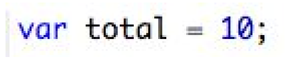
Web Development

JavaScript

Variables

JavaScript **variables** are containers for storing data values.



```
var total = 10;
```

All JavaScript **variables** must be **identified** with **unique names**.

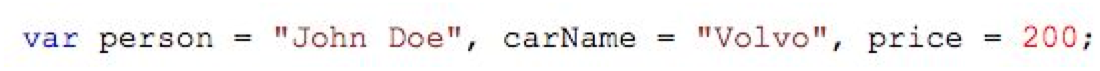
These unique names are called **identifiers**.

In JavaScript, the equal sign (=) is an "assignment" operator, not an "equal to" (==) operator.

Variables

You can declare many variables in one statement.

Start the statement with **var** and separate the variables by **comma**:

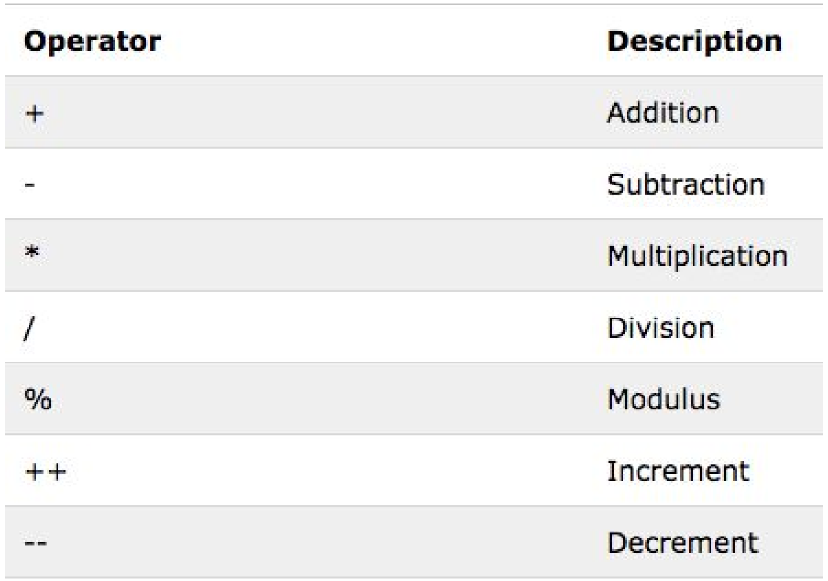


```
var person = "John Doe", carName = "Volvo", price = 200;
```

A variable declared without a value will have the value **undefined**.

If you re-declare a JavaScript variable, it will not lose its value.

Arithmetic Operators

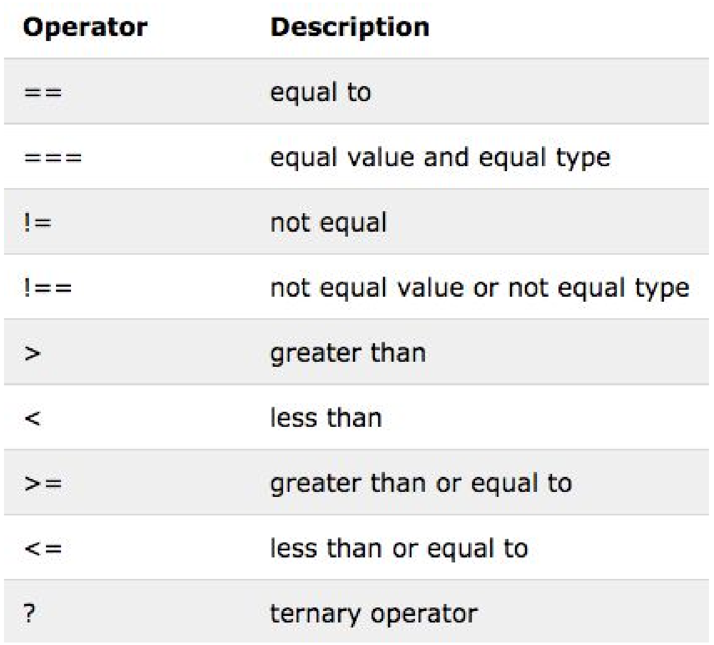


Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Assignment Operators

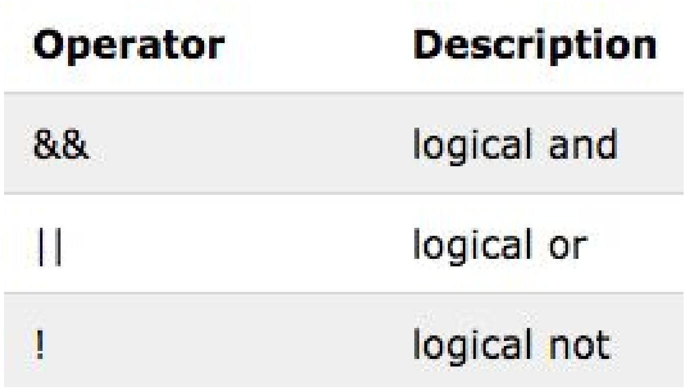
Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Comparison 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
?	ternary operator

