**REACT JS (library)**

* It is a js library to build SPA,also it is used for serverside rendering,used to create mobile native application using react native
* It is a declarative approach
* It is a component based architecture
* It increase the app perfomance

### **Declarative**

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Declarative views make your code more predictable and easier to debug.

### **Component-Based**

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.

### **Learn Once, Write Anywhere**

We don’t make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

React can also render on the server using Node and power mobile apps using [React Native](https://reactnative.dev/).

**Install command**

**Npm -g create-react-app**

**ALTERNATIVES**

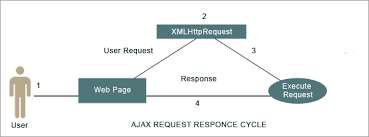
* Ember js
* Angular js(2009)=>Invented by google,framework to create SPA
* Vue js

**AJAX(2001)**

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

* Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
* Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
* With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
* XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
* AJAX is a web browser technology independent of web server software.
* A user can continue to use the application while the client program requests information from the server in the background.
* Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
* Data-driven as opposed to page-driven.

**AJAX REQUEST RESPONSE CYCLE**



**NOTE : -**

If you get **xmlhttp (XHR)** request or response in your app it is ajax request

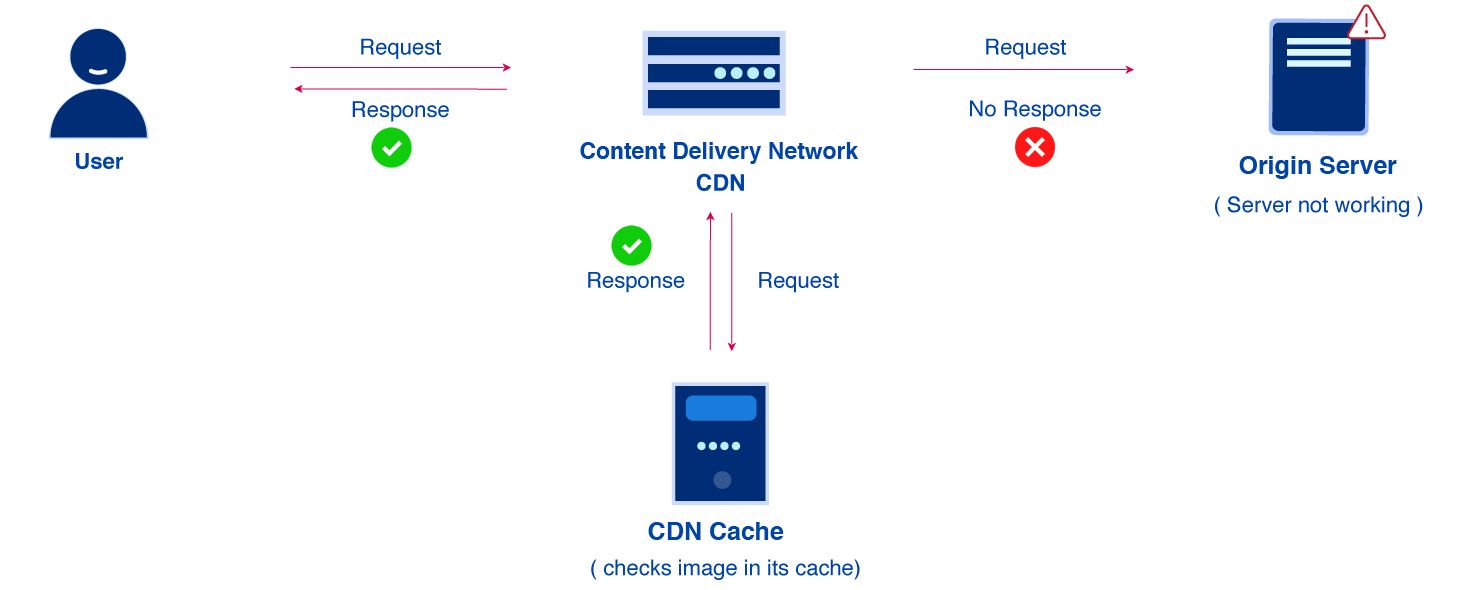
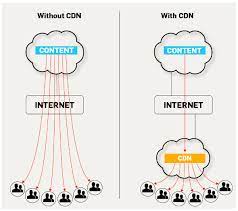
**What is XMLHttpRequest?**

XMLHttpRequest (XHR) objects are used to interact with servers. You can retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just part of a page without disrupting what the user is doing.

* XMLHttpRequest is used heavily in [AJAX](https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX) programming.
* XMLHttpRequest can be used to retrieve any type of data, not just XML.

**WHAT IS CDN(content delivery network** **)?**

A CDN is **a network of servers that distributes content from an “origin” server throughout the world by caching content close to where each end user is accessing the internet via a web-enabled device**. The content they request is first stored on the origin server and is then replicated and stored elsewhere as needed.



**Here's a list of the top CDN providers to get you started.**

* Cloudflare.
* CloudFront.
* Fastly.
* CDNetworks.
* Incapsula.
* Akamai.
* StackPath.

