# Parâmetros da função JavaScript

[❮ Anterior](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_function_definition.asp&usg=ALkJrhgFrOUXIbhiFb8cT5eBg5qP9fugJg)[Próximo ❯](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/js_function_invocation.asp&usg=ALkJrhhDiwdwIx0ehevRi-vdgDVVIo7LRw)

Uma função JavaScript não executa nenhuma verificação nos valores dos parâmetros (argumentos).

## Parâmetros de função e argumentos

No início deste tutorial, você aprendeu que as funções podem ter **parâmetros** :

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

Os **parâmetros de** função são os **nomes** listados na definição da função.

Os **argumentos de** função são os **valores** reais passados ​​para (e recebidos por) a função.

## Regras de parâmetros

As definições de função JavaScript não especificam tipos de dados para parâmetros.

As funções JavaScript não executam verificação de tipo nos argumentos passados.

As funções de JavaScript não conferem o número de argumentos recebidos.

## Padrões de Parâmetros

Se uma função é chamada com **argumentos ausentes** (menor do que declarado), os valores ausentes são definidos para: **indefinido**

Às vezes, isso é aceitável, mas às vezes é melhor atribuir um valor padrão ao parâmetro:

### Exemplo

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

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_undefined&usg=ALkJrhg25P9dZuts2a3aCfo7c2Ehm5d3nA)

Se uma função for chamada com **muitos argumentos** (mais do que declarado), esses argumentos podem ser alcançados usando **o objeto de argumentos** .

## O Objeto Argumentos

As funções de JavaScript possuem um objeto interno chamado objeto de argumentos.

O objeto argumento contém uma matriz dos argumentos usados ​​quando a função foi chamada (invocada).

Desta forma, você pode simplesmente usar uma função para encontrar (por exemplo) o valor mais alto em uma lista de números:

### Exemplo

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;  
}

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_arguments&usg=ALkJrhjQS7ei23vvo1xOnwnpHPzY3g8ADA)

Ou crie uma função para somar todos os valores de entrada:

### Exemplo

x = sumAll(1, 123, 500, 115, 44, 88);  
  
function sumAll() {  
    var i, sum = 0;  
    for (i = 0; i < arguments.length; i++) {  
        sum += arguments[i];  
    }  
    return sum;  
}

[Tente você mesmo "](https://translate.googleusercontent.com/translate_c?depth=1&hl=pt-BR&ie=UTF8&prev=_t&rurl=translate.google.com.br&sl=en&sp=nmt4&tl=pt-BR&u=https://www.w3schools.com/js/tryit.asp%3Ffilename%3Dtryjs_function_arguments_sum&usg=ALkJrhj4Jv1khhIld8Aq0rIxCQLteiuvYQ)

## Argumentos são passados ​​por valor

Os parâmetros, em uma chamada de função, são os argumentos da função.

Os argumentos de JavaScript são passados ​​por **valor** : a função só conhece os valores, não os locais do argumento.

Se uma função altera o valor de um argumento, não altera o valor original do parâmetro.

**As alterações nos argumentos não são visíveis (refletidas) fora da função.**

## Os objetos são aprovados por referência

Em JavaScript, as referências de objetos são valores.

Por causa disso, os objetos se comportarão como se fossem aprovados por **referência:**

Se uma função muda uma propriedade de objeto, ela muda o valor original.

**Alterações nas propriedades do objeto são visíveis (refletidas) fora da função.**