

Operators in JS



By Gajanan Kharat - CODEMIND Technology

Contact us 96650 446 98
96650 445 98

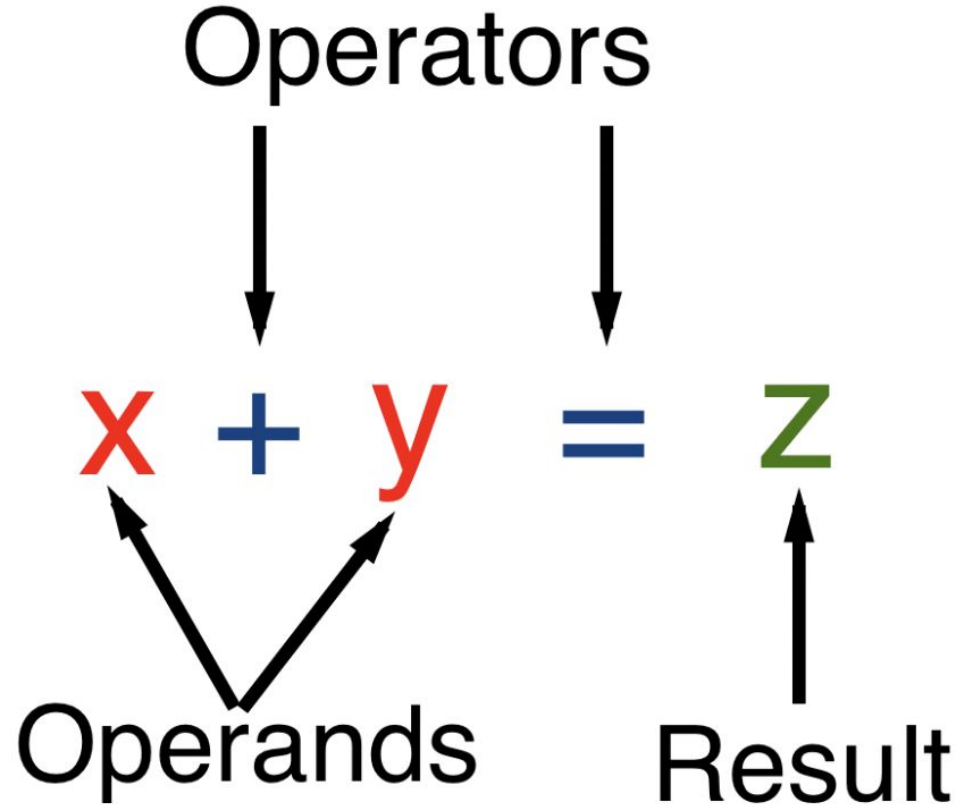
Operators

- Why Operators ?
 - What is Operands ?
 - Arithmetic Operator
 - Assignment Operator
 - Comparison Operator
 - Logical Operator
 - Special Operators
 - Typeof operator
 - Ternary or conditional operator
 - Type Conversion - Implicit and Explicit
 - Assignments
-

Operator and Operands

An operator is a special symbol used to perform operations on operands that is values and variables.

- Unary Operator
- Binary Operator
- Ternary Operator



Different Operators

Arithmetic		Assignment		Comparison		Logical	
+	Addition	=	Assignment	==	Equal	&&	AND
-	Subtraction	+=	Compound addition	===	Strict equal		OR
*	Multiplication	-=	Compound Subtraction	!=	Not Equal	!	NOT
**	Exponentiation	*=	Compound Multiplication	!==	Strict not equal		
/	Division	/=	Compound Division	>	Greater than		
%	Modulus	%=	Compound Modulus	<	Less than		
++	Increment			>=	Greater than or equal		
--	Decrement			<=	Less than or equal		

Other Operators:

- Ternary Operator or conditional operator
- typeof operator

Logical Operators

Logical Operators

- A truth table shows all possible true-false combinations of the terms
- Since `&&` and `||` each have two operands, there are four possible combinations of conditions `a` and `b`

a	b	a && b	a b
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

How to check number is even or odd ?

