What is React?

React, sometimes referred to as a frontend JavaScript framework, is a JavaScript library created by Facebook.

React is a tool for building UI components.

Create Your First Component

When creating a React component, the component's name *MUST* start with an upper case letter.

Class Component

A class component must include the `extends React.Component` statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a `render()` method, this method returns HTML.

Example

Create a Class component called `Car`

```
class Car extends React.Component {

  render() {

    return <h2>Hi, I am a Car!</h2>;

  }

}


ReactDOM.render(<Car />, document.getElementById('root'));
```

Note:-

Rendering a Component

React renders HTML to the web page by using a function called `ReactDOM.render()`.

The Render Function

The `ReactDOM.render()` 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.

Now your React application has a component called Car, which returns an `<h2>` element.

To use this component in your application, use similar syntax as normal HTML: `<Car />`

Example

Display the `Car` component in the "root" element :

```
ReactDOM.render(<Car />, document.getElementById('root'));
```

Function Component:-

Here is the same example as above, but created using a Function component instead.

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial.

Example

Create a Function component called `Car`

```
function Car() {

        return <h1>Hello World!</h1>;

      }


ReactDOM.render(<Car />, document.getElementById('root'));
```

Note:- Rendering a Component

React renders HTML to the web page by using a function called `ReactDOM.render()`.

The Render Function

The `ReactDOM.render()` 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.

Now your React application has a component called Car, which returns an `<h1>` element.

To use this component in your application, use similar syntax as normal HTML: `<Car />`

Example

Display the `Car` component in the "root" element :

```
ReactDOM.render(<Car />, document.getElementById('root'));
```

React Directly in HTML

The quickest way start learning React is to write React directly in your HTML files.

Start by including three scripts, the first two let us write React code in our JavaScripts, and the third, Babel, allows us to write JSX syntax and ES6 in older browsers.

Function Component example:-

Reactjsfucntion.html file :-

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>

    <div id="mydiv"></div>

    <script type="text/babel">
      function Car() {
        return <h1>Hello World!</h1>;
      }

      ReactDOM.render(<Car />, document.getElementById('mydiv'));
    </script>

  </body>
</html>
```

Output:-

Hello World!

Class Component example:-

Reactjsclass.html file code:-

```
<!DOCTYPE html>
<html>
  <head>
    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>

<div id="root"></div>

        <script type="text/babel">

class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}

ReactDOM.render(<Car />, document.getElementById('root'));
```

```
    </script>


  </body>

</html>
```

Output:-

Hi, I am a Car!

JSX which allows you to write HTML tags inside the JavaScript code:-

Example:-

Create a variable that contains HTML code and display it in the "root" node:

Reactjsvariable.html file code:-

```
<!DOCTYPE html>

<html>

  <head>

    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>

    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
crossorigin></script>

    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

  </head>
```

```
  <body>


<div id="root"></div>

        <script type="text/babel">


const myElement = (
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);

        </script>

  </body>
</html>
```

Output:-

- Apples
- Bananas
- Cherries

What is JSX?

JSX stands for JavaScript XML.

JSX allows us to write HTML in React.

JSX makes it easier to write and add HTML in React.

Example 1

JSX:

```
const myElement = <h1>I Love JSX!</h1>;


const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(myElement);
```

write above code inside reactjsjsx.html file & run in your browser:-

```
<!DOCTYPE html>

<html>
```

```
  <head>
    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
  </head>
  <body>

<div id="root"></div>

        <script type="text/babel">

const myElement = <h1>I Love JSX!</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);

        </script>

  </body>
</html>
```

Expressions in JSX

With JSX you can write expressions inside curly braces `{ }`.

The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result:

Example

Execute the expression `5 + 5`:

```
const myElement = <h1>React is {5 + 5} times better with JSX</h1>;
```

Reactjsexpression.html file code:-

```
<!DOCTYPE html>

<html>

 <head>

  <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>

  <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
crossorigin></script>

  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

 </head>

 <body>



<div id="root"></div>


        <script type="text/babel">
```

```
const myElement = <h1>React is {5 + 5} times better with JSX</h1>;

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(myElement);

        </script>

 </body>
</html>
```

Output:-

React is 10 times better with JSX

ust like HTML DOM events, React can perform actions based on user events.

React has the same events as HTML: click, change, mouseover etc.

Adding Events

React events are written in camelCase syntax:

`onClick` instead of `onclick`.

React event handlers are written inside curly braces:

`onClick={shoot}` instead of `onClick="shoot()"`.

React:

```
<button onClick={shoot}>Take the Shot!</button>
```

Example:

Put the `shoot` function inside the `Football` component:

```
function Football() {
  const shoot = () => {
    alert("Great Shot!");
  }

  return (
    <button onClick={shoot}>Take the shot!</button>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Football />);
```

write reactjsfootball.html file code for above example:-

```
<!DOCTYPE html>

<html>

  <head>

    <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>

    <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"
crossorigin></script>

    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

  </head>

  <body>


    <div id="root"></div>


    <script type="text/babel">

      function Football() {

  const shoot = () => {

    alert("Great Shot!");

  }


  return (

    <button onClick={shoot}>Take the shot!</button>

  );

}


const root = ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(<Football />);
  </script>


 </body>
</html>
```

Output:-