Logical Operators

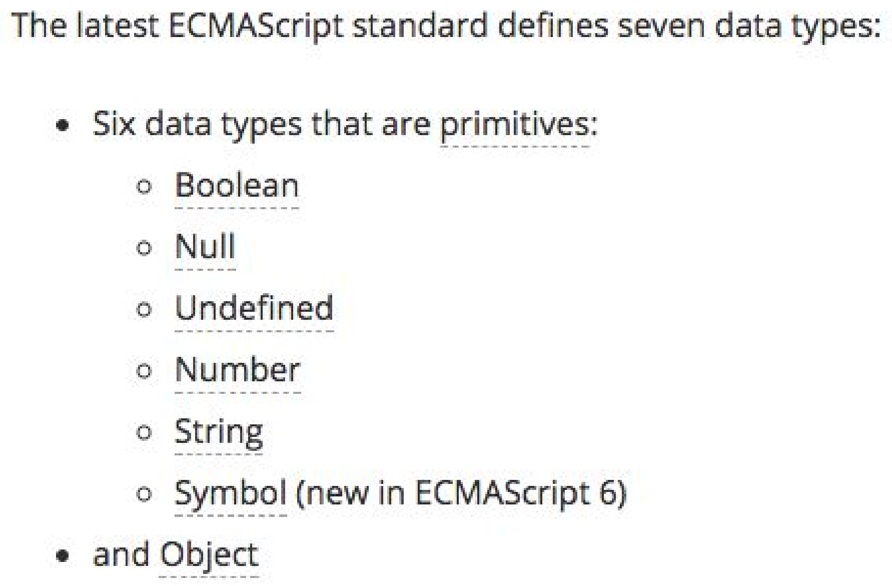


Operator	Description
&&	logical and
||	logical or
!	logical not

Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

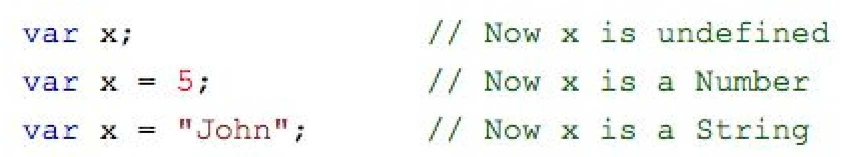


The latest ECMAScript standard defines seven data types:

- Six data types that are primitives:
  - Boolean
  - Null
  - Undefined
  - Number
  - String
  - Symbol (new in ECMAScript 6)
- and Object

Dynamic Type

JavaScript has dynamic types. This means that the same variable can be used to hold different data types.



```
var x;               // Now x is undefined
var x = 5;           // Now x is a Number
var x = "John";      // Now x is a String
```

Type Operators

typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Example

```
(function () {}) instanceof Object === true ? “YES”: “No”
```

Weird Parts of JS

Coercion

```
var x = 16 + "Volvo";
```

```
var x = 16 + 4 + "Volvo";
```

```
var x = "Volvo" + 16 + 4;
```

null > 0
null == 0
null >= 0

Function

Syntax

```
function name(parameter1, parameter2, parameter3) {
    code to be executed
}
```

The () Operator Invokes the Function

Objects

- **Name/Value pairs:** A name which maps to a unique value. The name may be defined more than once, but only can have one value in any given context. That value may be more name/value pairs.

- **Object:** A collection of name value pairs. The simplest definition when talking about JS.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

```
objectName.propertyName        objectName["propertyName"]
```

Scope

Local : Inside a function

Global : Not inside a function

Scope determines the accessibility (visibility) of variables.

Variables defined inside a function are not accessible (visible) from outside the function.