`givenNumber % 2 == 0` → Returns **true** then it is even number

`givenNumber % 2 == 0` → Returns **false** then it is odd number

Example:

`8%2 == 0` → Returns 0 hence 8 is even number

`23%2 == 0` → Returns false hence 23 is odd number

Arithmetic

.. + .. Add
.. - .. Subtract
.. * .. Multiply
.. / .. Divide
.. % .. Remainder
.. ** .. Exponential

Assignment

.. = .. Assign value
.. += .. Add then assign
.. -= .. Subtract then assign
.. *= .. Multiply then assign

Logical

.. || .. Or
.. && .. And

Equality

.. === .. Equality
.. == .. Equality with coercion

Conversion

+ .. Convert to number
- .. Convert to number then negate it
! .. Convert to boolean then inverse it

Relational / Comparison

.. >= .. Greater than or equal to
.. <= .. Less than or equal to
.. != .. Not equal after coercion
.. !== .. Not equal

Increment / Decrement

.. ++ Postfix increment
.. -- Postfix decrement

++.. Prefix increment
--.. Prefix decrement

Others

typeof ..
.. instanceof ..
(..
...spread-operator
.
..[..
new ..
delete ..
(.. ? .. : ..)

Operator Precedence

Given multiple operators are used in an expression, the "Operator Precedence" determines which operator will be executed first. The higher the precedence, the earlier it will get executed.

Operator Associativity

Given multiple operators have the same precedence, "Associativity" determines in which direction the code will be parsed.

See the **Operator Precedence and Associativity table** here:

<http://bit.ly/operatortable>

7 Coercion

When trying to compare different "types", the JavaScript engine attempts to convert one type into another so it can compare the two values.

Type coercion priority order:

1. String
2. Number
3. Boolean

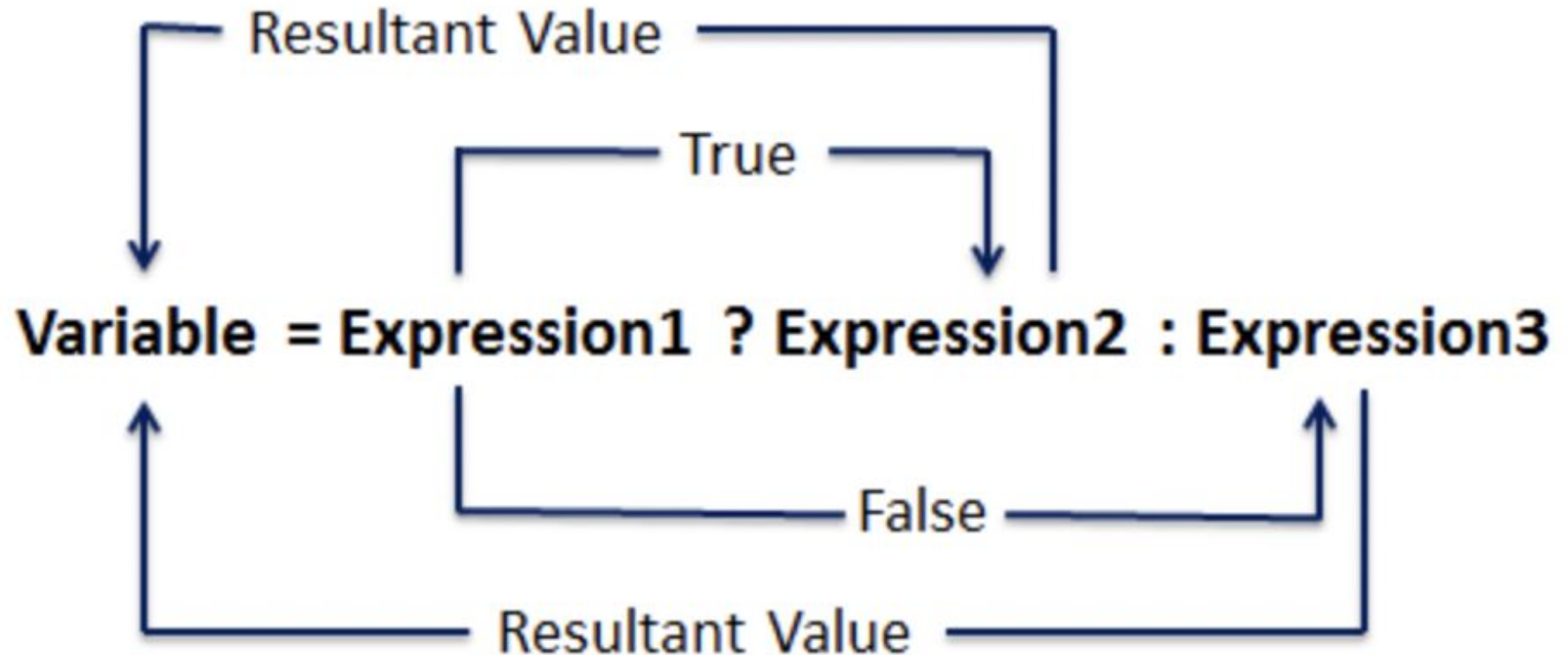
Coercion in action

Does this make sense?



```
2 + "7"; // "27"  
true - 5 // -4
```

Conditional Operator or Ternary Operator



What is difference between == and === ?

→ IMP

- Using '===' operator, it checks whether two numbers are equal by it's value and type as well.

- Note: === is most favored instead ==

- When we use '==' operator JS will compare only values and not it's data type.

Note: Internally it try to convert them to the same type and then compare them

- Example "5" == 5 JS converts the "5" to a number 2 before checking if they are equal which is why we get result as true.
- Examples:

Assignment 01: File-05_operatorA.js Don't forget to log result on console using string template only

1. Write a normal function in such a way that it should accept one string value as argument.
 - ~~1.1.~~ Function name → squareOfWordLength
 - 1.2. Find the length of word and return it's length square.
 - 1.3. Invoke or call this function for values one by one
 - 1.3.1. "JavaScript"
 - 1.3.2. "Google Chrome"
 - 1.3.3. "Developer Smart"
2. Given a string "I am Angular Developer" write a FE with no arg and no return value
 - 2.1. Find the string length and divide by total number words available in that string. Log the result on console
 - 2.2. Find the string length and multiple by the total words available in string

Assignment 0B: Make sure only use ternary operators and create fun expression for each step.

File-05_operatorB.js

1. Find the greatest number amongst two number.
 - 1.1. Variable name that can be used to store function - greaterNumber
 - 1.2. function Expr with two args and no return value
 - 1.3. Number to be checked → 10, -10 → 800, 899
2. Check → 29, 44, 0, 101 → even or odd numbers
 - 2.1. isEvenOrOddNum → Variable name that can be used to store function
 - 2.2. Fun Expression with one arg and it must return true (given number is even) or false (Given number is odd) based on number that is passed as a value
3. Check → which word has even or odd length "JavaScript", "developer", "Google"
 - 3.1. wordLength → Variable name that can be used to store function
 - 3.2. Write a function expression with one arg and return possible value "EVEN" or "ODD"

Assignment 0C: Pls use ternary operator for step 1 and step 2 , File → 05_operatorTernaryAssigC.js

Step 1. Write a normal function 'maleMarriageEligibility()' with 3 args gender, age and boyName. Function should return msg as per the step 1.2 according to condition check. Please use the **ternary operator** for conditional check

- 1.1. If gender is Male and age >=21 then
- 1.2. Hey \${boyName} you are eligible for Marriage else Not eligible for Marriage
- 1.3. Invoke the function for values:
 - 1.3.1. maleMarriageEligibility("Male", 25, "Billgates");
 - 1.3.2. maleMarriageEligibility("Male", 17, "Stew Jobs");

Step 2. Write a function "femaleMarriageEligibility()" with 3 args gender, age and girlName. Function should return msg as per the step 2.2 according to condition check. Please use the **?:** for conditional check

- 2.1 If gender is Female and age>=18 then only
- 2.2 Hey \${girlName} you are eligible for Marriage else not eligible for marriage
- 2.3 Call this function with values:
 - 2.3.1 femaleMarriageEligibility("Female", 16, "Jenifer");
 - 2.3.2 femaleMarriageEligibility("Female", 27, "Malinda Gates");

Assignment 0C: Pls use ternary operator and function expression, File → 05_operatorAssigC.js

Fun expression with no return value to check TCS interview eligibility such as, If Graduation score is greater than equal to 70% **OR** HSC score is greater than equal 80% **OR** SSC score is greater than 90% then only

