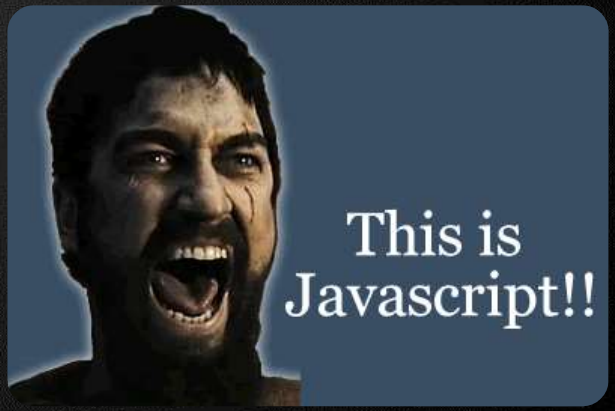
Telerik Academy

Operators and Expressions

Performing Simple Calculations with JavaScript



Javascript Fundamentals

Telerik Software Academy

https://telerikacademy.com

Follow us

Telerik Academy

Table of Contents

Telerik Academy

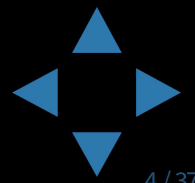
Operators in JavaScript

Arithmetic, Logical, Comparison, Assignment, Etc.

Telerik Academy

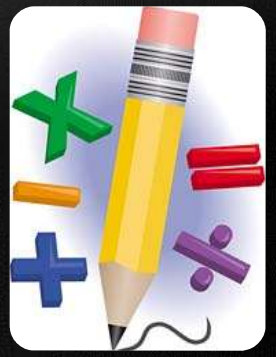
What is an Operator?

- An operator is a symbol that represents an operation performed over data at runtime
  - Takes one or more arguments (operands)
  - Produces a new value
- Operators have precedence (priority)
  - Precedence defines which will be evaluated first
- Expressions are sequences of operators and operands that are evaluated to a single value



Telerik Academy

Operators in JavaScript



- Operators in JavaScript :
  - Unary – take one operand
  - Binary – take two operands
  - Ternary (? :) – takes three operands
- Except for the assignment operators, all binary operators are left-associative
- The assignment operators and the conditional operator (? :) are right-associative

Telerik Academy

Operators by categories in JavaScript

Category	Operators
Arithmetic	`+ - * / % ++ --`
Logical	`&& || ^ !`
Binary	`& | ^ ~ << >> >>>`
Comparison	`== != < > <= >= === !==`
Assignment	`= += -= *= /= %= |= ^= <<= >>=`
Concatenation	`+`
Other	`. [] () ?: new in , delete void typeof instanceof ...`

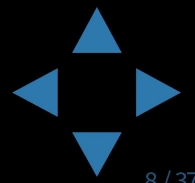
Telerik Academy

Arithmetic Operators

Telerik Academy

Arithmetic Operators

- Arithmetic operators +, -, *, / are the same as in math
- Division operator / returns number or `Infinity` or `NaN`
  - Division in JavaScript returns floating-point number (i.e. 5 / 2 = 2.5)
- Remainder operator % returns the remainder from division of numbers
  - Even on real (floating-point) numbers
- The special addition operator ++ increments (while -- decrements) a variable's value



Arithmetic Operators – *Example*

```
const squarePerimeter = 17;
const squareSide = squarePerimeter / 4;
const squareArea = squareSide * squareSide;

console.log(squareSide); // 4.25
console.log(squareArea); // 18.0625

let a = 5;
let b = 4;

console.log(a + b); // 9
console.log(a + b++); // 9
console.log(a + b); // 10
console.log(a + (++b)); // 11
console.log(a + b); // 11

console.log(12 / 3); // 4
console.log(11 / 3); // 3.6666666666666665
```

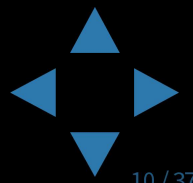
Telerik Academy

Arithmetic Operators – *Example*

```
console.log(11 % 3);   // 2
console.log(11 % -3);  // 2
console.log(-11 % 3);  // -2

console.log(1.5 / 0.0);  // Infinity
console.log(-1.5 / 0.0); // -Infinity
console.log(0.0 / 0.0);  // NaN

const x = 0;
console.log(5 / x);
```

Telerik Academy

Arithmetic Operators

Demo

Telerik Academy

Logical Operators

Telerik Academy

Logical Operators

- Logical operators take boolean operands and return boolean result
- Operator `!` turns `true` to `false` and `false` to `true`
- Behavior of the operators `&&`, `||` and `^` (`1` == `true`, `0` == `false`) :

