Immediately Invoked Function Expressions (IIFE) in JavaScript

Last Updated : 11 Mar, 2024

Immediately Invoked Function Expressions (IIFE) are JavaScript functions that are executed immediately after they are defined. They are typically used to create a local scope for variables to prevent them from polluting the global scope.

Syntax:

```
(function (){
// Function Logic Here.
})();
```

Immediately Invoked Function Expressions (IIFE) Examples

Example: Here's a basic example of an IIFE .

Javascript

```
(function() {
    // IIFE code block
    var localVar = 'This is a local variable';
    console.log(localVar); // Output: This is a local variable
})();
```

Output

```
This is a local variable
```

Explanation: The function is wrapped in parentheses `(function() { ... })`, followed by `()` to immediately invoke it.

Example 2: Here's another example of an IIFE that stores and display result.

Javascript

```
var result = (function() {
    var x = 10;
    var y = 20;
    return x + y;
})();

console.log(result); // Output: 30
```

Output

```
30
```

Explanation: The IIFE is immediately invoked and returns the sum of `x` and `y`. The result of the IIFE, which is `30`, is assigned to the variable `result`.

IIFEs are commonly used to create private scope in JavaScript, allowing variables and functions to be encapsulated and inaccessible from outside the function.

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

Javascript

```
var counter = (function() {
    var count = 0;

    return {
        increment: function() {
            count++;
        },
        decrement: function() {
            count--;
        },
        getCount: function() {
            return count;
```

```
        }
    };
})();

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

console.log(counter.getCount()); // Output: 3

// Trying to access the private count variable directly
console.log(counter.count); // Output: undefined (cannot access priv
```

Output

```
3
undefined
```

Explanation: `Here, count` is a private variable scoped to the IIFE, inaccessible from outside. The returned object exposes methods (`increment`, `decrement`, and `getCount`) that allow controlled manipulation and access to the private `count` variable.

Use Cases Of IIFE

- Avoid polluting the global namespace.
- To create closures in JavaScript.
- IIFE is used to create private and public variables and methods.
- It is used to execute the async and await function.
- It is used to work with require function.