**WHAT IS npx?**

NPX: The npx stands **for Node Package Execute** and it comes with the npm, when you installed npm above 5.2.0 version then automatically npx will installed. It is an npm package runner that can execute any package that you want from the npm registry without even installing that package.0

**npx naming restriction : -**

Some rules:

* The name must be less than or equal to 214 characters. This includes the scope for scoped packages.
* The names of scoped packages can begin with a dot or an underscore. This is not permitted without a scope.
* New packages must not have uppercase letters in the name.
* The name ends up being part of a URL, an argument on the command line, and a folder name. Therefore, the name can't contain any non-URL-safe characters.
* The names should not start with **react**

**WHAT IS REACT SCRIPTS?**

react-scripts is **a set of scripts from the create-react-app starter pack**. create-react-app helps you kick off projects without configuring, so you do not have to setup your project by yourself. react-scripts start sets up the development environment and starts a server, as well as hot module reloading.

**WHAT IS Hot Module Replacement?**

Hot Module Replacement (HMR) exchanges, adds, or removes [modules](https://webpack.js.org/concepts/modules/) while an application is running, without a full reload. This can significantly speed up development in a few ways:

* Retain application state which is lost during a full reload.
* Save valuable development time by only updating what's changed.
* Instantly update the browser when modifications are made to CSS/JS in the source code, which is almost comparable to changing styles directly in the browser's dev tools.

**What is build in react JS?**

The command line npm run build creates the index. html file, and the corresponding javascript is all bundled into one minified js file, all placed in a single folder, 'build'. – Paulos3000.

**What is the build folder in react?**

**Commend : npm run build**

When we run **npm run build** , we see output in /build folder. Builds the **app for production** to the build folder. It correctly bundles React in production mode and optimizes the build for the best performance. The build is minified and the filenames include the hashes

**What is jest?**

**Commend :** **Npm run test**

Jest is a JavaScript testing framework maintained by Facebook, Inc., designed and built by Christoph Nakazawa with a focus on simplicity and support for large web applications. It works with projects using Babel, TypeScript, Node.js, React, Angular, Vue.js and Svelte.

**What is NPM eject?**

**Commend :** **npm run eject ​**

This command will **remove the single build dependency** from your project. Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc.) into your project as dependencies in package

**what is eslint used for ?**

ESLint is **a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code**, with the goal of making code more consistent and avoiding bugs. ... ESLint uses an AST to evaluate patterns in code. ESLint is completely pluggable, every single rule is a plugin and you can add more at runtime.

**VDOM(virtual DOM) :-**

The virtual DOM (VDOM) is a programming concept where **an ideal, or “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**. ... They may also be considered a part of “virtual DOM” implementation in React.

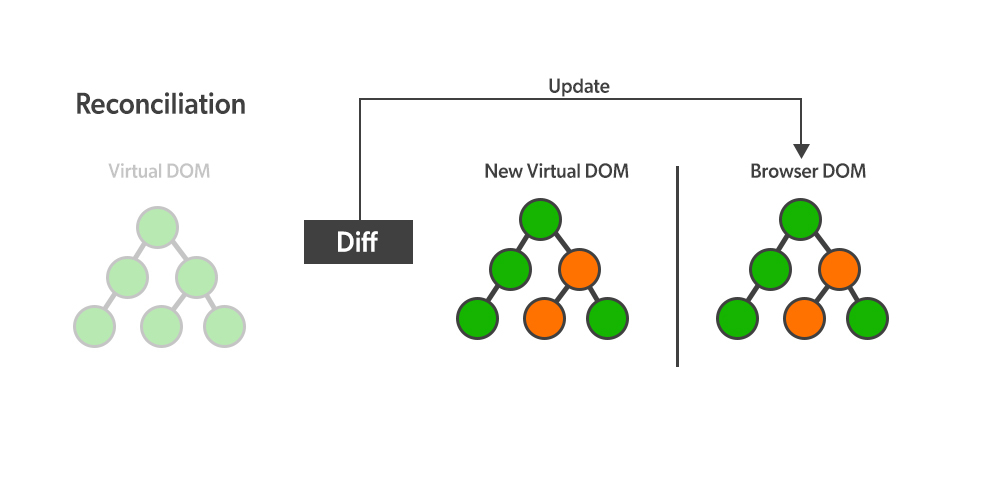
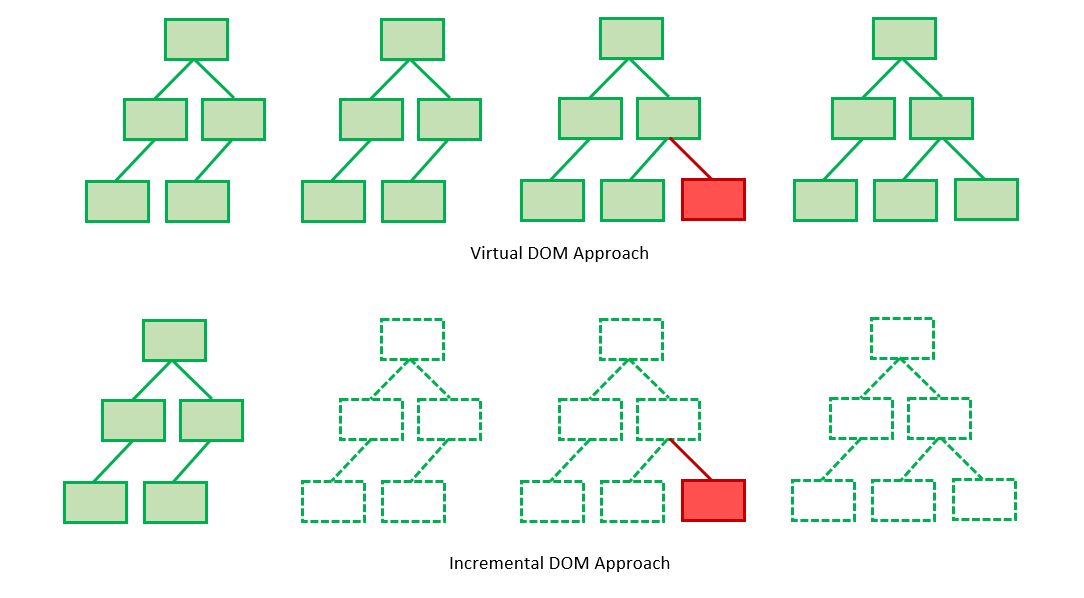
* It has two phase

