연산자 데이터에 규칙을 적용해서 상태를 변경시키는 약속된 기호

수학연산자.

할당연산자.

비교연산자.

논리연산자.

산술연산자.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers (literals or variables). 수학 수행하다. 연산을 하다.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + | Addition |
| - | Subtraction |
| \* | Multiplication |
| / | Division |
| % | Modulus |
| ++ | Increment |
| -- | Decrement |

The **addition** operator (+) adds numbers:

var x = 100 + 50;

var x = a + b;

var x = (100 + 50) \* a;

|  |  |  |
| --- | --- | --- |
| **Operand** | **Operator** | **Operand** |
| 100 | + | 50 |

The **addition** operator (+) adds numbers:

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Example** | **Same As** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |

JavaScript Comparison and Logical Operators

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | equal to |
| === | equal value and equal type |
| != | not equal |
| !== | not equal value or not equal type |
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |

우선순위

JavaScript Operator Precedence Values

|  |  |  |  |
| --- | --- | --- | --- |
| **Value** | **Operator** | **Description** | **Example** |
| 19 | ( ) | Expression grouping | (3 + 4) |
|  |  |  |  |
| 18 | . | Member | person.name |
| 18 | [] | Member | person["name"] |
|  |  |  |  |
| 17 | () | Function call | myFunction() |
| 17 | new | Create | new Date() |
|  |  |  |  |
| 16 | ++ | Postfix Increment | i++ |
| 16 | -- | Postfix Decrement | i-- |
|  |  |  |  |
| 15 | ++ | Prefix Increment | ++i |
| 15 | -- | Prefix Decrement | --i |
| 15 | ! | Logical not | !(x==y) |
| 15 | typeof | Type | typeof x |
|  |  |  |  |
| 14 | \* | Multiplication | 10 \* 5 |
| 14 | / | Division | 10 / 5 |
| 14 | % | Modulo division | 10 % 5 |
| 14 | \*\* | Exponentiation | 10 \*\* 2 |
|  |  |  |  |
| 13 | + | Addition | 10 + 5 |
| 13 | - | Subtraction | 10 - 5 |
|  |  |  |  |
| 12 | << | Shift left | x << 2 |
| 12 | >> | Shift right | x >> 2 |
|  |  |  |  |
| 11 | < | Less than | x < y |
| 11 | <= | Less than or equal | x <= y |
| 11 | > | Greater than | x > y |
| 11 | >= | Greater than or equal | x >= y |
|  |  |  |  |
| 10 | == | Equal | x == y |
| 10 | === | Strict equal | x === y |
| 10 | != | Unequal | x != y |
| 10 | !== | Strict unequal | x !== y |
|  |  |  |  |
| 6 | && | And | x && y |
| 5 | || | Or | x || y |
|  |  |  |  |
| 3 | = | Assignment | x = y |
| 3 | += | Assignment | x += y |
| 3 | -= | Assignment | x -= y |
| 3 | \*= | Assignment | x \*= y |
| 3 | /= | Assignment | x /= y |

아래 예제를 풀어보자.

<http://www.w3schools.com/js/js_arithmetic.asp>

## The typeof Operator

You can use the JavaScript **typeof** operator to find the type of a JavaScript variable:

### **Example**

typeof "John"                // Returns string   
typeof 3.14                  // Returns number  
typeof false                 // Returns boolean  
typeof [1,2,3,4]             // Returns object  
typeof {name:'John', age:34} // Returns object

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_typeof)

|  |  |
| --- | --- |
| **Note** | In JavaScript, an array is a special type of object. Therefore typeof [1,2,3,4] returns object. |

## Undefined

In JavaScript, a variable without a value, has the value**undefined**. The typeof is also **undefined**.

### **Example**

var person;                  // Value is undefined, type is undefined

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_undefined)

Any variable can be emptied, by setting the value to **undefined**. The type will also be **undefined**.

### **Example**

person = undefined;          // Value is undefined, type is undefined

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_undefined_2)

## Empty Values

An empty value has nothing to do with undefined.

An empty string variable has both a value and a type.

### **Example**

var car = "";                // The value is "", the typeof is string

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_empty)

## Null

In JavaScript null is "nothing". It is supposed to be something that doesn't exist.

Unfortunately, in JavaScript, the data type of null is an object.

|  |  |
| --- | --- |
| **Note** | You can consider it a bug in JavaScript that typeof null is an object. It should be null. |

You can empty an object by setting it to null:

### **Example**

var person = null;           // Value is null, but type is still an object

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_null)

You can also empty an object by setting it to undefined:

### **Example**

var person = undefined;     // Value is undefined, type is undefined

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_datatypes_undefined_1)

## Difference Between Undefined and Null

typeof undefined             // undefined  
typeof null                  // object  
null === undefined           // false  
null == undefined            // true

[Try it Yourself »](http://www.w3schools.com/js/tryit.asp?filename=tryjs_typeof_undefined_3)