

Операторы JavaScript



- Выражения
- Приоритеты операций
- Ассоциативность
- Арифметические операторы
- Равенство и идентичность
- Условные операторы
- Строковые операторы
- Битовые операторы
- Оператор присваивания
- Прочие операторы

Выражения



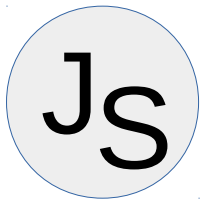
Литералы

```
1.7
"JavaScript is fun!"
true
null
/java/
{x:2, y:2}
[2,3,5,7,11,13,17,19]
function(x){return x*x;}
i
sum
```

Выражения

```
i + 1.7
(i + 1.7) - sum
strlen(app.generatePassword(10)) + 1
```

Операторы



.	свойство объекта
[]	элемент массива
()	вызов функции
new	конструктор
++	инкремент
--	декремент
-	унарный минус
+	унарный плюс
~	побитовое дополнение
!	логическое дополнение
delete	удаление свойства
typeof	тип данных
void	неопредел. значение
* / %	умножение, деление, остаток
+ -	сложение, вычитание
+	конкатенация

<< >>	сдвиг влево, вправо
>>>	сдвиг вправо с доп. нулём
< >	больше/меньше
>= <=	больше/меньше либо равно
instanceof	проверка класса
in	существует ли свойство
== !=	(не)равенство
=== !==	(не)идентичность
& ^	побитовое и/или/искл. или
&&	логическое и/или
?:	тернарный условный оператор
=	присваивание
*= /= %= += -= <<= >>= >>>= &=^= =	присваивание с операцией
,	список (множественная оценка)

Классификация операторов



- По количеству операндов
 - унарный
 - бинарный
 - тернарный
- По типам операндов

```
'a' * 'b'
```

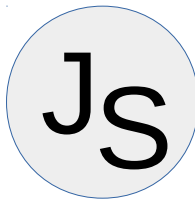
```
'2' * '3'
```

```
5 + 3
```

```
'5' + '3'
```

```
5 + '3'
```

Приоритет. Ассоциативность.



- Приоритет

```
w = x + y * z;  
w = (x + y) * z;
```

- Ассоциативность

```
w = x + y + z;           // w = (x + y) + z;
```

```
x = ~-~y;                 // x = ~( -(~y) );
```

```
w = x = y = z;           // w = (x = (y = z));
```

```
q = a?b:c?d:e?f:g;       // q = a?b:(c?d:(e?f:g));
```

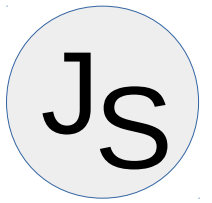
Арифметические операторы.



+	сложение, конкатенация	valueOf(), toString()
-	вычитание	
*	умножение	
/	деление	
%	остаток	
-	унарный минус	
+	унарный плюс	
++	инкремент	
--	декремент	

```
i=1;  
x = ++i;  
y = i++;
```

Равенство и идентичность



= присваивание

```
w = "строка";
```

=== ИДЕНТИЧНОСТЬ

```
5 === '5' // false
```

```
'понедельник' === 'понедельник' // true
```

```
NaN === NaN // false
```

```
false === false // true
```

```
[1,2] === [1,2] // false
```

```
a = b = new Date(); a === b; // true
```

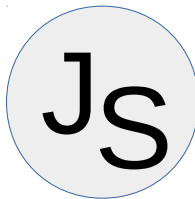
```
a=new Date(); b=new Date(); a===b; // false
```

```
null === null // true
```

```
undefined === undefined // true
```

```
false !== false // false
```

Равенство и идентичность



== равенство

```
null == undefined           // true
17 == '17'                   // true
false == 0                    // true
true == '1'                   // true
new Number(55) == '55';      // true

new Date(1500000000000) ==
    'Fri Jul 14 2017 05:40:00 GMT+0300 (MSK)'
                                // true

[1,2,3] == [1,2,3]           // false
[1,2,3] != [1,2,3]           // true
```