1. Render phase
2. Commit phase

**NOTE : -** It works on diffing algorithm(finding difference between DOM and VDOM)

**WHAT IS DIFFING ALGORITHM ?**

* The **Diffing Algorithm** ... When diffing two trees, **React** first compares the two root elements. The behavior is different depending on the types of the root element
* Diffing is **a heuristic algorithm based on two assumptions**: Two elements of different types will produce different trees. The developer can hint at what elements will remain stable across renders with a key prop



**Ways to create html element**

|  |  |
| --- | --- |
| html | <div id=”demo”>  <h1 class=”h1>  Hello h1  </h1>  <p>  I am project  </p>  <button class=”btn”>  Click me  </button>  </div> |
| dom | let div = document.createElement("div");  let h1 = document.createElement("h1");  let p = document.createElement("p");  let btn = document.createElement("button");  div.className = "div";  h1.className = "h1";  p.className = "p";  btn.className = "btn";  h1.innerHTML = "i am h1";  p.innerHTML = "i am para";  btn.innerHTML = "clickme";  div.appendChild(h1);  div.appendChild(p);  div.appendChild(btn);  document.body.appendChild(div) |
| React  React way continue | import React from "react";  import ReactDOM from "react-dom";  let element = React.createElement(    "div",    {      id: "demo",    },    React.createElement(      "h1",      {        classname: "h1",      },      "i am h1"    ),    React.createElement("p", null, "i am para"),    React.createElement(      "button",      {        classname: "btn",      },      "clickme"    )  );  ReactDOM.render(element, document.getElementById("root")); |
| JSX | import React from "react";  import ReactDOM from "react-dom";  let element = (    <div id="demo">      <h1 className="h1">iam h1</h1>      <p>i am para</p>      <button>clickme</button>    </div>  );  ReactDOM.render(element, document.getElementById("root")); |

**WHAT IS JSX ?**

This funny tag syntax is neither a string nor HTML.

It is called JSX, and it is a syntax extension to JavaScript**. We recommend using it with React to describe what the UI should look like**. JSX may remind you of a template language, but it comes with the full power of JavaScript.

React [doesn’t require](https://reactjs.org/docs/react-without-jsx.html) using JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages.

**RULES OF JSX**

* JSX follows XML rules so Elements should be closed ,even it is selfclosing tag  
  eg :- <p>hello =>wrong  
   <p>hello</p> =>correct  
    
   <img/> =>wrong  
   <img></img> =>correct
* Same level elements should be wrapped with one element or fragments(<></>) **eg:-**  
   <p>…………….</p>  
   <h1>///////////</h1> =>wrong

<>  
 <P>…………</p>  
 <h1>////</h1>   
 </> =>correct  
  
  
**NOTE = fragment is for developer usage not for end user(fragments was introduced in version 16.8)**

* If you have a multiple lines of jsx enclose it with a paranthesis  
    
  **eg:-** (  
   <>  
   <p>hi</hi>  
   <p>hello</p>  
   </>  
   )
* Syntax for jsx expression={ }  
    
  The expression can be a React variable, or property, or any other valid JavaScript expression. JSX will execute the expression and return the result:  
    
    
    
   **eg:-** (  
   <h1>{var\_name}</h1>  
   )
* The class attribute is a much used attribute in HTML, but since JSX is rendered as JavaScript, and the class keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.  
  Use attribute className instead.
* Same way instead of label FOR attribute use HTMLFOR
* If you have any hifen(-) you should use camelcase  
  **eg :-** aria-Label

**COMPONENTS**

Components are reusable UI building blocks.

Components are **independent and reusable bits of code**. They serve the same purpose as JavaScript functions, but work in isolation and return JSX. Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

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.   
You can see in the below image we have broken down the UI of GeeksforGeeks’s homepage into individual components

There are two types of components : -

* **ES6 class ==🡺 class based components**
* **Js function == 🡺 functional based components**

**PROPS(properties)**

* The communication(sending) of data from parent component to child component is called **PROPS**
* Props are unidirectional which means parent to child not child to parent
* Props are immutable (read only) so we cannot modify the props from inside the component.

React Props

* It is **an object which stores the value of attributes of a tag and work similar to the HTML attributes**. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.
* React Props are like function arguments in JavaScript and attributes in HTML
* It is **an object which stores the value of attributes of a tag and work similar to the HTML attributes**. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function.

***PROPS.CHILDREN***

Essentially, props. children is **a special prop, automatically passed to every component, that can be used to render the content included between the opening and closing tags when invoking a component**. These kinds of components are identified by the official documentation as “boxes”.

**DEFAULT PROPS**

The defaultProps is **a React component property that allows you to set default values for the props argument**. If the prop property is passed, it will be changed. The defaultProps can be defined as a property on the component class itself, to set the default props for the class.

