**https://www.javatpoint.com/react-hooks**

**What Is JSX:**

JSX is an Extension of JavaScript. JSX stands for JavaScript XML. JSX allows us to write HTML in React

**ReactJS Reconciliation:**

## **React Components**

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions but work in isolation and return HTML.

Components come in two types, Class components, and Function components, in this tutorial we will concentrate on Function components.

### **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.

### **Function Component**

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

## **React Props**

React Props are like function arguments in JavaScript and attributes in HTML.

To send props into a component, use the same syntax as HTML attributes:

# React State

The state is an updatable structure that is used to contain data or information about the component. The state in a component can change over time

# React Props Validation

Props are an important mechanism for passing the **read-only** attributes to React components. The props are usually required to use correctly in the component. If it is not used correctly, the components may not behave as expected. Hence, it is required to use **props validation** in improving react components.

**App.propTypes**

|  |  |
| --- | --- |
| PropTypes.any | The props can be of any data type. |
| **2.** | PropTypes.array | The props should be an array. |
| **3.** | PropTypes.bool | The props should be a boolean. |
| **4.** | PropTypes.func | The props should be a function. |
| **5.** | PropTypes.number | The props should be a number. |
| **6.** | PropTypes.object | The props should be an object. |
| **7.** | PropTypes.string | The props should be a string. |
| **8.** | PropTypes.symbol | The props should be a symbol. |

# React Component Life-Cycle

1. Initial Phase
2. Mounting Phase
3. Updating Phase
4. Unmounting Phase
5. <https://www.javatpoint.com/react-component-life-cycle>

# React Controlled Vs. Uncontrolled Component

## **Controlled Component**

A controlled component is bound to a value, and its changes will be handled in code by using **event-based callbacks**. Here, the input form element is handled by the react itself rather than the DOM

Controlled components have functions that govern the data passing into them on every **onChange** event occurs. This data is then saved to state and updated with setState() metho

## **Uncontrolled Component**

It is similar to the traditional HTML form inputs. Here, the form data is handled by the DOM itself. It maintains their own state and will be updated when the input value changes. To write an uncontrolled component, there is no need to write an event handler for every state update, and you can use a ref to access the value of the form from the DOM.

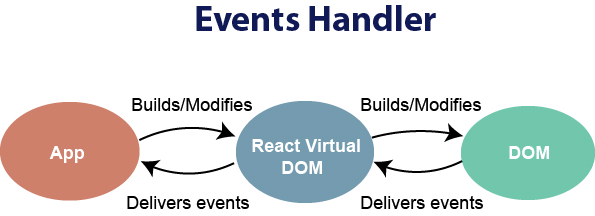
## **Difference table between controlled and uncontrolled component**

|  |  |  |
| --- | --- | --- |
| **SN** | **Controlled** | **Uncontrolled** |
| **1.** | It does not maintain its internal state. | It maintains its internal states. |
| **2.** | Here, data is controlled by the parent component. | Here, data is controlled by the DOM itself. |
| **3.** | It accepts its current value as a prop. | It uses a ref for their current values. |
| **4.** | It allows validation control. | It does not allow validation control. |
| **5.** | It has better control over the form elements and data. | It has limited control over the form elements and data. |

# React Events

An event is an action that could be triggered as a result of the user action or system generated event. For example, a mouse click, loading of a web page, pressing a key, window resizes, and other interactions are called events.

React has its own event handling system which is very similar to handling events on DOM elements. The react event handling system is known as Synthetic Events. The synthetic event is a cross-browser wrapper of the browser's native event.



# React Keys

A key is a unique identifier. In React, it is used to identify which items have changed, updated, or deleted from the Lists

# React Refs

Refs is the shorthand used for **references** in React. It is similar to **keys** in React. It is an attribute which makes it possible to store a reference to particular DOM nodes or React elements

<https://www.javatpoint.com/react-refs>

# React Fragments

his render method can return **single** elements or **multiple** elements

<React.Fragment>  </React.Fragment>

<>   </>

## **React Router Installation**

React contains three different packages for routing. These are:

1. **react-router:** It provides the core routing components and functions for the React Router applications.
2. **react-router-native:** It is used for mobile applications.
3. **react-router-dom:** It is used for web applications design.

## **Components in React Router**

There are two types of router components:

* **<BrowserRouter>:** It is used for handling the dynamic URL.
* **<HashRouter>:** It is used for handling the static request.

# React Higher-Order Components

# React Code Splitting

The React app bundled their files using tools like **Webpack** or **Browserfy**. Bundling is a process which takes multiple files and merges them into a single file, which is called a **bundle**. The bundle is responsible for loading an entire app at once on the webpage. We can understand it from the below example.

1. **import** { add } from './math.js';
2. console.log(add(16, 26)); // 42

# React Context

Context allows passing data through the component tree without passing props down manually at every level.

In React application, we passed data in a top-down approach via props. Sometimes it is inconvenient for certain types of props that are required by many components in the React application. Context provides a way to pass values between components without explicitly passing a prop through every level of the component tree.

# React Hooks

Hooks are a new addition in React 16. They let you use React features such as state and lifecycle methods. And that allows in functional components

There are 3 rules for hooks:

* Hooks can only be called inside React function components.
* Hooks can only be called at the top level of a component.
* Hooks cannot be conditional.
* [Basic Hooks](https://legacy.reactjs.org/docs/hooks-reference.html#basic-hooks)
  + [useState](https://legacy.reactjs.org/docs/hooks-reference.html#usestate) : The React useState Hook allows us to track state in a function component.
  + [useEffect](https://legacy.reactjs.org/docs/hooks-reference.html#useeffect) : he useEffect Hook allows you to perform side effects in your components.

