Javascript Functions

Introduction

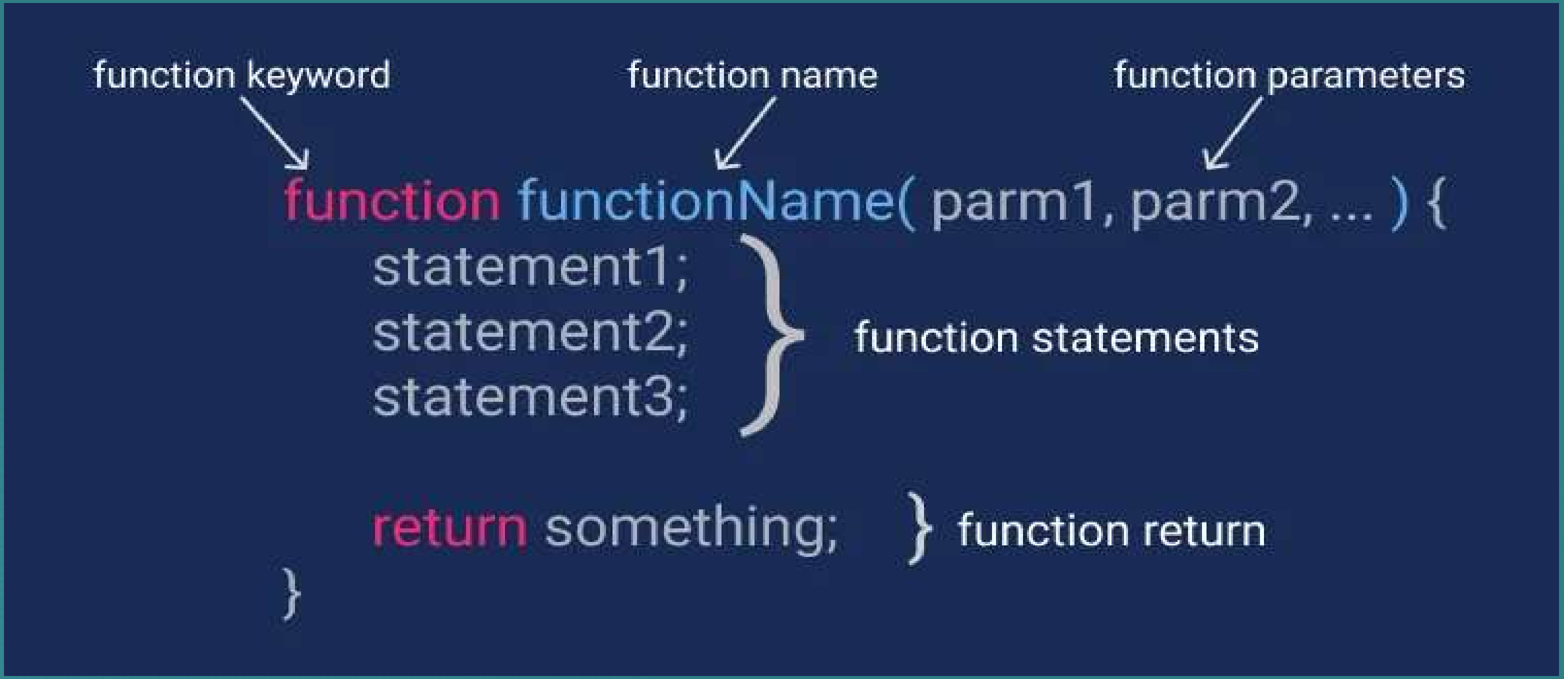
- A function contains code that will be executed by an event or by a call to that function.
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

- **Code reusability**: We can call a function several times so it save coding.
- **Less coding**: It makes our program compact. We don’t need to write many lines of code each time to perform a common task.

Function Definition



```
function functionName( parm1, parm2, ... ) {
    statement1;
    statement2;
    statement3;

    return something;
}
```

- A function is defined using the keyword **function** and is immediately followed by the function name and a pair of parenthesis.
- Inside the parenthesis, you can define any number of arguments that are passed to the function when it is called.
- The function can return a value by using the return statement, or it can end the function by using the **return** keyword.

Java script Functions

- The keyword **`function`** is used to define a function:

```
function add(x,y)
{
   return(x+y);
  }
```

Madhu Bhan

- Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the <head> section.