**What Are PropTypes In React?**

PropTypes are a mechanism to ensure that components use the correct data type and pass the right data, and that components use the right type of props, and that receiving components receive the right type of props

**WHAT IS PROPS DRILLING?**

Prop Drilling is **the process by which you pass data from one part of the React Component tree to another by going through other parts that do not need the data but only help in passing it around** . Imagine someone living in Lagos, Nigeria placing an order on Amazon for a package.

**STATE**

* React components has a built-in state object. The state object is **where you store property values that belongs to the component**. When the state object changes, the component re-renders.
* 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.

**REACT HOOKS :-)**

* React hooks are used to make stateless components into statefull components with the help of (useState)
* React hooks can be only used in functional based components
* After react 16.8 onwords react hooks were introdeced
* Hooks are functions
* It cannot be used in Class Based Components
* Hooks cannot be used in if else,for,functions

Important question

1. Difference between functional based comp and cbc
2. Stateless and statefull comp
3. Porps and state
4. Vdom made up of \_\_\_\_\_ele ,realdom made up of \_\_\_\_\_\_\_
5. Define prop,state,jsx
6. Rules of jsx
7. Dev server name which is installed during react app initialition
8. What dose the installation of react js liberary adds
9. Difference between react and reactdom liberary
10. Why cannot the browser read jsx
11. Define ReactDOM.render()
12. What dose the jsx returns
13. Different ways in react to create the vDom elements
14. What Is the base class in react
15. Define react.component and explain why it used
16. Difference b/w npm and npx
17. Define props drilling
18. How do you overcome props drilling
19. Difference between element element and components
20. Explain react workflow
21. Explain how Vdom works
22. What is the difference b/w dom and vdom
23. Difference b/w react and angular
24. What makes vdom faster
25. Define diffing algorithm
26. What is reconcilitation
27. Version of react
28. What is fragments
29. How many outermost elements can be have in the jsx expression
30. What is one of core types in react
31. In reactjs why there is a need to capitalize the components
32. Is it possible to nest jsx elements into other jsx elements
33. Difference b/w import and export

**React Lifecycle**

It has four phases they are : -

* Mounting(only one time)
* Updating(n number of time)
* Unmounting(only one time)
* Error boundries(n number of time)

**Mounting phase(methods)**

* Constructor(props)
* Static getDerivedStateFromProps(props,state)
* Render()
* componentDidMount()

**Update phase (methods)**

* Static getDerivedStateFromProps(props,state)
* shouldComponentUpdate()
* render()
* getSnapShotBeforeUpdate(prevProps,prevState)
* componentDidUpdate(prevProps,prevState,snapshot)