Операторы отношений

JS

Сравнение < > <= >=

```
52 >= '13' //true
new Date() > new Date('01-01-2016') //true
[1,2,3] <= 17 //false
```

in

```
var point = { x:1, y:1 };
var has_x_coord = "x" in point; // true
var has_y_coord = "y" in point; // true
var has_z_coord = "z" in point; // false
var ts = "toString" in point; // true (насл.)
```

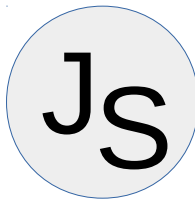
Операторы отношений



instanceof

```
var d = new Date();  
d instanceof Date;    // true  
d instanceof Object;  // true  
d instanceof Number;  // false  
var a = [1, 2, 3];  
a instanceof Array;    // true  
a instanceof Object;  // true  
a instanceof RegExp;   // false
```

Строковые операторы

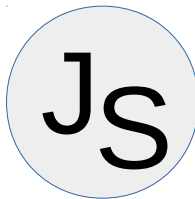


Конкатенация +

```
"hello" + " " + "there"           // "hello there"  
a = "2"; b = "2"; c = a + b;      // 22
```

```
1 + 2           // 3  
"1" + "2"       // "12"  
"1" + 2         // "12"  
11 < 3          // false  
"11" < "3"      // true  
"11" < 3        // false  
"one" < 3       // false (NaN < 3)
```

Логические операторы



И && ИЛИ ||

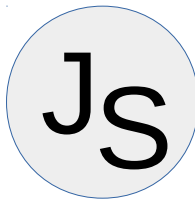
```
0 && true           // 0
true && '15'         // '15'
```

```
if (a == b) stop( );
(a == b) && stop( );
```

НЕ !

```
!0                 // true
!!'177'           // true
```

Битовые операторы



И & ИЛИ | исключающее ИЛИ ^ НЕ ~

```
0x1234 & 0x00FF          // 0x0034
9 | 10                    // 11
9 ^ 10                     // 3
~0x0f                      // 0xfffffffff0 (-16)
```

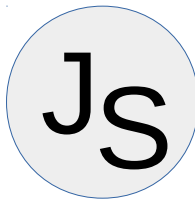
влево <<

вправо со знаком >>

вправо с заполнением нулями >>>

```
7 << 1                     // 14
7 >> 1                      // 3
-7 >> 1                     // -4
-1 >>> 4                    // 0xfffffffff
```

Операторы присваивания



=

```
(a = b) == 0  
i = j = k = 0;
```

+=

*=

<<=

&=

-=

/=

>>=

|=

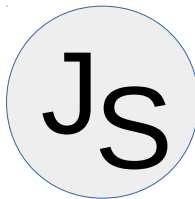
%=

>>>=

^=

```
total += sales_tax;  
total = total + sales_tax;
```

Другие операторы



Условный оператор ?:

```
x > 0 ? x*y : -x*y  
greeting = "hello " +  
    (username != null ? username : "there");
```

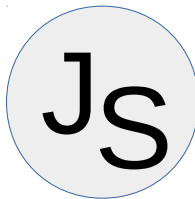
typeof

```
typeof i  
(typeof value == "string") ? "'" + value + "'" : value
```

new

```
o = new Object;  
d = new Date( );  
c = new Rectangle(3.0, 4.0, 1.5, 2.75);  
obj[i] = new constructors[i]( );
```

Другие операторы



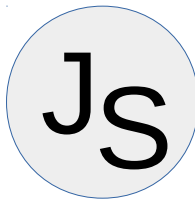
delete

```
var o = {x:1, y:2};  
delete o.x;           // true  
typeof o.x;           // "undefined"  
delete o.x;           // true  
delete o;              // false  
delete 1;              // true (!)  
x = 1;  
delete x;              // true  
x;                     // Runtime error: x is not defined
```

void

```
void window.open()    // undefined
```


Другие операторы



Список ,

```
i=0, j=1, k=2;  
for ( , , )
```

Доступ к элементу массива и свойству объекта

```
document.lastModified  
document['lastModified']  
frames[0].length  
document.write("hello world")  
data['val'+i]
```

