STATEMENTS AND FUNCTIONS IN JAVASCRIPT

Introduction of JavaScript Statements

JavaScript statements are composed of:
Values, Operators, Expressions, Keywords and Comments
This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":
Example:

```
document.getElementById("demo").innerHTML = "Hello Dolly.";
```

Most JavaScript programs contain many JavaScript statements.
The statements are executed, one by one, in the same order as they are written.
JavaScript programs (and JavaScript statements) are often called JavaScript code.

Semicolons:

Semicolons separate JavaScript statements.
Add a semicolon at the end of each executable statement:
Examples

```
let a, b, c;  // Declare 3 variables
a = 5;       // Assign the value 5 to a
b = 6;       // Assign the value 6 to b
c = a + b;    // Assign the sum of a and b to c
```

When separated by semicolons, multiple statements on one line are allowed:

```
a = 5; b = 6; c = a + b;
```

JavaScript White Space
JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.
The following lines are equivalent:

```
let person = "Hege";
let person="Hege";
```

A good practice is to put spaces around operators (= + - * /):

```
let x = y + z;
```

JavaScript Line Length and Line Breaks
For best readability, programmers often like to avoid code lines longer than 80 characters.
If a JavaScript statement does not fit on one line, the best place to break it is after an operator:
Example

```
document.getElementById("demo").innerHTML =
"Hello Dolly!";
```

JavaScript Code Blocks

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.
The purpose of code blocks is to define statements to be executed together.
One place you will find statements grouped together in blocks, is in JavaScript functions:

Example

```
function myFunction() {
  document.getElementById("demo1").innerHTML = "Hello Dolly!";
  document.getElementById("demo2").innerHTML = "How are you?";
}
```

JavaScript Keywords

JavaScript statements often start with a **keyword** to identify the JavaScript action to be performed.
Our Reserved Words Reference lists all JavaScript keywords.
Here is a list of some of the keywords you will learn about in this tutorial:

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

JavaScript function

Definition and Usage

The function statement declares a function.
A declared function is "saved for later use", and will be executed later, when it is invoked (called).
In JavaScript, functions are objects, and they have both properties and methods.
A function can also be defined using an expression (See Function Definitions).
Read our JavaScript Tutorial to learn all you need to know about functions. Start with the introduction chapter about JavaScript Functions and JavaScript Scope. For more detailed information, see our Function Section on Function
Definitions, Parameters, Invocation and Closures.
Syntax

```
function functionName(parameters) {
  code to be executed
}
```

Parameters

Parameter	Description
functionName	Required. The name of the function. Naming rules: same as JavaScript variables.
parameters	Optional. A set of arguments (parameter names), separated by commas. The arguments are real values received by the function from the outside. Inside the function, the arguments are used as local variables. If a function is called with a missing argument, the value of the missing argument is set to undefined.

More Examples:

Return the value of PI:

```
function myFunction() {
  return Math.PI;
}
```

Return the product of a and b:

```
function myFunction(a, b) {
  return a * b;
}
```

A function with different arguments can produce different results.
Convert Fahrenheit to Celsius:

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
```

Functions can be used as variables.
Instead of:

```
temp = toCelsius(32);
text = "The temperature is " + temp + " Centigrade";
```

You can use:

```
text = "The temperature is " + toCelsius(32) + " Centigrade";
```

JavaScript functions have a built-in object called arguments.
The arguments.length property returns the number of arguments received by the function:

```
function myFunction(a, b) {
  return arguments.length;
}
```

Click to call a function that outputs "Hello World":

```
<button onclick="myFunction()">Click me</button>

<p id="demo"></p>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Hello World";
}
</script>
```

When a function expression is stored in a variable, the variable contains a function:

```
const x = function (a, b) {return a * b};
```

When a function is stored in a variable, the variable can be used as a function:

```
const x = function (a, b) {return a * b};
let z = x(4, 3);
```