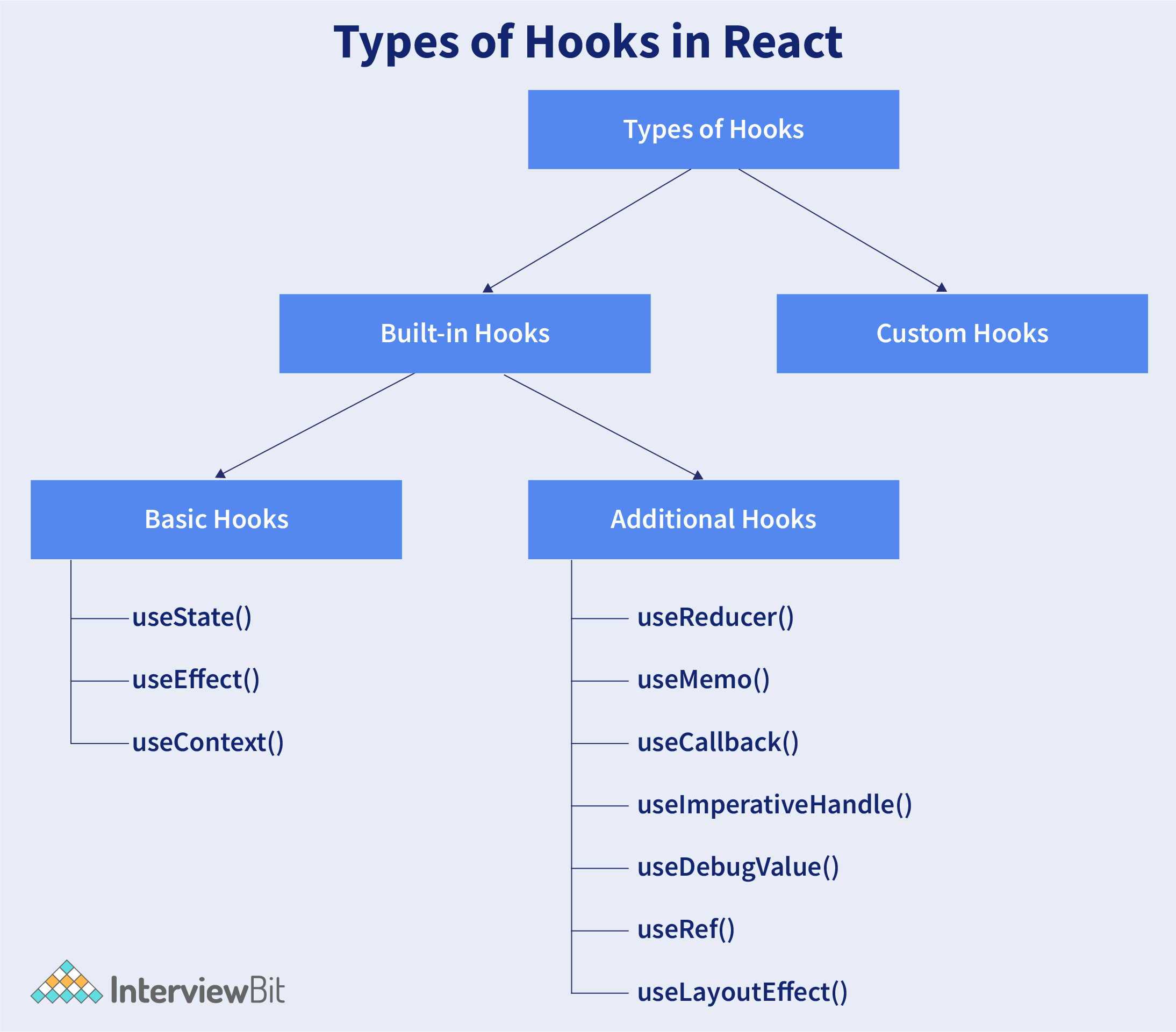
**Unmount phase(methods)**

* componentWillUnmount()

**What Are React Hooks?**

These are in-built functions that allow developers to use state and lifecycle methods within components in React. Each component’s lifecycle has 3 phases which are mount, unmount, and update. Alongside that, components have states and properties. Hooks allow developers to use these methods whilst improving code reuse with greater flexibility navigating the component tree.

**Types of hooks**

****

**Why react hooks were introduced?**

**How useState hooks work?,what are the arguments accepted by this hook and what is returned by this hook?**

**Systax :-**

Const [currentStateValue,functionToUpdateState]=useState(initial state value)

* useState hook is a function which is used to store state value in a functional components
* react hook “useState” cannot be called inside the call back,react hook must be called in a react function component
* it accepts the initial value of the state as arguments  
  (Number,string,Boolean,Object etc)
* it returns

**What are the difference in using hooks and class components with respect to state management ?**

When using useState() in class components , always the state variable is an object. Where as the state variable in hooks can be of any type like number,string,boolean,object and array.

When state variable is an object,setState() in class components automatically merges the new value to the state object,But in class of setter function in useState(),we need to explicitely merge the updated object property using spread operator

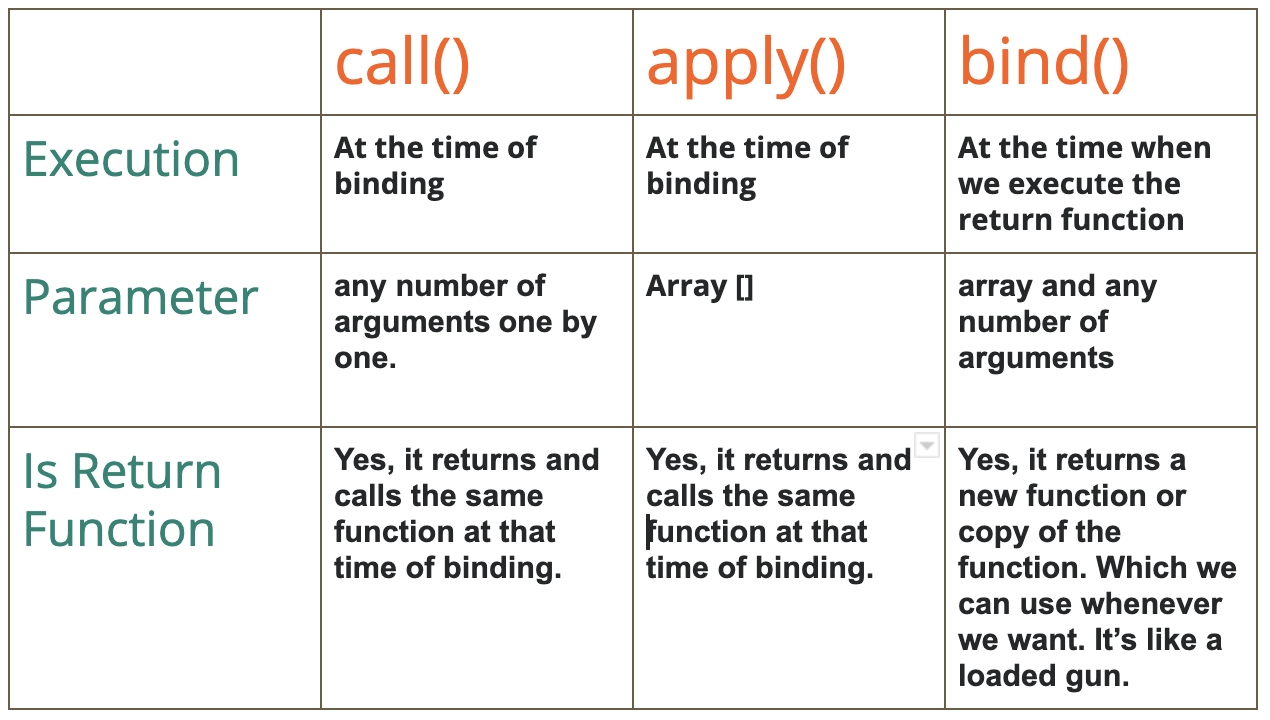
**setState and prevState in react**

setState() and prevState() are useState hooks that are used to change state in a React class component.

setState() indicates that this component and its children components are changed and need to be re-rendered with the updated state. setState is the primary method used to update the UI in response to event handlers and server responses.

prevState() is the same as the setState but the only difference between them is that if we want to change the state of a component based on the previous state of that component, we use this.setState(), which provides us the prevState.