Вызов функции ()

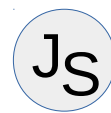
```
document.close( )  
Math.sin(x)  
alert("Welcome " + name)  
Date.UTC(2000, 11, 31, 23, 59, 59)  
funcs[i].f(funcs[i].args[0], funcs[i].args[1])
```

Операторы JavaScript



- Выражения
- Приоритеты операций
- Ассоциативность
- Арифметические операторы
- Равенство и идентичность
- Условные операторы
- Строковые операторы
- Битовые операторы
- Оператор присваивания
- Прочие операторы

Выражения



Литералы

```
1.7
"JavaScript is fun!"
true
null
/java/
{x:2, y:2}
[2,3,5,7,11,13,17,19]
function(x){return x*x;}
i
sum
```

Выражения

```
i + 1.7
(i + 1.7) - sum
strlen(app.generatePassword(10)) + 1
```

Операторы



.	свойство объекта
[]	элемент массива
()	вызов функции
new	конструктор
++	инкремент
--	декремент
-	унарный минус
+	унарный плюс
~	побитовое дополнение
!	логическое дополнение
delete	удаление свойства
typeof	тип данных
void	неопредел. значение
* / %	умножение, деление, остаток
+ -	сложение, вычитание
+	конкатенация

<< >>	сдвиг влево, вправо
>>>	сдвиг вправо с доп. нулём
< >	больше/меньше
>= <=	больше/меньше либо равно
instanceof	проверка класса
in	существует ли свойство
== !=	(не)равенство
=== !==	(не)идентичность
& ^	побитовое и/или/искл. или
&&	логическое и/или
?:	тернарный условный оператор
=	присваивание
*= /= %= += -= <<= >>= >>>= &=^= =	присваивание с операцией
,	список (множественная оценка)

Классификация операторов



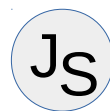
- По количеству операндов

- унарный
- бинарный
- тернарный

- По типам операндов

```
'a' * 'b'  
'2' * '3'  
5 + 3  
'5' + '3'  
5 + '3'
```

Приоритет. Ассоциативность.



- Приоритет

```
w = x + y * z;  
w = (x + y) * z;
```

- Ассоциативность

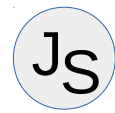
```
w = x + y + z;           // w = (x + y) + z;
```

```
x = ~~~y;                // x = ~(~(~y));
```

```
w = x = y = z;           // w = (x = (y = z));
```

```
q = a?b:c?d:e?f:g;       // q = a?b:(c?d:(e?f:g));
```

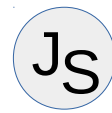
Арифметические операторы.



+	сложение, конкатенация	valueOf(), toString()
-	вычитание	
*	умножение	
/	деление	
%	остаток	
-	унарный минус	
+	унарный плюс	
++	инкремент	
--	декремент	

```
i=1;  
x = ++i;  
y = i++;
```

Равенство и идентичность



= присваивание

```
w = "строка";
```

=== идентичность

```
5 === '5' // false
'понедельник' === 'понедельник' // true
NaN === NaN // false
false === false // true
[1,2] === [1,2] // false
a = b = new Date(); a === b; // true
a=new Date(); b=new Date(); a===b; // false
null === null // true
undefined === undefined // true
false !== false // false
```


Равенство и идентичность

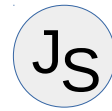


== равенство

```
null == undefined           // true
17 == '17'                   // true
false == 0                    // true
true == '1'                   // true
new Number(55) == '55';      // true

new Date(15000000000000) ==
  'Fri Jul 14 2017 05:40:00 GMT+0300 (MSK)'
                                // true
[1,2,3] == [1,2,3]            // false
[1,2,3] != [1,2,3]            // true
```

