**JS assignment**

**Topics: IIFE, call apply, bind, anonymous functions, Inner function, HOF**

1. **What is an IIFE (Immediately Invoked Function Expression) and why would you use it in JavaScript? Give an example of IIFE.**

**Answer:**

An IIFE (Immediately Invoked Function Expression) is a JavaScript function that is defined and executed immediately after it’s created, without needing to be called later in the code.

It’s typically written as a function expression wrapped in parentheses, followed by another pair of parentheses to execute it right away.

IIFE is used for,

* **Avoid polluting the global scope**  
  Variables inside an IIFE are scoped to that function, preventing name conflicts with other scripts.
* **Encapsulation**  
  It creates a private scope where you can store temporary variables or logic without exposing them.
* **Initialization logic**  
  Useful for running setup code that should only execute once.

Example:

(function () {

console.log(“Hello World”);

})(); // Output: Hello World

1. **What is the purpose of using the bind() method in JavaScript and how is it different from call() and apply()?**

**Answer:**

The bind() method in JavaScript is used to create a new function with a fixed this value (and optionally preset arguments), without immediately executing it.

* To ensure a function always runs with a specific this context, no matter how or where it’s called.
* Commonly used when passing a method as a callback, where this might otherwise get lost.

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **What it does** | **When it runs** | **Arguments passing style** |
| **bind()** | Returns a new function with this bound (and optionally preset arguments). Does not run immediately. | Later, when the returned function is called. | Arguments passed normally when calling the new function. |
| **call()** | Calls the function immediately with a specific this. | Immediately | Pass arguments individually. |
| **apply()** | Calls the function immediately with a specific this. | Immediately | Pass arguments as an array. |

1. **What is a Higher-Order Function (HOF) in JavaScript and how is it different from regular functions? Explain with an example.**

**Answer:**

A Higher-Order Function (HOF) in JavaScript is a function that either takes another function as an argument or returns a function as its result

or both.

* **Regular function** → takes data as input and returns data.
* **Higher-Order function** → takes a function as input, returns a function, or both — allowing more dynamic and reusable code.

1. **Write a function called multiplyBy that takes a number as input and returns a new function that multiplies any number passed into it by the original number.**

**Answer:**

const myRes=function multipyBy(number){

return function(originalNumber){

return number \* originalNumber;

}

}

const multiply=myRes(10);

console.log(multiply(5));

1. **Write a function named sortArray that takes in two parameters:**

**1. An array of numbers**

**2. A boolean value ascending that indicates whether the array should be sorted in ascending or descending order.**

* **The sortArray function should return the sorted array. Use an anonymous function to do the actual sorting, rather than using the built-in sort method.**

**Answer:**

const sortedArray=function(myArray, permission){

if(permission){

for(let i=0;i<myArray.length;i++){

for(let j=0;j<myArray.length-i-1;j++)

if(myArray[j]> myArray[j+1]){

let temp=myArray[j];

myArray[j]=myArray[j+1];

myArray[j+1]=temp;

} } }

else{

for(let i=0;i<myArray.length;i++){

for(let j=0;j<myArray.length-i-1;j++)

if(myArray[j]< myArray[j+1]){

let temp=myArray[j];

myArray[j]=myArray[j+1];

myArray[j+1]=temp;

}

}

}

return myArray;

}

let array1=[4,8,2,1,9]

console.log(sortedArray(array1,false))