**Difference between call apply bind**



**Handling Events**

Handling events with React elements is very similar to handling events on DOM elements. There are some syntax differences:

* React events are named using camelCase, rather than lowercase.
* With JSX you pass a function as the event handler, rather than a string.
* For example, the HTML:

<button **onclick**="activateLasers()">

Activate Lasers

</button>

* is slightly different in React:

<button **onClick**={activateLasers}> Activate Lasers

</button>

* Another difference is that you cannot return false to prevent default behavior in React. You must call preventDefault explicitly. For example, with plain HTML, to prevent the default form behavior of submitting, you can write:

<form onsubmit="console.log('You clicked submit.'); return false">

<button type="submit">Submit</button>

</form>

* In React, this could instead be:

function Form() {

function handleSubmit(e) {

e.preventDefault(); console.log('You clicked submit.');

}

return (

<form onSubmit={handleSubmit}>

<button type="submit">Submit</button>

</form>

);

}

Here, e is a synthetic event. React defines these synthetic events according to the [W3C spec](https://www.w3.org/TR/DOM-Level-3-Events/), so you don’t need to worry about cross-browser compatibility. React events do not work exactly the same as native events. See the [SyntheticEvent](https://reactjs.org/docs/events.html) reference guide to learn more.

When using React, you generally don’t need to call addEventListener to add listeners to a DOM element after it is created. Instead, just provide a listener when the element is initially rendered.

# Conditional Rendering

In React, you can create distinct components that encapsulate behavior you need. Then, you can render only some of them, depending on the state of your application.

Conditional rendering in React works the same way conditions work in JavaScript. Use JavaScript operators like [if](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else) or the [conditional operator](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Conditional_Operator) to create elements representing the current state, and let React update the UI to match them.

**REACT REFS**

Ref’s are simple properties,it allows us to interact with DOM native elements

**Refs and the DOM**

Refs provide a way to access DOM nodes or React elements created in the render method.

In the typical React dataflow, [props](https://reactjs.org/docs/components-and-props.html) are the only way that parent components interact with their children. To modify a child, you re-render it with new props. However, there are a few cases where you need to imperatively modify a child outside of the typical dataflow. The child to be modified could be an instance of a React component, or it could be a DOM element. For both of these cases, React provides an escape hatch.

### When to Use Refs

There are a few good use cases for refs:

* Managing focus, text selection, or media playback.
* Triggering imperative animations.
* Integrating with third-party DOM libraries

Avoid using refs for anything that can be done declaratively.

**What is use ref hook in react?**

The useRef is **a hook that allows to directly create a reference to the DOM element in the functional component**. Syntax: const refContainer = useRef(initialValue); The useRef returns a mutable ref object. This object has a property called

### **List in React**

Showing a list of elements happened almost in every project I was ever writing. And React really simplifies the rendering of lists inside the JSX by supporting the Javascript .map() method.

The .map() method in Javascript iterates through the parent array and calls a function on every element of that array. Then it creates a new array with transformed values. It doesn’t change the parent array.

Now, let’s imagine that you are creating a simple Todo App. The main element of your project will be a list of tasks. To show them, you will need data, which in most cases will be an array of elements, can be objects or strings, for example.

While you have the array with data, you can use the .map() method inside the JSX component the transform the data into the element.

.

### **Keys in React**

When creating a list of elements in a way you can see above, there’s one more necessary thing, and it’s a key.

key is a special attribute that you need to include in the element, and it should be a string. Keys in each list should be unique, which means that you shouldn’t use values that can be the same as the key. In other words, keys should be unique among the siblings, not globally.

For example, you should not use status or text as a key in our example.

Key is kind of the id for the element, and it helps React to identify which element was changed, added, or removed.

It’s a good practice to select a value that is a unique identifier for an item in the array, commonly it’s the ID.

**What is uncontrolled component?**

* + An uncontrolled component is **a component that renders form elements, where the form element's data is handled by the DOM (default DOM behavior)**. To access the input's DOM node and extract its value you can use a ref.
  + In most cases, we recommend using [controlled components](https://reactjs.org/docs/forms.html#controlled-components) to implement forms. In a controlled component, form data is handled by a React component. The alternative is uncontrolled components, where form data is handled by the DOM itself.
  + To write an uncontrolled component, instead of writing an event handler for every state update, you can [use a ref](https://reactjs.org/docs/refs-and-the-dom.html) to get form values from the DOM.
  + Since an uncontrolled component keeps the source of truth in the DOM, it is sometimes easier to integrate React and non-React code when using uncontrolled components. It can also be slightly less code if you want to be quick and dirty. Otherwise, you should usually use controlled components.

**onChange EVENT**

* The HTML DOM onchange event **occurs when the value of an element has been changed**. It also works with radio buttons and checkboxes when the checked state has been changed.
* The change in the state of an object is known as an **Event**. In html, there are various events which represents that some activity is performed by the user or by the browser. When [javascript](https://www.javatpoint.com/javascript-tutorial) code is included in [HTML](https://www.javatpoint.com/html-tutorial), js react over these events and allow the execution. This process of reacting over the events is called **Event Handling**. Thus, js handles the HTML events via **Event Handlers**.