Scope Example

```
var carName = " Volvo";

// code here can use carName

function myFunction() {

    // code here can use carName

}
```

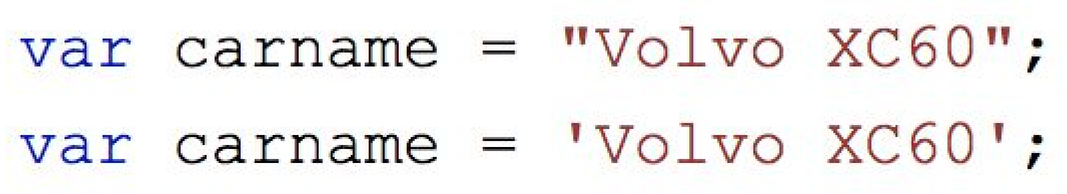
Global

```
// code here can not use carName

function myFunction() {
    var carName = "Volvo";

    // code here can use carName

}
```

Local

Strings

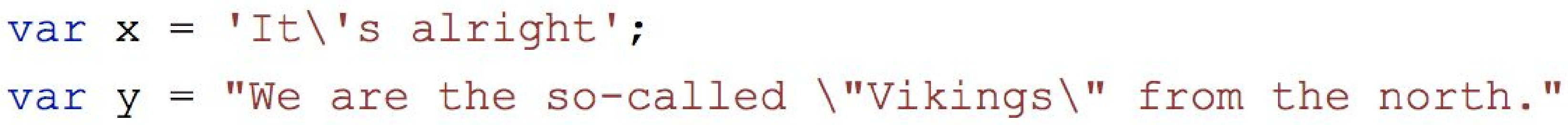
A sequence of characters inside quotes. (Either single or double quotes)



```
var carname = "Volvo XC60";
var carname = 'Volvo XC60';
```

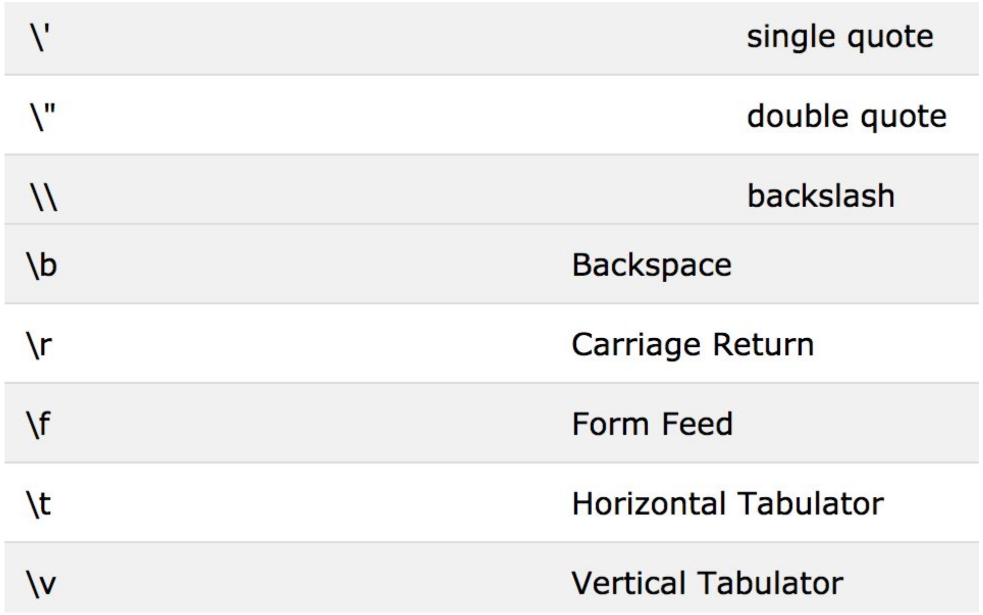
Escape Character

\ escape character turns special characters into string characters.



```
var x = 'It\'s alright';
var y = "We are the so-called \"Vikings\" from the north."
```

Special Characters



\'	single quote
\"	double quote
\\	backslash
\b	Backspace
\r	Carriage Return
\f	Form Feed
\t	Horizontal Tabulator
\v	Vertical Tabulator

String methods

- Properties: .length - returns the length of a string
- Methods:
    - .indexOf(sub[, start]) - returns the index of sub in a string otherwise -1
    - .slice(start[, end]) - returns the extracted part in a new string.
    - .substring(start[, end]) - similar to slice(), cannot accept negative indices.
    - .substr(start, length) - similar to slice(), the second param specifies the length of the extracted part.
    - .replace(old, new) - replaces a specified value with another value in a string
    - .toUpperCase() , toLowerCase()
    - .concat(str1, str2) or + operator
    - charAt(index) - returns the char at a specified index(position)
    - .split(separator) - split the string on the separator and returns an array.