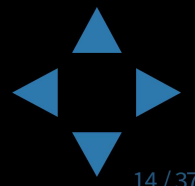
Operation	||	||	||	||	&&	&&	&&	&&	^	^	^	^
Operand1	0	0	1	1	0	0	1	1	0	0	1	1
Operand2	0	1	0	1	0	1	0	1	0	1	0	1
Result	0	1	1	1	0	0	0	1	0	1	1	0

Logical Operators – *Example*

- Using the logical operators:

```
let a = true;
let b = false;

console.log(a && b); // False
console.log(a || b); // True
console.log(a ^ b); // True
console.log(!b); // True
console.log(b || true); // True
console.log(b && true); // False
console.log(a || true); // True
console.log(a && true); // True
console.log(!a); // False
console.log((5 > 7) ^ (a == b)); // False
```

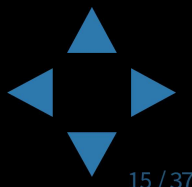


Telerik Academy

true-like and false-like values

- JavaScript, as a weakly typed language, can use every value as `true` or `false`
- Every value can be converted to it's boolean representation using double not - `!!`

```
console.log(
  !!'', // empty string is false-like
  !!'0', // non-empty strings are true-like
  !!0, // zero is false-like
  !!35, // non-zero numbers are true-like
  !![], // objects are true-like
  !!NaN, // NaN is false-like
  !!'true', // true
  !!'false' // true,
  !!null, // both null and undefined are false-like
  !!undefined
);
```

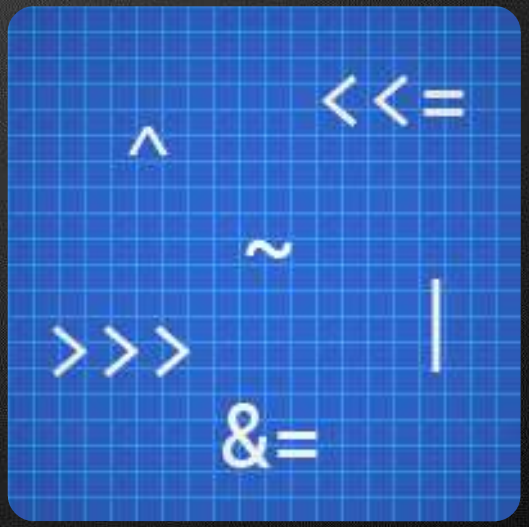
Telerik Academy

Logical Operators

Demo

Telerik Academy

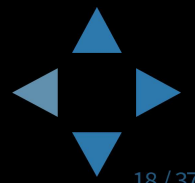
Bitwise Operators



Telerik Academy

Bitwise Operators

- Bitwise operators are used on integer numbers. They are applied bit by bit.
- Operator ~ turns all `0` to `1` and all `1` to `0`
  - Like ! for boolean expressions but bit by bit
- The operators |, & and ^ behave like ||, && and ^ for boolean expressions but bit by bit
- The << and >> shift (move) the bits to the left or to the right



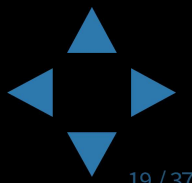
Telerik Academy

Bitwise Operators - *Examples*:

```
let a = 3;                  // 00000000 00000011
let b = 5;                  // 00000000 00000101

console.log(a | b);         // 00000000 00000111
console.log(a & b);         // 00000000 00000001
console.log(~a & b);        // 00000000 00000100
console.log(a ^ b);         // 00000000 00000110

console.log(true << 1);     // 00000000 00000010
console.log(true >> 1);     // 00000000 00000000
```

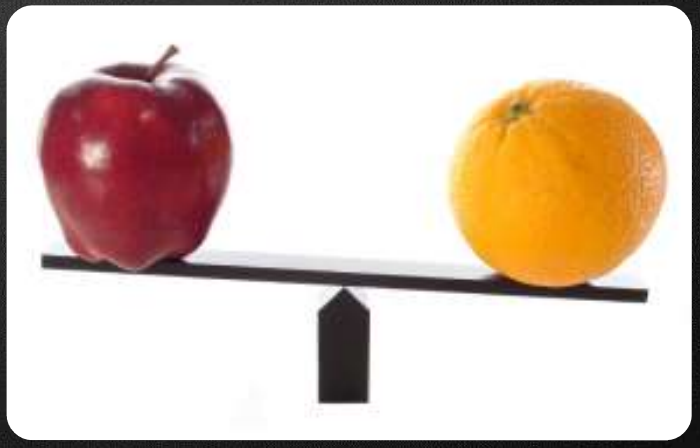
Telerik Academy

Bitwise Operators

Demo

Telerik Academy

Comparison and Assignment Operators



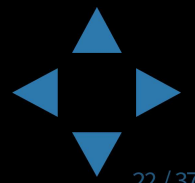
Telerik Academy

Comparison Operators

- Comparison operators are used to compare variables
  - ==, <, >, >=, <=, !=,===,!==
  - For equality comparison, the use of === and !== is preferred

```
let a = 5;
let b = 4;

console.log(a >= b);    // True
console.log(a != b);    // True
console.log(a == b);    // False
console.log(0 == '');   // True
console.log(0 === ''); // False
```