**What is WEB PACK ?**

Webpack is **an open-source JavaScript module bundler**. It is made primarily for JavaScript, but it can transform front-end assets such as HTML, CSS, and images if the corresponding loaders are included. webpack takes modules with dependencies and generates static assets representing those modules.

**React Routing**

* Routing is nothing but navigating
* BrowserRouter: BrowserRouter is **a router implementation that uses the HTML5 history API(pushState, replaceState and the popstate event) to keep your UI in sync with the URL**. It is the parent component that is used to store all of the other components.

**React router :-**

* React router is a routing library built on the top of the react which is used to create the routing in react application
* React router dom is the version of react router version 5 designed for web application
* React router version is divided into three packages  
   1) react router: it is a common core components between DOM and native version  
   2)react-router-dom: this is the dom version designed for the web application  
   3) react-router-native : this is the native version designed for react native mobile application

**What is react router Why do we need a router in react?**

React Router is **a standard library for routing in React**. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.1

**Why is switch keyword used in react router v4?**

The switch component **looks through all of its child routes and it displays the first one whose path matches the current URL**. This component is what we want to use in most cases for most applications, because we have multiple routes and multiple plate pages in our app but we only want to show one page at a time

**Why we need exact In react router dom?**

React router does partial matching, so /users partially matches /users/create , so it would incorrectly return the Users route again! **The exact param disables the partial matching for a route and makes sure that it only returns the route if the path is an EXACT match to the current url**.

**REACT ROUTER v5 components**

* BrowserRouter
* Route
* Link
* Switch

**BrowserRouter**: It helps to

keep UI in sync with URL and it's a parent component, which stores all other

components.

**Route**: its conditionally shown

component, it renders the UI only if the path matches the current URL.

**Link**: it helps to create the links

to different routes and implements the navigation around Application.

(works like <a> tag)

**Switch:** It helps to render only the

first route that matches the location. Basically to render a single

components.

**REACT ROUTER v6 components**

* BrowserRouter
* Route
* Routes

**NOTE 😊**

1. **EXACT :-** it is used to match the exact value with the url
2. **PATH :-** path specifies the path name we assign to our component.
3. **COMPONENT** :- it refers to the component which will render a matching path

**HASH ROUTER**

A browser Router that using html 5 history API

**EG😉**

<BrowserRouter

basename={optionalString}

forceRefresh={optionalBool}

getUserConfirmation={optionalFunc}

keyLength={optionalNumber}

>

<App />

</BrowserRouter>;

<HashRouter  
basename={optionalString}  
getUserConfirmation={optionalFunc}  
hashType={optionalString}  
>  
<App />  
</HashRouter>

**forceRefresh — bool**

BrowserRouter don’t reload a page after changing url location.If true the router will use full page refreshes on page navigation. You may want to use this to imitate the way a traditional server-rendered app would work with full page refreshes between page navigation.

**getUserConfirmation — function**

A function to use to confirm navigation. Defaults to using [window.confirm](https://developer.mozilla.org/en-US/docs/Web/API/Window/confirm).

functionName = (message, callback) => {  
 // this is the default behavior  
 const allowTransition = window.confirm(message);  
 callback(allowTransition);  
 }<BrowserRouter  
 getUserConfirmation={functionName}  
/>

**keyLength — number**

The length of location.key. Defaults to 6.

<BrowserRouter keyLength={8} />

If you only start to learn React, you needn’t it.

**hashType**

The type of encoding to use for window.location.hash. Available values are:

* "slash" - Creates hashes like #/ and #/main/profile
* "noslash" - Creates hashes like # and #main/profile
* "hashbang" - Creates ["ajax crawlable"](https://developers.google.com/webmasters/ajax-crawling/docs/learn-more) (deprecated by Google) hashes like #!/ and #!/main/profile

Defaults to "slash".

HashRouter

<HashRouter  
 basename={optionalString}  
 getUserConfirmation={optionalFunc}  
 hashType={optionalString}  
>  
 <App />  
</HashRouter>

HashRouter can take 3 properties: **basename**, **getUserConfirmation**and **hashType**.

**WHAT is useEffect Hooks?**

The Effect Hook, useEffect , **adds the ability to perform side effects from a function component**. It serves the same purpose as componentDidMount , componentDidUpdate , and componentWillUnmount in React classes, but unified into a single API.

useEffect after render: We know that, the useEffect() is used for **causing side effects in functional components** and it is also capable for handling componentDidMount(), componentDidUpdate() and componentWillUnmount() life-cycle methods of class based components into functional component.

**Higher order component (HOC)**

* A higher-order component (HOC) is **an advanced technique in React for reusing component logic**. HOCs are not part of the React API, per se. They are a pattern that emerges from React's compositional nature. Concretely, a higher-order component is a function that takes a component and returns a new component.
* **a higher-order component is a function that takes a component and returns a new component.**

What is withRouter hoc?

withRouter is **a higher-order component provided by react-router-dom which gives you access to history , match and location object from props of particular object which is wrapped with withRouter** .

# Context API

Context provides a way to pass data through the component tree without having to pass props down manually at every level.

In a typical React application, data is passed top-down (parent to child) via props, but such usage can be cumbersome for certain types of props (e.g. locale preference, UI theme) that are required by many components within an application. Context provides a way to share values like these between components without having to explicitly pass a prop through every level of the tree.

## **When to Use Context**

Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language. For example, in the code below we manually thread through a “theme” prop in order to style the Button component.

