IIFE(Immediate invoked function expression)

An IIFE (Immediately Invoked Function Expression) is a JavaScript function that executes immediately upon its definition.

It provides a way to encapsulate code and create a private scope, allowing for the isolation of variables and preventing interference with other parts of the code.

Use Cases of IIFE

1- Avoiding global namespace pollution

2- compatibility with older javascript

3- Encapsulating code

4- Modular Pattern

5- To create closures in javascript

6- IIFE is used to ceate private and public variables and methods

7- it is used to execute the async and await function

Ex: 1st

(function(){

    console.log('Hi IIFE!!')

})()

Ex: 2nd

var result = (()=> {

    var x = 10;

    var y = 20;

    return x + y;

})()

console.log(result);

Ex: 3rd

((name)=> {

    console.log(`Hi! I'm Learning ${name}`);

})("IIFE")

Ex: 4th using async and await

(async ()=>{

    const response = await fetch('https://jsonplaceholder.typicode.com/users');

    const data = await response.json();

    console.log(data);

})()

Ex: 5th

For(var i = 0; i < 2; i++) {

    const button = document.createElement("button");

    button.innerText = `Button ${i}`;

    button.onclick = function () {

      console.log(i);

    };

    document.body.appendChild(button);

  }

  console.log(i); // 2

Explanation: above example for loop with var first it will run the loop till end

it shows us calculate, updated value means 2

FOR OVERCOME THIS CHALLENGE => we have used IIFE to get the current value

  for (var i = 0; i < 2; i++) {

    const button = document.createElement("button");

    button.innerText = `Button ${i}`;

    button.onclick = (function (copyOfI) {

      return function () {

        console.log(copyOfI);

      };

    })(i);

    document.body.appendChild(button);

  }

  console.log(i); // 2

Ex: 6th

Here’s an example demonstrating how an IIFE can be used to create private variables:

var counter = (()=> {

    var count = 0;

    return{

        increment: function(){

                count++;

        },

        decrement: function(){

            count--;

        },

        getCount: function(){

            return count;

        }

    }

})()

counter.increment();

counter.decrement()

console.log(counter.getCount());

// here we are trying to access the private variale directly

Ex: 7th

var counter = (function() {  
 let privateCounter = 0; // private variable

function changeBy(val) // private function  
{  
 privateCounter += val;  
}

return {

increment(){

changeBy(1)

},

decrement(){

changeBy(-1)

},

getCount() {

return count;

}

};

})();

// Increment the counter  
 counter.increment()  
counter.increment()

console.log(counter.getCount()); // Output: 2  
counter.decrement()  
console.log(counter.getCount());