JavaScript Functions

A function is a block of code written specifically to complete a specific task. It usually accepts data and then returns a result. For a function to execute, it must be invoked or called.

Syntax

To define a JavaScript Function, the *function*  keyword is used. It is then followed by name and a set of parenthesis. Function names are allowed to have letters, digits, dollar signs, and underscores. Parameters are included in the parantheses, separated by commas. The code to be executed is then placed inside the curly braces.

function name(parameter1, parameter2){

Body

}

**Arrow Functions**

Arrow functions, with a shorter syntax than a function expression, are best used for non-method functions. It does not support this, arguments, super or new.target of its own and are not used as constructors.

Syntax

(parameter1, parameter2, parameterN) => { statements }

(parameter1, parameter2, parameterN) => expression

//equivalent to => { return expression; }

* Arrow functions do not need a separate this value, which is a new object of a constructor. This is ideal for an object-oriented style of programming because without the need to newly define its own this value, the arrow function makes use of the this value of the enclosing execution context.
* Because the this value comes from a lexical context, arrow functions are able to ignore strict mode rules that are related to this.
* The methods call() or apply() are accepted as long as they pass in parameters and thisArg is ignored.
* arguments is only referenced from the arguments of the enclosing scope since arrow functions do not have their own arguments object.
* The prototype property is not supported by arrow functions.
* Arrow functions do not have support to be used as generators.

**Nested Functions**

JavaScript variables can either be local or global in scope. By using closures, global variables can be made into local or private variables. A function is both able to gain access to a variable that is declared inside and outside the function. A global variable, defined without the keyword var, can be accessed and used by scripts throughout the page or window while a local variable can only be used by the function to which it was declared.

With nested functions, it is possible to have access to the scope that was already created above, before the function was defined. With nested functions, the dilemma of needing a global variable that a function can access but not be disturbed by other functions is solved.