JavaScript Function Parameters

‹ Previous Next ›

A JavaScript function does not perform any checking on parameter values (arguments).

Function Parameters and Arguments

Earlier in this tutorial, you learned that functions can have **parameters**:

```
functionName(parameter1, parameter2, parameter3) {
    code to be executed
}
```

Function **parameters** are the **names** listed in the function definition.

Function **arguments** are the real **values** passed to (and received by) the function.

Parameter Rules

JavaScript function definitions do not specify data types for parameters.

JavaScript functions do not perform type checking on the passed arguments.

JavaScript functions do not check the number of arguments received.

Parameter Defaults

If a function is called with **missing arguments** (less than declared), the missing values are set to: **undefined**

Sometimes this is acceptable, but sometimes it is better to assign a default value to the parameter:

Example

```
function myFunction(x, y) {
    if (y === undefined) {
          y = 0;
    }
}
```

Try it Yourself »

ECMAScript 2015 allows default parameters in the function call:

```
function (a=1, b=1) { // function code }
```

The Arguments Object

JavaScript functions have a built-in object called the arguments object.

The argument object contains an array of the arguments used when the function was called (invoked).

This way you can simply use a function to find (for instance) the highest value in a list of numbers:

Example

```
x = findMax(1, 123, 500, 115, 44, 88);

function findMax() {
    var i;
    var max = -Infinity;
    for (i = 0; i < arguments.length; i++) {
```

```
        if (arguments[i] > max) {
            max = arguments[i];
        }
    }
    return max;
}
```

Try it Yourself »

Or create a function to sum all input values:

Example

```
x = sumAll(1, 123, 500, 115, 44, 88);

function sumAll() {
    var i;
    var sum = 0;
    for (i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }
    return sum;
}
```

Try it Yourself »

If a function is called with **too many arguments** (more than declared), these arguments can be reached using **the arguments object**.

Arguments are Passed by Value

The parameters, in a function call, are the function's arguments.

JavaScript arguments are passed by **value**: The function only gets to know the values, not the argument's locations.

If a function changes an argument's value, it does not change the parameter's original value.

Changes to arguments are not visible (reflected) outside the function.

Objects are Passed by Reference

In JavaScript, object references are values.

Because of this, objects will behave like they are passed by **reference:**

If a function changes an object property, it changes the original value.

Changes to object properties are visible (reflected) outside the function.

‹ Previous

Next ›