Telerik Academy

Assignment Operators

- Assignment operators are used to assign a value to a variable
  - =, +=, -=, |=, …

```
let x = 6;
let y = 4;

console.log(y *= 2); // 8

let z = y = 3;       // y=3 and z=3

console.log(z);      // 3
console.log(x |= 1); // 7
console.log(x += 3); // 10
console.log(x /= 2); // 5
```

Telerik Academy

Comparison and Assignment Operators

Demo

Telerik Academy

Other Operators

Telerik Academy

Other Operators

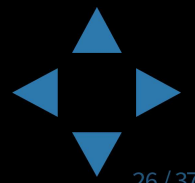
- String concatenation operator + is used to concatenate strings
- If the second operand is not a string, it is converted to string automatically

```
let first = "First";
let second = "Second";

console.log(first + second); // FirstSecond

let output = "The number is : ";
let number = 5;

console.log(output + number); // The number is : 5
```



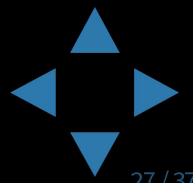
Telerik Academy

Other Operators

- Member access operator `.` is used to access object members
- Square brackets `[]` are used as indexers, to access a member with a certain name
- Parentheses `()` are used to override the default operator precedence or to invoke functions
- Conditional operator `?:` has the form
  - (if `b` is true then the result is `x` else the result is `y`)

```
b ? x : y
```

- The `new` operator is used to create new objects
- The `typeof` operator returns the type of the value



Telerik Academy

Other Operators – *Example*

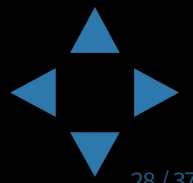
- Using some other operators:

```
let a = 6;
let b = 4;

console.log(a > b ? 'a > b' : 'b >= a'); // a > b

let c = b = 3; // b = 3; followed by c = 3;

console.log(c); // 3
console.log(new Number(6) instanceof Number); // true
console.log(6 instanceof Number); // false
console.log((a + b) / 2); // 4
console.log(typeof c); // number
console.log(void(3 + 4)); // undefined
```

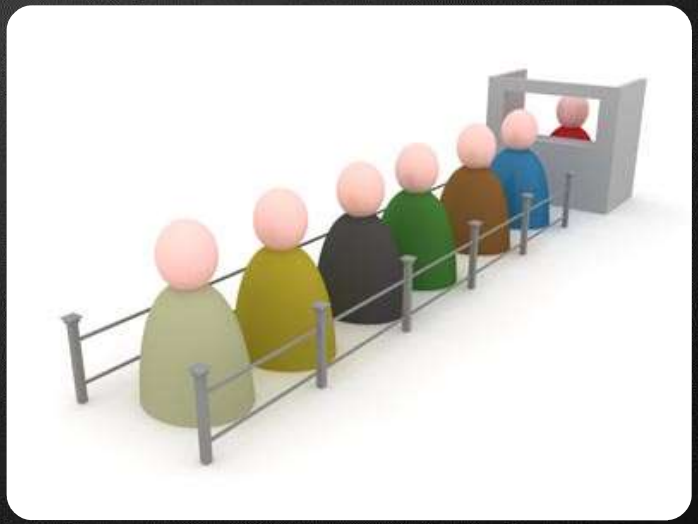
Telerik Academy

Other Operators

Demo

Telerik Academy

Operators Precedence



Telerik Academy

Operators Precedence

- When in doubt, take a look at the MDN Precedence chart
- Parenthesis operator always has highest precedence
- It's considered a good practice to use parentheses, even when it's not necessary
  - Improves code readability

Telerik Academy

Expressions

Telerik Academy

Expressions

- Expressions are sequences of operators, literals and variables that are evaluated to some value

```
let r = (150 - 20) / 2 + 5; // r = 70

// Expression for calculation of circle area
let surface = Math.PI * r * r;

// Expression for calculation of circle perimeter
let perimeter = 2 * Math.PI * r;
```

Telerik Academy

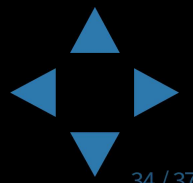
Expressions

- Expressions have:
  - Type (integer, real, boolean, ...)
  - Value

```
let a = 2 + 3; // a = 5

let b = (a + 3) * (a - 4) + (2 * a + 7) / 4;  // b = 12

let greater = (a > b) || ((a == 0) && (b == 0));
```

Telerik Academy

Expressions

Demo