- 1.1. Function expression args → gradScore, hscScore, sscScore, candidateName
- 1.2. Congrates {candidate_name} you are eligible for TCS interview. Else Unfortunately you are not eligible for interview
- 1.3. Invoke Fun Expr with values as
 - 1.3.1. 80, 86, 90, "your name"
 - 1.3.2. 70, 65, 55, "your frd name"
 - 1.3.3. 60, 79, 88, " your other frd name "

NaN - Not a Number

- NaN is a number – typeof NaN return number
- It is the result of numerical operations where result is not a number
- NaN can occur in several ways like
 - Converting an invalid string to a number
 - `var fullName = "Hello";`
 - `var myNumber = +fullName;`
 - `console.log(myNumber);`
 - Trying to do arithmetic with a non-numeric string will result in NaN (Not a Number)
 - Divide zero by zero
 - If you use NaN in a mathematical operation, the result will also be NaN

+ Operator: '+' can be used for different ways like:-

- Addition: Addition of numbers
- Concatenation: String concatenation
- Conversion: String to number conversion

Type Conversion

It is the process of converting data from one type to another type

There are two type conversion in JS

- Implicit conversion: Automatic type conversion
- Explicit conversion: Explicit type conversion

Type Conversion

It is the process of converting data from one type to another type

There are two type conversion in JS

- **Implicit conversion**: Automatic type conversion

In certain situations, JavaScript automatically converts one data type to another (to the right type). This is known as implicit conversion

When we compare using `==` operator, internally It does implicit conversion from string to number and then it compares.

- **Explicit conversion**: Explicit type conversion

You can also convert one data type to another as per your needs. The type conversion that you do manually is known as explicit type conversion.

Implicit conversion to String

```
// numeric string used with + gives string type
```

```
let result;
```

```
result = '3' + 2;
```

```
console.log(result) // "32"
```

```
result = '3' + true;
```

```
console.log(result); // "3true"
```

```
result = '3' + undefined;
```

```
console.log(result); // "3undefined"
```

```
result = '3' + null;
```

```
console.log(result); // "3null"
```

Implicit boolean conversion to Number

```
// if boolean is used, true is 1, false is 0
```

```
let result;
```

```
result = '4' - true;  
console.log(result); // 3
```

```
result = 4 + true;  
console.log(result); // 5
```

```
result = 4 + false;  
console.log(result); // 4
```

Implicit string conversion to Number

```
// numeric string used with - , / , * results number type
```

```
let result;
```

```
result = '4' - '2';
```

```
console.log(result); // 2
```

```
result = '4' - 2;
```

```
console.log(result); // 2
```

```
result = '4' * 2;
```

```
console.log(result); // 8
```

```
result = '4' / 2;
```

```
console.log(result); // 2
```

Undefined used with number, boolean or null gives NaN

```
// Arithmetic operation of undefined with number, boolean or null gives NaN
```

```
let result;
```

```
result = 4 + undefined;  
console.log(result); // NaN
```

```
result = 4 - undefined;  
console.log(result); // NaN
```

```
result = true + undefined;  
console.log(result); // NaN
```

```
result = null + undefined;  
console.log(result); // NaN
```

Explicit conversion: Convert number strings and boolean values to numbers,
In that case we can use `Number()`;

```
// string to number  
result = Number('324');  
console.log(result); // 324
```

```
result = Number('324e-1')  
console.log(result); // 32.4
```

```
// boolean to number  
result = Number(true);  
console.log(result); // 1
```

```
result = Number(false);  
console.log(result); // 0
```

Explicit conversion: Invalid string to number return NaN

If a string is an invalid number, the result will be `NaN`. For example,

```
let result;  
result = Number('hello');  
console.log(result); // NaN  
  
result = Number(undefined);  
console.log(result); // NaN  
  
result = Number(NaN);  
console.log(result); // NaN
```

Explicit conversion: string to number using + operator

```
var numberInString = "100";  
console.log(typeof numberInString)  
  
var myNumber = +numberInString;  
console.log(typeof myNumber)
```


Explicit conversion: number to string data type conversion using toString() method

```
var myNumber = 100;  
console.log(typeof myNumber); // number  
var afterConversion = myNumber.toString();  
console.log(typeof afterConversion); // string
```

Assignment: 0C

1. Check out few interesting fact and log result on console with reason:

- `0 == ' '`
- `0 == '0'`
- `0 == false`
- `null == undefined`
- `1 == [1]`
- `1 == true`
- `1 == '1'`