**What is React JS?**

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ‘V’ denotes the view in MVC. ReactJS is an open-source, component-based front end library responsible only for the view layer of the application. It is maintained by Facebook.

React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and React will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.  
A React application is made of multiple components, each responsible for rendering a small, reusable piece of HTML. Components can be nested within other components to allow complex applications to be built out of simple building blocks. A component may also maintain an internal state – for example, a TabList component may store a variable corresponding to the currently open tab.

**Note:** React is not a framework. It is just a library developed by Facebook to solve some problems that we were facing earlier.  
**Prerequisites**: Download Node packages with their latest version.

React implements a virtual DOM that is basically a DOM tree representation in JavaScript. So when it needs to read or write to the DOM, it will use the virtual representation of it. Then the virtual DOM will try to find the most efficient way to update the browser’s DOM.

“virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM. This process is called reconciliation.

React compares the Virtual DOM and pre-updated Virtual DOM and only marks the sub-tree of components that are updated. This process is called diffing.

**JSX**

But instead of using regular JavaScript, React code should be written in something called JSX.

const ele = <h1>This is sample JSX</h1>;

* It is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript.
* It makes it easier for us to create templates.
* Instead of separating the markup and logic in separated files, React uses *components* for this purpose.

**Using JavaScript expressions in JSX:**

In React we are allowed to use normal JavaScript expressions with JSX. To embed any JavaScript expression in a piece of code written in JSX we will have to wrap that expression in curly braces {}.

import React from 'react';

import ReactDOM from 'react-dom';

const name = "Learner";

const element = <h1>Hello, { name }.Welcome to GeeksforGeeks.< /h1>;

ReactDOM.render(

    element,

    document.getElementById("root"));

**Specifying attribute values**: JSX allows us to specify attribute values in two ways

1. **As for string literals:** We can specify the values of attributes as hard-coded strings using quotes:

const ele = <h1 className = "firstAttribute">Hello!</h1>;

**2. As expressions:** We can specify attributes as expressions using curly braces {}:

const ele = <h1 className = {varName}>Hello!</h1>;

**Wrapping elements or Children in JSX**

import React from 'react';

import ReactDOM from 'react-dom';

const element = <div>

                   <h1>This is Heading 1 < /h1>

                   <h2>This is Heading 2</h2 >

                   <h3>This is Heading 3 < /h3>

                </div > ;

ReactDOM.render(

    element,

    document.getElementById("root"));

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

A **Component** is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.

**Functional Components**: Functional components are simply javascript functions. We can create a functional component in React by writing a javascript function. These functions may or may not receive data as parameters

const Component=()=>

{

return <h1>Hello</h1>;

}

**ECMAScript:** It is the specification defined in ECMA-262 for creating a general purpose scripting language.In simple terms it is a standardization for creating a scripting language.It was introduced by Ecma International, and is basically an implementation with which we learn how to create a scripting language.   
 **Javascript:** A general purpose scripting language that conforms to the ECMAScript specification.It is basically an implementation which tells us how to use a scripting language.

**ES6**

Javascript ES6 has been around for a few years now, and it allows us to write code in a clever way which basically makes the code more modern and more readable. It’s fair to say that with the use of ES6 features we write less and do more, hence the term ‘write less, do more’ definitely suits ES6.   
ES6 introduced several key features like const, let, arrow functions, template literals, default parameters, and a lot more

**const & let**

**Arrow functions**

Arrow functions(also known as ‘fat arrow functions’) are a more concise syntax for writing function expressions.Introduced in ES6, arrow functions are definitely one of the most impactful changes in javascript. These function expressions makes your code more readable, more modern

**Template literal**

`Hello ${name}`

**Object and Array Destructing**

**Default Parameters**

**Rest parameter and spread operator**

* Both the spread and the rest operator make use of triple dots (…), and sometimes it’s hard to differentiate which one is rest or spread. Simply remember that:

When … is at the end of function parameter, it’s rest parameter.

* When … occurs in function call or alike, its called a spread operator

**Props**

React allows us to pass information to a Component using something called **props** (stands for properties). Props are basically kind of global variable or object

**State**

The state is an instance of React Component Class can be defined as an object of a set of **observable** properties that control the behavior of the component. In other words, the State of a component is an object that holds some information that may change over the lifetime of the component