### **React.createContext**

const MyContext = React.createContext(defaultValue);

Creates a Context object. When React renders a component that subscribes to this Context object it will read the current context value from the closest matching Provider above it in the tree.

### Context.Provider

<MyContext.Provider **value={/\* some value \*/}>//value is very important**

Every Context object comes with a Provider React component that allows consuming components to subscribe to context changes.

The Provider component accepts a value prop to be passed to consuming components that are descendants of this Provider. One Provider can be connected to many consumers. Providers can be nested to override values deeper within the tree.

### **Context.Consumer**

<MyContext.Consumer>

{value => /\* render something based on the context value \*/}

</MyContext.Consumer>

A React component that subscribes to context changes. Using this component lets you subscribe to a context within a [function component](https://reactjs.org/docs/components-and-props.html#function-and-class-components).

Requires a [function as a child](https://reactjs.org/docs/render-props.html#using-props-other-than-render). The function receives the current context value and returns a React node. The value argument passed to the function will be equal to the value prop of the closest Provider for this context above in the tree. If there is no Provider for this context above, the value argument will be equal to the defaultValue that was passed to createContext().

### **Context.displayName**

Context object accepts a displayName string property. React DevTools uses this string to determine what to display for the context.

For example, the following component will appear as MyDisplayName in the DevTools:

const MyContext = React.createContext(/\* some value \*/);

MyContext.displayName = 'MyDisplayName';

<MyContext.Provider> // "MyDisplayName.Provider" in DevTools

<MyContext.Consumer> // "MyDisplayName.Consumer" in DevTools

## **What is Error Boundaries ? 😊😊😉**

A JavaScript error in a part of the UI shouldn’t break the whole app. To solve this problem for React users, React 16 introduces a new concept of an “error boundary”.

Error boundaries are React components that **catch JavaScript errors anywhere in their child component tree, log those errors, and display a fallback UI** instead of the component tree that crashed. Error boundaries catch errors during rendering, in lifecycle methods, and in constructors of the whole tree below them.

**Note**

Error boundaries do **not** catch errors for:

* Event handlers
* Asynchronous code (e.g. setTimeout or requestAnimationFrame callbacks)
* Server side rendering
* Errors thrown in the error boundary itself (rather than its children)
* Error boundaries work like a JavaScript catch {} block, but for components. Only class components can be error boundaries. In practice, most of the time you’ll want to declare an error boundary component once and use it throughout your application.

[**what is axios*in react***](https://www.google.com/search?rlz=1C1CHBF_enIN972IN972&sxsrf=APq-WBtuYJ6uVwREJrvBaj_uJZ5bRIGlGg:1647507886714&q=what+is+axios+in+react&spell=1&sa=X&ved=2ahUKEwiymuiS5cz2AhUnslYBHXCzAIgQirwEKAB6BAgBEDI) **?**

* Axios is **a promise-based HTTP Client for node.** **js and the browser**. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node. js http module, while on the client (browser) it uses XMLHttpRequests.
* Axios is a promise based HTTP client for the browser and Node.js. Axios makes it easy to send asynchronous HTTP requests to REST endpoints and perform CRUD operations. It can be used in plain JavaScript or with a library such as Vue or React.

## **What is the difference between NavLink and Link?**

The Link component is used to navigate the different routes on the site. But NavLink is used to add the style attributes to the active routes.

In our routing app, we have three routes which are [home, /users, /contact] Let’s style them using NavLink.

We need to add a new prop called activeClassName to the NavLink component so that it applies that class whenever the route it is active.

**useLocation()**

useLocation: **This hook returns the location object used by the react-router**. This object represents the current URL and is immutable. Whenever the URL changes, the useLocation() hook returns a newly updated location object.

**FIREBASE**

Google Firebase is a Google-backed application development software that **enables developers to develop iOS, Android and Web apps**. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

**forwordRef() :-**

React forwardRef is **a method that allows parent components pass down (i.e., “forward”) refs to their children**. Using forwardRef in React gives the child component a reference to a DOM element created by its parent component. This then allows the child to read and modify that element anywhere it is being used.

**WEBPACK :-**

Webpack is **a popular module bundling system built on top of Node.** **js**. It can handle not only combination and minification of JavaScript and CSS files, but also other assets such as image files (spriting) through the use of plugins.

**Do I need webpack With React?**

Well, **we don't necessarily need webpack to work with React**, other alternatives could be Browserify, Parsel, Brunch, etc, but honestly, I don't know how well they fit in with React. js. Webpack is the most widely used and an accepted module bundler and task runner throughout React. js community.

# Code-Splitting

## **Bundling**

Most React apps will have their files “bundled” using tools like [Webpack](https://webpack.js.org/), [Rollup](https://rollupjs.org/) or [Browserify](http://browserify.org/). Bundling is the process of following imported files and merging them into a single file: a “bundle”. This bundle can then be included on a webpage to load an entire app at once.

**REDUX**

# Why Use React Redux?

Redux itself is a standalone library that can be used with any UI layer or framework, including React, Angular, Vue, Ember, and vanilla JS. Although Redux and React are commonly used together, they are independent of each other.

If you are using Redux with any kind of UI framework, you will normally use a "UI binding" library to tie Redux together with your UI framework, rather than directly interacting with the store from your UI code.

**React Redux is the official Redux UI binding library for React**. If you are using Redux and React together, you should also use React Redux to bind these two libraries.