Some examples of side effects are: fetching data, directly updating the DOM, and timers.

useEffect accepts two arguments. The second argument is optional.

useEffect(<function>, <dependency>)

* + [useContext](https://legacy.reactjs.org/docs/hooks-reference.html#usecontext) : React Context is a way to manage state globally.
* [Additional Hooks](https://legacy.reactjs.org/docs/hooks-reference.html#additional-hooks)
  + [useReducer](https://legacy.reactjs.org/docs/hooks-reference.html#usereducer) : The useReducer Hook is similar to the useState Hook.

It allows for custom state logic

It accept useReducer(<reducer>, <initialState>)

* + [useCallback](https://legacy.reactjs.org/docs/hooks-reference.html#usecallback) : The React useCallback Hook returns a memoized callback function.
  + [useMemo](https://legacy.reactjs.org/docs/hooks-reference.html#usememo)

The useCallback and useMemo Hooks are similar. The main difference is that useMemo returns a memoized value and useCallback returns a memoized function

* + [useRef](https://legacy.reactjs.org/docs/hooks-reference.html#useref) : The useRef Hook allows you to persist values between renders.

It can be used to store a mutable value that does not cause a re-render when updated.

It can be used to access a DOM element directly.

useRef() only returns one item. It returns an Object called current.

When we initialize useRef we set the initial value: useRef(0)

1. React.createContext
2. Context.provider
3. Context.Consumer
4. Class.contextType

## **Pre-requisites for React Hooks**

1. Node version 6 or above
2. NPM version 5.2 or above
3. Create-react-app tool for running the React App

## **What is Sass**

Sass is a CSS pre-processor.

Sass files are executed on the server and sends CSS to the browser.

You can learn more about Sass in our [Sass Tutorial](https://www.w3schools.com/sass/default.php).

### React.PureComponent

React.PureComponent is similar to [React.Component](https://legacy.reactjs.org/docs/react-api.html#reactcomponent). The difference between them is that [React.Component](https://legacy.reactjs.org/docs/react-api.html#reactcomponent) doesn’t implement [shouldComponentUpdate()](https://legacy.reactjs.org/docs/react-component.html#shouldcomponentupdate), but React.PureComponent implements it with a shallow prop and state comparison.

**Difference between ES5 and ES6 :**

| **SR.NO.** | **ES5** | **ES6** |
| --- | --- | --- |
| 1. | ECMA script is a trademarked scripting language specification defined by Ecma international. The fifth edition of the  same is known as ES5 | ECMA script is a trademarked scripting language specification defined by Ecma international. The sixth edition of the same is known as ES6 |
| 2. | It was introduced in 2009. | It was introduced in 2015. |
| 3. | It supports primitive data types that are string, number, boolean, null, and undefined. | In ES6, there are some additions to JavaScript data types. It introduced a new primitive data type ‘symbol’ for supporting unique values. |
| 4. | There are only one way to define the variables by using the var keyword. | There are two new ways to define variables that are let and const. |
| 5. | It has a lower performance as compared to ES6. | It has a higher performance than ES5. |
| 6. | Object manipulation is time-consuming in ES5. | Object manipulation is less time-consuming in ES6. |
| 7. | In ES5, both function and return keywords are used to define a function. | An arrow function is a new feature introduced in ES6 by which we don’t require the function keyword to define the function. |
| 8. | It provides a larger range of community supports than that of ES6 | It provides a less range of community supports than that of ES5 |

## **Form Validation Library**

React developers can use Formik to validate forms and display helpful error messages if needed.

# What are Promises in React.js?

In React, there are various ways to work with asynchronous data, but using Promises is one of the most popular ones. A Promise is an object that represents an asynchronous operation. It stands for a value that will be resolved in the future

Promises allow you to perform asynchronous operations in JavaScript. To construct a Promise from scratch, you can use the Promise constructor. This takes a function which takes two parameters: “resolve”, a function to call when the operation completes, and “reject”, a function to call if the operation fails. You then have to call one of these functions when your operation completes.

JavaScript gives you two methods of handling the result of a promise, the first being ***.then()***, a function that takes a function as a parameter, which it will pass the resolved value to.

# Progressive Web App

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://cra.link/PWA

serviceWorkerRegistration.unregister();

As the comment states, switching serviceWorker.unregister() to serviceWorker.register() will opt you in to using the service worker.

## Why Opt-in?

Offline-first Progressive Web Apps are faster and more reliable than traditional web pages, and provide an engaging mobile experience:

* All static site assets that are a part of your webpack build are cached so that your page loads fast on subsequent visits, regardless of network connectivity (such as 2G or 3G). Updates are downloaded in the background.
* Your app will work regardless of network state, even if offline. This means your users will be able to use your app at 10,000 feet and on the subway.
* On mobile devices, your app can be added directly to the user's home screen, app icon and all. This eliminates the need for the app store.

# react-indexed-db-hook

react-indexed-db-hook is forked from [**react-indexed-db**](https://github.com/assuncaocharles/react-indexed-db). This library is a wrapper around the browser's IndexedDB database in an "easier to use" React Hook.

npm install react-indexed-db-hook

#### **What are breadcrumbs used for in React Router?**

Breadcrumbs are typically used as a navigational aid, allowing users to keep track of their location within a website or application. In React Router, breadcrumbs can be used to dynamically generate a trail of links based on the user’s current location. This can be especially helpful in large applications with complex routing.

#### **What are named components in React Router?**

Named components are simply React components that have been given a name attribute. This is used by React Router to automatically match up the component with the corresponding route.