**React**

React is a popular JavaScript library for building user interfaces. It was developed by Facebook and has gained widespread adoption in the web development community due to its simplicity, flexibility, and performance.

React key concepts:

Components: React applications are built using components, which are reusable and self-contained pieces of code responsible for rendering a part of the UI. Components can be simple, like a button, or complex, like an entire form or page.

Virtual DOM: React uses a virtual DOM to efficiently update the UI. Instead of directly manipulating the browser's DOM, React creates a virtual representation of the DOM in memory and updates it when the state of a component changes. It then calculates the difference between the virtual DOM and the actual DOM and updates only the necessary parts, which results in faster rendering.

JSX: React introduces JSX, a syntax extension that allows you to write HTML-like code within JavaScript. JSX makes it easier to write and understand React components by combining HTML structure with JavaScript logic.

State and Props: React components can have two types of data: state and props. State represents the internal state of a component and can change over time, while props are immutable data passed from a parent component to a child component.

Unidirectional Data Flow: React follows a unidirectional data flow, where data flows down from parent components to child components via props. This helps in maintaining a clear and predictable data flow throughout the application.

Lifecycle Methods: React components have lifecycle methods that allow you to hook into various stages of a component's lifecycle, such as when it is initialized, rendered, updated, or unmounted. These methods provide opportunities to perform tasks like fetching data, updating the DOM, or cleaning up resources.

React Hooks:

Introduced in React 16.8, hooks are functions that allow you to use state and other React features in functional components without writing a class.

Hooks provide a way to reuse stateful logic between components and promote a more functional style of component composition.

Creating React app:

Using Create React App to set up a new React project with a pre-configured build environment. We can create a new React project by running:

Syntax:

npx create-react-app my-app

cd my-app

npm start

Writing React Components:

Functional Component:

Syntax:

const Name = () = >{

return(

<div> </div>

);

}

Example:

import React from 'react';

const MyComponent = () => {

return (

<div>

<h1>Hello, World!</h1>

</div>

);

}

export default MyComponent;

Class Component:

Syntax:

class Name{

Render(){

return(

<div> </div>

);

}

}

Example:

import React, { Component } from 'react';

class MyComponent extends Component {

render() {

return (

<div>

<h1>Hello, World!</h1>

</div>

);

}

}

export default MyComponent;