[Apply](#Apply)

[Bind](#Bind)

[Call](#Call)

[Concat Function](#ConcatFunction)

[Destructuring](#Destructuring)

[Fetch](https://www.geeksforgeeks.org/javascript-fetch-method/)

[Filter](#FilterFunctioin)

[Filter Function](#FilterFunctioin)

Functional component

[High Order Function](#HighOrderFunction)

[Join](#join)

[JSX](#JSX)

[Map](#MapFunction)

[Map Function](#MapFunction)

[Polymorphism](#polymorphism)

[Props Function](#Props)

[QuerySelectorAll Method](#querySelectorAll)

[QuerySelector Method](#querySelector)

[Reduce](#ReduceFunction)

[Reduce Function](#ReduceFunction)

[rest](#rest)

[Split](#split)

[Spread](#Spread)

[UseState](#useState)

[UseEffect](#useEffect)

[UseReference](#useRef)

[UseContext](#UseContext)

[UseReducer](#useReducer)

[Javascript Module](#JavaScriptmodules)

Object Literal

Multiline string

Class Props

Babel definition & use

CustomHook

[State v/s Props](#StateVersusProps)

[Difference between class component and functional component](#ClassFunctionalComponentDifference)

A **higher order function** is a function that takes one or more functions as arguments, or returns a function as its result. There are several different types of higher order functions like map and reduce.

map() creates a new array from calling a function for every array element.

map() does not execute the function for empty elements.

map() does not change the original array.

The filter() method creates a new array filled with elements that pass a test provided by a function.

The filter() method does not execute the function for empty elements.

The filter() method does not change the original array.

The **reduce**() method is an iterative method. It runs a "reducer" callback function over all elements in the array, in ascending-index order, and accumulates them into a single value. Every time, the return value of callbackFn is passed into callbackFn again on next invocation as accumulator.

The concat() method joins two or more strings.

The concat() method does not change the existing strings.

The concat() method returns a new string.

Document: querySelector() method. The Document method querySelector() returns the first Element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.

The querySelectorAll() method returns all elements that matches a CSS selector(s).

The querySelectorAll() method returns a [NodeList](https://www.w3schools.com/js/js_htmldom_nodelist.asp).

The querySelectorAll() method throws a SYNTAX\_ERR exception if the selector(s) is invalid

The split() method splits a string into an array of substrings.

The split() method returns the new array.

The split() method does not change the original string.

If (" ") is used as separator, the string is split between words.

The join() method returns an array as a string.

The join() method does not change the original array.

Any separator can be specified. The default is comma (,).

The **polymorphism** is a core concept of an object-oriented paradigm that provides a way to perform a single action in different forms. It provides an ability to call the same method on different JavaScript objects.

Destructuring is a JavaScript expression that allows us to extract data from arrays, objects, and maps and set them into new, distinct variables. Destructuring allows us to extract multiple properties, or items, from an array​ at a time.

The **rest** operator in javaScript allows a function to take an indefinite number of arguments and bundle them in an array, thus allowing us to write functions that can accept a variable number of arguments, irrespective of the number of parameters defined.

The Spread operator allows an iterable to expand in places where 0+ arguments are expected. It is mostly used in the variable array where there is more than 1 value is expected. It allows us the privilege to obtain a list of parameters from an array.

The syntax of the Spread operator is the same as the Rest parameter but it works completely opposite of it.

With the call() function, you can write a method that can be used on different objects.

Apply function in the JavaScript is used to call a function on different objects with the given this value, and the arguments are passed in the form of an array. This method allows us to write methods that can be used on different objects and hence increase the reusability of code

With the bind() method, an object can borrow a method from another object.

JSX stands for JavaScript XML, a syntax extension employed in React that combines JavaScript and HTML-like code into a single cohesive language. With JSX in React, developers can seamlessly write HTML-like elements and components within their JavaScript files.

In ReactJS, the props are a type of object where the value of attributes of a tag is stored. The word “props” implies “properties”, and its working functionality is quite similar to HTML attributes. Basically, these props components are read-only components.

Props stand for properties. They are read-only values that can be passed across components in order to display relevant data in your React apps.

Copy & Concat 🡪 …spread

To get the rest of the parameters use ….rest

**useState** is React Hook that allows you to add state to a functional component. It returns an array with two values: the current state and a function to update it. The Hook takes an initial state value as an argument and returns an updated state value whenever the setter function is called.

The **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>)

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.

React **UseContext** is a way to manage state globally. It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone.

he **useReducer** Hook is used to store and update states, just like the useState Hook. It accepts a reducer function as its first parameter and the initial state as the second. useReducer returns an array that holds the current state value and a dispatch function to which you can pass an action and later invoke it.

**JavaScript modules** allow you to break up your code into separate files. This makes it easier to maintain the code-base. ES Modules rely on the import and export statements.

**Functional Components vs Class Compon****ents:**

|  |  |
| --- | --- |
| [**Functional Components**](https://www.geeksforgeeks.org/reactjs-functional-components/) | [**Class Components**](https://www.geeksforgeeks.org/reactjs-class-based-components/) |
| A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX). | A class component requires you to extend from React. Component and create a render function that returns a React element. |
| There is no render method used in functional components. | It must have the render() method returning JSX (which is syntactically similar to HTML) |
| Functional components run from top to bottom and once the function is returned it can’t be kept alive. | The class component is instantiated and different life cycle method is kept alive and is run and invoked depending on the phase of the class component. |
| Also known as Stateless components as they simply accept data and display them in some form, they are mainly responsible for rendering UI. | Also known as Stateful components because they implement logic and state. |
| React lifecycle methods (for example, componentDidMount) cannot be used in functional components. | React lifecycle methods can be used inside class components (for example, componentDidMount). |
| Hooks can be easily used in functional components to make them Stateful.  Example:  const [name,SetName]= React.useState(' ') | It requires different syntax inside a class component to implement hooks.  Example:  constructor(props) {    super(props);    this.state = {name: ' '} } |
| Constructors are not used. | Constructor is used as it needs to store state. |

## **State Versus Props**

Let’s go through the fundamental differences between state and [props](https://reactjs.org/docs/render-props.html):

|  |  |  |
| --- | --- | --- |
|  | **State** | **Props** |
| **Use Case** | State is used to store the data of the components that have to be rendered to the view | Props are used to pass data and event handlers to the children components |
| **⇙Mutability** | State holds the data and can change over time | Props are immutable—once set, props cannot be changed |
| **Component** | State can only be used in class components | Props can be used in both functional and class components |
| **Updation** | Event handlers generally update state | The parent component sets props for the children components |

https://www.freecodecamp.org/news/props-in-react/#:~:text=Props%20stand%20for%20properties.,the%20sum%20of%20both%20parameters.

Create component steps

Cmd

E:

Cd test123

Make any subdirectory

cd subdirectory

npx create-react-app <any name>

cd <any name>

code . to

react office site link is <https://react.dev/learn> (Start with Writing markup with JSX)

<https://react.dev/learn#writing-markup-with-jsx>

<https://react.dev/learn#displaying-data>

<https://react.dev/learn#updating-the-screen>

or <https://reactjs.org>

Home work

Clear the form with the help of useState.

What are the features of ES6?

Apply different color in UseRef hooks. (Hint : you have to apply useState)