

JS Introduction

- [JS Home](#)
- [JS Introduction](#)
- [JS How to..](#)
- [JS Syntax](#)
- [JS Variables](#)
- [JS Operators](#)
- [JS Datatypes](#)
- [JS Conditionals](#)
- [JS Loops](#)

JS Core Object

- [JS Boolean](#)
- [JS Number](#)
- [JS Global](#)
- [JS Math](#)
- [JS String](#)
- [JS Arrays](#)
- [JS Arrays Methods](#)
- [JS Date](#)

JS Object

- [JS Object Introduction](#)
- [JS Create Object](#)
- [JS Object Properties](#)

JS Events

- [JS Events Intro](#)
- [JS Event Flow](#)
- [JS Mouse Event](#)
- [JS Window Event](#)

JS Function

- [JS Functions Intro](#)
- [JS Function Type](#)
- [JS Function Return](#)
- JS Function Closures**
- [JS Function Private](#)

JS BOM

- [BOM Location Object](#)
- [BOM History Object](#)
- [BOM Screen Object](#)
- [BOM System Object](#)
- [BOM Window Object](#)
- [BOM Interval & Timeout](#)
- [BOM Navigator Object](#)

JS Form

- [JS Basic Form](#)
- [JS Form Event](#)
- [JS Form Scripting](#)
- [JS Form Validation](#)
- [JS Form Input Controls](#)

JS DOM

- [JS DOM Intro](#)
- [DOM Metadata](#)
- [DOM Modify](#)
- [DOM CSS](#)
- [DOM Text Object](#)
- [DOM CSS Style object](#)
- [DOM Stylesheet](#)
- [DOM Element](#)
- [Attributes](#)
- [DOM Element Object](#)


[« JS Function Return](#)
[JS Function Private »](#)

Javascript Closures and Recursions

More advanced applications of Javascript functions.

Javascript Function: Closures

In Javascript, **closures** are functions that have access to variables from another functions scope. This is achieved by creating a function within a function.

The outer function creates a **reference** to the inner anonymous function, and thus making it possible to call the inner function via the reference.

The idea of closures is that the local variables must remain **accessible** to inner functions even when it seems it is out of scope.

Syntax: Javascript Function Closures

```

1 <script>
2
3 </script>
```

Example: Javascript Function Return Values

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Javascript Functions - Closures</title>
5   <script>
6     function Cars(brand, name){
7       var text = "Car's brand is " + brand + " and name is " + name; //local variable
8
9       var output = function() //Anonymous function
10      {
11        document.write(text + "<hr>");
12      }
13      return output;
14    }
15  </script>
16 </head>
17 <body>
18 <script>
19 var input1 = Cars("Honda", "Accord"); /*A reference to the anonymous function*/
20
21 var input2 = Cars("Toyota", "Camry");
22
23 alert(input1);
24
25 input1(); //Anonymous function called.
26 input2();
27 </script>
28 </body>
29 </html>
```

Note: All javascript functions are closures, as they are objects and have a scope chain associated with them.

[Give it a TRY! »](#)

Javascript Functions: Recursion

By definition, a **recursive function** is created when a function calls itself by name.

In Recusion, the program starts executing at the beginning of the function and backs up to where it was when it called the function and starts executing from that point, if the test condition is satisfied.

But most importantly, there must be a way to stop the recursion at some point, or else it would be infinite, ultimately causing the program to crash.

Syntax: Javascript Anonymous Function as a Variable

```

1 <script>
2 var variable = function()
3 {
```

DOM Event Object
DOM HTML Object
DOM Navigation Tree

JS Multimedia

JS Images
JS Animation
JS Canvas

JS Error Handling

JS Try & Catch
JS Debug

JS Regular Exp

JS Regex Intro
JS Regex Strings
JS Regex Meta
JS Regex Meta Apps
JS Regex Form

JS Cookies

JS Cookies Intro
JS Cookies Create

JS JSON

JS JSON Intro
JS JSON Serialization

JS Adv Techniques

JS Adv Functions
JS Tamper Proof

JS Libraries

JS Libraries
JS Jquery
JS AngularJS
JS D3

JS Best Practices

JS Maintainability
JS Performance
JS Deployment

JS References

JavaScript Objects
JavaScript DOM
JavaScript BOM

```
4 .....  
5 .....  
6 }  
7 </script>
```

Example: Javascript Functions - Recursion

```
1 <!DOCTYPE html>  
2 <html>  
3 <head>  
4   <title>Javascript Functions : Recursion</title>  
5   <script>  
6     var counter = 0;  
7     function fibo(num)  
8     {  
9       counter++;  
10      switch(num)  
11      {  
12        case 0:  
13          return(0);  
14          break;  
15        case 1:  
16          return(1);  
17          break;  
18        default:  
19          return(fibo(num - 1) + fibo(num -2));  
20          break;  
21      }  
22    }  
23  </script>  
24 </head>  
25 <body>  
26 <script>  
27 for (i=0; i < 30 ; i++)  
28 {  
29   output = fibo(i);  
30   document.write(output + " ");  
31 }  
32 alert("The Recursion Function was called " + output + " times");  
33 </script>  
34 </body>  
35 </html>
```

Note: For each iteration the function **fibo()** is called.

[Give it a TRY! »](#)



[« JS Function Return](#)

[JS Function Private »](#)

TUTORIALS

REFERENCES

COLOR CODE GENERATOR

Demos

- HTML5
- CSS3
- JAVASCRIPT
- JQUERY
- JAVA
- PHP
- ASP
- MYSQL
- PYTHON
- XML
- SITEMAP

- HTML5
- CSS3
- JAVASCRIPT
- JQUERY
- JAVA
- PHP
- ASP
- MYSQL
- PYTHON
- XML
- ALL REFERENCES



- HTML5 DEMOS
- CSS3 DEMOS
- JAVASCRIPT DEMOS
- JQUERY DEMOS
- PHP DEMOS
- JAVA DEMOS
- ASP DEMOS
- MY SQL DEMOS
- PYTHON DEMOS
- XML DEMOS
- Everything else...

- AB
- Cof
- Leç
- Pri