JavaScript Closures

Back to JS page

Table of Contents

- JavaScript Closures
  - Global and Local Variables
    - Example 1
  - A Counter Dilemma
    - Example 2
  - JavaScript Nested Functions
    - Example 3
  - JavaScript Closures
    - Example 4
  - Document
  - Reference

Global and Local Variables

A function can access all variables defined inside it (local), and all variables defined outside it (global).

A closure is a function that has access to the parent scope, even after the parent function has closed.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Closures</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Global and Local Variables</h4>
<p id="demo"></p>

<script>
let a = 4;  // global variable

function myFunction() {
  let b = 2;  // local variable
  return a * b;
}

document.getElementById("demo").innerHTML = "a * b = " + myFunction();
</script>

</body>
</html>
```

Example 1

Result View Example

A Counter Dilemma

Suppose we want to use a variable to count something. We want this counter to be accessible to all functions, but we don't want it to be modified by other code.

Using a global variable won't work because it can be changed from anywhere:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Closures</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>A Counter Dilemma</h4>
<p id="demo1"></p>
<p id="demo2"></p>

<script>
let counter = 0;  // global counter - can be modified by anyone

function add() {
  counter += 1;
}

// Calling add() 3 times
add();
add();
add();

document.getElementById("demo1").innerHTML = "Counter: " + counter;

// Global variable can be accidentally changed
counter = 100;
document.getElementById("demo2").innerHTML = "Counter after modification: " + counter;
</script>

</body>
</html>
```

Example 2

Result View Example

JavaScript Nested Functions

A function can have nested functions. The inner function has access to the outer function's variables:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Closures</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>JavaScript Nested Functions</h4>
<p id="demo"></p>

<script>
function outerFunction() {
  let outerVar = "I am from outer function";

  function innerFunction() {
    document.getElementById("demo").innerHTML = outerVar;
  }

  innerFunction();
}

outerFunction();
</script>

</body>
</html>
```

Example 3

Result View Example

JavaScript Closures

A closure is a function that remembers its outer variables and can access them.

In JavaScript, all functions are closures because they remember where they were created:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Closures</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>JavaScript Closures</h4>
<p id="demo"></p>

<script>
function createCounter() {
  let counter = 0;  // private variable

  function increment() {
    counter += 1;
    return counter;
  }

  return increment;  // return the inner function (closure)
}

const myCounter = createCounter();

document.getElementById("demo").innerHTML =
  myCounter() + "<br>" +
  myCounter() + "<br>" +
  myCounter();
</script>

</body>
</html>
```

Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Closures