Javascript Functions

The code inside the function will execute when "something" **invokes** (calls) the function:

Madhu Bhan

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code

when a user clicks a button(Event)

```
<html>
<body>
<script>
function msg()   {
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
</body></html>
```

When an event occurs like button click, then you can use some ***event handler*** *to trigger the event which will call the function.*

When it is invoked (called) from JavaScript code

```
<html><head><title>Even Number Tester</title>
 <script type="text/javascript">
  function isEven(n)
  {   if (n%2==0)
        return true;
   else
        return false;    }
 </script> </head>
<body> <script type="text/javascript">
  testNum = parseInt(prompt("Enter a positive integer", "7"));
     if (isEven(testNum)) {
    document.write(testNum + " is an <b>Even</b> number.");    }
  else {    document.write(testNum + " is an <b>odd </b> number.");
   }
 </script></body></html>
```

Function in head section

Function definitions goes in the HEAD

HEAD is loaded first, so the function is defined before code in the BODY is executed

Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

Madhu Bhan

```
<html> <head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload
event")
}
</script></head>
<body> ………. message()
</body></html>
```

Function parameters

```
function Name(paramet1, paramet2, paramet3,...)
{ // Statements }
```

Parameter Rules:

- There is no need to specify the data type for parameters in JavaScript function definitions.
- It does not perform type-checking based on the passed-in JavaScript functions.
- It does not check the number of received arguments.

Example 1

Largest of 3 Numbers-

```
<body>
        <script>
      function GFG( var1, var2, var3 ) {
         if( var1 > var2 ) {
            if( var1 > var3 ) {
               return var1;              }
            else {
               return var3;              }
         }
         else {
            if( var2 > var3 ) {
               return var2;               }
            else {
               return var3;               }          }       }
      Largest = GFG(4, 50, 6);
 document.write(Largest);    </script>
</body>
```

Madhu Bhan

Example 2

- The default parameters are used to initialize the named parameters with default values in case, when no value or undefined is passed.

```
<html> <body>
<script>
function GFG(num1, num2 = 2) {
return (num1 * num2);        }
Product = GFG(4);
document.write(Product);
</script>
</body></html>
```

Arguments Pass by Value

In a function call, the parameters are called as arguments. The pass-by value sends the value of the variable to the function. If the function changes the value of arguments then it does not affect the original value.

```
<script>
                    function PASSbyVALUE(var1, var2)
                   {
                           var1 = 100; var2 = 200;
                    document.write("var1 =" + var1 +" var2 =" +var2);
                  }

                          var1 = 10;              var2 = 20;
             /* The value of variable before Function call */
             document.write('<br/>');
             document.write("var1 =" + var1 +" var2 =" +var2);
             /* Function call */
             PASSbyVALUE(var1,var2);
             /* The value of variable after Function call */
             document.write("var1 =" + var1 +" var2 =" +var2);
             document.write('<br/>');
       </script>
```

Output

```
(Before function calling)

var1 =10 var2 =20

(Inside function)

var1 =100 var2 =200

(After function calling)

var1 =10 var2 =20
```

Calling JavaScript function in an Anchor Tag

```
<html>
<head>
<script language="JavaScript">
function anchor_test()
{
    window.alert("This is an anchor test.")
}
</script>
</head>
<body>
<a href="javascript:anchor_test()">Test</a>
</body>
</html>
```

Global and Local Variable

- **Global Variable:** These are variables that are defined in global scope i.e. outside of functions. These variables have global scope, so they can be accessed by any function directly.

- **Local Variable:** When you use JavaScript, local variables are variables that are defined within functions. They have local scope, which means that they can only be used within the functions that define them. Accessing them outside the function will throw an error

How to use variables

- Variables that are created inside a function are local variables, and local variables can only be referred to by the code within the function.
- Variables created outside of functions are global variables, and the code in all functions has access to all global variables.
- If you forget to code the var keyword in a variable declaration, the JavaScript engine assumes that the variable is global. This can cause debugging problems.
- In general, it's better to pass local variables from one function to another as parameters than it is to use global variables. That will make your code easier to understand with less chance of errors.

Example 3

```
<script>
  var petName = 'Rocky' // Global variable

  function myFunction() {
    fruit = 'apple'; // Considered global-without key word
      document.write(‘My pet name is ' +petName)
     }
myFunction();

    document.write('My pet name is ' + petName +
      '<br> Fruit name is ' +       fruit)
</script>
```

Example 4

```
<script>
  myfunction();
  anotherFunc();
  var petName;
  function myfunction() {
    var petName = "Sizzer"; // local variable
    document.write(petName);   }
  function anotherFunc() {
    var petName = "Tom"; // local variable
    document.write(petName);
  }
  document.write(petName);
</script>
```