Операторы отношений



Сравнение < > <= >=

```
52 >= '13' //true
new Date() > new Date('01-01-2016') //true
[1,2,3] <= 17 //false
```

in

```
var point = { x:1, y:1 };
var has_x_coord = "x" in point; // true
var has_y_coord = "y" in point; // true
var has_z_coord = "z" in point; // false
var ts = "toString" in point; // true (насл.)
```

Порядок сравнения

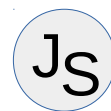
- два числа
- две строки (Unicode)
- число и строка -> 2 числа
- false

For a more robust string comparison algorithm, see the

`String.localeCompare()` method, which also takes locale-specific definitions of "alphabetical order" into account. For case-insensitive comparisons, you must first convert the strings to all lowercase or all uppercase using `String.toLowerCase()` or `String.toUpperCase()`.

The `<=` (less-than-or-equal) and `>=` (greater-than-or-equal) operators do not rely on the equality or identity operators for determining whether two values are "equal." Instead, the less-than-or-equal operator is simply defined as "not greater than," and the greater-than-or-equal operator is defined as "not less than." The one exception is when either operand is (or converts to) NaN, in which case all four comparison operators return false.

Операторы отношений



instanceof

```
var d = new Date();
d instanceof Date;    // true
d instanceof Object;  // true
d instanceof Number;  // false
var a = [1, 2, 3];
a instanceof Array;   // true
a instanceof Object;  // true
a instanceof RegExp;  // false
```

Строковые операторы

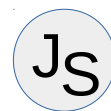


Конкатенация +

```
"hello" + " " + "there"      // "hello there"
a = "2"; b = "2"; c = a + b;  // 22
```

```
1 + 2      // 3
"1" + "2"  // "12"
"1" + 2     // "12"
11 < 3      // false
"11" < "3"  // true
"11" < 3    // false
"one" < 3   // false (NaN < 3)
```

Логические операторы



И && ИЛИ ||

```
0 && true           // 0
true && '15'         // '15'
```

```
if (a == b) stop( );
(a == b) && stop( );
```

НЕ !

```
!0                  // true
!!'177'             // true
```

Битовые операторы



И & ИЛИ | исключающее ИЛИ ^ НЕ ~

```
0x1234 & 0x00FF // 0x0034
9 | 10 // 11
9 ^ 10 // 3
~0x0f // 0xffffffff0 (-16)
```

влево <<

вправо со знаком >>

вправо с заполнением нулями >>>

```
7 << 1 // 14
7 >> 1 // 3
-7 >> 1 // -4
-1 >>> 4 // 0x0fffffff
```

Операторы присваивания



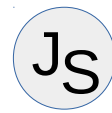
=

```
(a = b) == 0  
i = j = k = 0;
```

+=	*=	<<=	&=
-=	/=	>>=	=
	%=	>>>=	^=

```
total += sales_tax;  
total = total + sales_tax;
```

Другие операторы



Условный оператор ?:

```
x > 0 ? x*y : -x*y  
greeting = "hello " +  
    (username != null ? username : "there");
```

typeof

```
typeof i  
(typeof value == "string") ? "" + value + "" : value
```

new

```
o = new Object;  
d = new Date( );  
c = new Rectangle(3.0, 4.0, 1.5, 2.75);  
obj[i] = new constructors[i]( );
```


Другие операторы



delete

```
var o = {x:1, y:2};
delete o.x;           // true
typeof o.x;           // "undefined"
delete o.x;           // true
delete o;              // false
delete 1;              // true (!)
x = 1;
delete x;              // true
x;                     // Runtime error: x is not defined
```

void

```
void window.open()    // undefined
```

Другие операторы



Список ,

```
i=0, j=1, k=2;  
for ( , , )
```

Доступ к элементу массива и свойству объекта

```
document.lastModified  
document['lastModified']  
frames[0].length  
document.write("hello world")  
data['val'+i]
```

Вызов функции ()

```
document.close( )  
Math.sin(x)  
alert("Welcome " + name)  
Date.UTC(2000, 11, 31, 23, 59, 59)  
funcs[i].f(funcs[i].args[0], funcs[i].args[1])
```