

# Javascript

- **Introduction to Javascript**

- What is JS
- Purpose of JS
- Features of JS
- Program
- Execution

- **Installation of NodeJS**

- **Variables**

- Declaration
- Assignment
- Initialization
- Scope Statement
- Var
- Let
- Const

- **Data Types**

- Number
- Boolean
- String
- Null
- Undefined
- BigInt
- Symbol
- Object

- **Functions**

- What is Function
- Purpose of Function
- Syntax to create Function

- Function Definition
- Function Scope
- Function Block
- Function Call
- Types of Function
  - Normal function
  - Parameters
  - Arguments
  - Return
  - Callback
  - Function Expression
  - Anonymous
  - Arrow function
  - Async Function
  - Higher Order function
- Syntax
- Purpose
- Examples
- **Object**
  - What is Object
  - Purpose
  - Creation of Object
  - Properties
  - Crud Operation on Object
    - Read
    - Insert
    - Update
    - Delete
- **Array**
  - Syntax
  - Purpose
  - Literal Notation

- Index
- **Array Methods**
  - Push
  - Pop
  - Shift
  - Unshift
  - ForEach
  - Map
  - Filter
  - Splice
  - Slice
  - Includes
  - indexOf
- **Selection Statement and Loops**
  - If
  - Else
  - Else if
  - Switch
  - For
  - For of
  - For in
- **Spread**
  - How to copy properties from one object into another object.
  - How to copy elements from one array into another array.
- **Rest**
  - Rules of parameter
  - Order of parameter
- **Destructuring**
  - Object
  - Array
- **Clousers**

- What is Closures
- How to create it
- Syntax
- Purpose
- **Promises**
  - Creation of Promise
  - Resolve
  - Reject
  - States of Promise
    - Pending
    - Rejected
    - Fulfilled
  - Accessing the data from Promise
    - Then catch
    - Async Await
    - Try catch
- **DOM and DOM Manipulation**
  - What is DOM
  - Dom Objects
  - How to access the DOM Object
  - How to manipulate HTML with DOM
  - getElementById
  - getElementsByClass
  - QuerySelector
  - innerText
  - Appnedchild
- **CreateElement**
  - How to create a DOM Element
  - How to add content in it
  - How to add DOM element in the DOM
  - How to remove Dom Element

- **Module Concept in JS**

- What is module concept
- Why we use it
- Named export
- Default export
- Importing named export
- Importing default export

## **React JS**

### **Introduction to React JS**

- What is React JS
- Purpose of React JS
- Why do we required React JS
- Features of React JS
- Drawback of HTML and JS to create UI

### **React Elements**

- What is React Element
- How to create React Element
- Integration of HTML and React
- How to add Inline, Internal and external CSS
- Detailed Understanding About the React.createElement()
- How to create User Interface with React Element

### **ReactDOM**

- What is React DOM
- How to integrate React DOM with HTML
- How ReactDOM is used to add React Element in the DOM
- Understanding of ReactDOM.render()
- Virtual DOM

- How Virtual DOM works

## **JSX**

- What is JSX
- Syntax of JSX
- How JSX is different from HTML
- Rules of JSX
- Integrating babel with Html
- Creating UI with JSX
- Advantages of JSX
- How JSX simplifies Creation of UI in React JS
- What is babel
- Integration of babel with HTML

## **React Components**

- What is React Component
- Advantage of React Component
- How to create React Component
- Types of React Component
- Introduction to Functional Component
- Creating the Functional Component
- Introduction to Class Component
- Creating the class Components
- How components can be used for Reusability

## **Props**

- What is Props
- Purpose of the props
- How to use Props in Functional Component and Class Component
- How to pass Props

- Access the Props
- Pass the different types of data as a props

## **Vite Tool**

- What is Vite Tool
- How Vite is used to create Basic React Application
- How to run and stop React Application
- Accessing the React Application
- Understanding the Folder Structure of React Application
- NPM
- What is Node Package Manager
- How to install different Packages using NPM

## **Functional Components**

- Understanding of Functional Component in Detail
- How to create Functional Component
- Why Functional Components are used over class Components
- How to render the Functional Component
- Sequence of Calling the Functional Component
- Flow of React Application

## **Introduction to Hooks**

- Introduction to the Hooks
- Why hooks are introduced
- Rules of using the hooks
- How to import the Hooks and use it
- Listing the important hooks

## **State and setState**

- What is State
- Why do we required state

- How state can be used to create Dynamic User Interface
- Creation of State
- Introduction to first hook useState()
- Understanding of useState()
- How setState() is used and purpose of it
- Understanding in depth of setState and its Working
- Implementing the Counter App
- Implementing the Dynamic Card with Dark and Light Theme
- Implement Theme feature

### **Rendering the List using Map()**

- What is map() in JS
- How it Works
- Understanding in details about map()
- How map() used to create UI
- Iterating Through Map
- Keys and List.

### **Axios**

- What is Axios
- Installing and Integrating Axios with React App
- How to do get() Request with Axios
- Handling the Promise with then and catch
- Handling the Promise with async await

### **Interaction between Components**

- Understanding the Relationship between Components
- Parent Child Relation
- Sharing the Data From Parent Component to Child Component using Props
- Props Drilling
- Problems With Props Drilling



- Introduction to Context

## **Context API**

- Introduction to Context API
- Why Context API
- How Context API solves the Problems of Props Drilling
- Limitation of Context API
- How to Create the Context
- How to access Provider Component
- Understanding of Provider Component
- Storing the Data in Context
- How to make Available the context data to Child Components
- useContext() hook
- Purpose of useContext() hook
- How to access data from context using the useContext() hook

## **React Routing**

- What is Routing
- How to implement routing in React App
- Installing and Configuring the react-router-dom
- BrowserRouter
- Routes
- Route
- Link

## **useRef() hook**

- What is useRef() hook
- How it works
- Purpose of the useRef() hook
- Syntax of useRef() hook
- How useRef() hook used to manage the data or store the data
- Difference between useRef() hook and useState() hook
- DOM Manipulation using useRef() hook

## **useEffect() hook (Recording will be Provided)**

- What are sideEffects in the React
- Pure functions in JS
- Lifecycle of Components
- Phases of Lifecycle
- What is Mounting and Unmounting
- Mount Phase
- Unmount Phase
- Update Phase
- How useEffect can be used to perform sideEffects in Different Phases of component
- Understanding How useEffect() hook works

## **useReducer() hook**

- Understanding of useReducer() hook
- How to manage complex state operation in the reducer
- Reducer
- Dispatch
- Action object
- Types
- Difference between useState() and useReducer() hook

## **Lazy Loading**

- What is Lazy Loading
- Benefits of Lazy Loading
- How to Lazy Loading will improve Performance
- Implementation of Lazy Loading

## **Redux with Functional Component**

- What is Redux
- Why Redux
- How redux will help in state management

- Installing and Configuring redux
- Store
- Dispatch
- Reducer
- How to combine Multiple reducers
- Configuring Reducers with Redux Store
- Action
- ActionCreator
- Action Types
- Redux Pattern
- useSelector() hook
- useDispatch() hook
- Implementing Redux in React Application