publish the placeholder content as shown below:



Now replace the placeholder content in the App.js file with a component who returns some simple HTML elements.

src\App.js

import React, { Component } from "react";

class App extends Component {

render(){

return (

<h1>Hello JSX</h1>

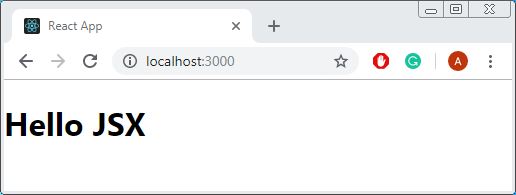
);

}

}

export default App;

In the render method, we specify the JSX that will be inserted into the DOM as HTML when the component's view is rendered.



Using JSX in Return Statement of Component

We add a new file in the **src** folder and call it as **Vehicles.js**, add the below content in file. We have created Vehicles component which returns valid JSX statement.

src\Vehicles.js

import React from "react";

export default function Vehicles() {

return(

<h1>

Car

</h1>

);

}

The above example shows both opening and closing paired tags in action. It is suggested to write HTML tags always in lowercase this helps to distinguishes them from custom JSX tags.

Now update **App.js**, first import our **Vehicles** Component and add to the template as shown below. It is the view component that controls our entire app or page.

src\App.js

import React, { Component } from "react";

import Vehicles from "./Vehicles";

class App extends Component {

render(){

return (

<Vehicles />

);

}

}

export default App;

When we use a function or class to create our components, we will always name them using capitalize and call them using the JSX syntax like below:

<Vehicles />

Using JavaScript Expressions in JSX

JSX will only accept expressions between curly braces. The curly braces tell our transpiler to process what is between the curly braces as normal JavaScript. Note we cannot write full statements as we might expect, only bits of JavaScript that return a value. Here are a few common types of expressions you may use:

1. Variable and object values
2. Function calls and references
3. Expressions and operators
4. Loops and iteration

Example 1

import React from "react";

const price = 2000;

export default function Vehicles() {

return(

<div>

Car Price: {price}

</div>

);

}

Example 2

import React from "react";

const price = 2000;

const tax = price \* 0.10;

export default function Vehicles() {

return(

<div>

Car Price: {price + tax}

</div>

);

}

Example 3

import React from "react";

const car = "MG Hector";

const specifications = {

length : 4655,

width : 1835,

height : 1760

}

const getDimensions = specifications => (

`${specifications.length}(mm) ${specifications.width}(mm) ${specifications.height}(mm)`

);

export default function Vehicles() {

return(

<div>

<p>{car}</p>

<p>{getDimensions(specifications)}</p>

</div>

);

}

Example 4

import React from "react";

const price = 8000;

export default function Vehicles() {

if (price < 5000) {

return <div>Car Price: {price + (price \* 0.05)}</div>

} else {

return <div>Car Price: {price + (price \* 0.15)}</div>

}

}

Example 5

import React from "react";

const cars = ["MG Hector", "Hyundai Venue", "Toyota Fortuner",

"Hyundai Creta"];

const carList = cars.map((car) =>

<li key={car.toString()}>{car}</li>

);

export default function Vehicles() {

return <ul>{carList}</ul>

}

Using Events in JSX

Expressions are used to tell React how to respond to events when they are triggered by an element. A list of **li** returned by the **Vehicles** component and used the **onClick** prop to tell React how to respond when the **click** event is triggered.

import React, { Component } from "react";

class Vehicles extends Component {

constructor(props) {

super(props)

this.list\_element = React.createRef()

this.state = {

items: [

{ text: 'MG Hector'},

{ text: 'Hyundai Venue'},

{ text: 'Toyota Fortuner'},

{ text: 'Hyundai Creta'}

]

}

}

doSomething(text, index) {

alert(text);

}

render() {

return (

<ol>

{this.state.items.map((item, index) => (

<li key={item.text} onClick={() => this.doSomething(item.text, index)}>

<span>{item.text}</span>

</li>

))}

</ol>

);

}

}

export default Vehicles;