JavaScript Syntax

‹ Previous Next ›

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

```
var x, y;           // How to declare variables
x = 5; y = 6;       // How to assign values
z = x + y;          // How to compute values
```

JavaScript Values

The JavaScript syntax defines two types of values: Fixed values and variable values.

Fixed values are called **literals**. Variable values are called **variables**.

JavaScript Literals

The most important rules for writing fixed values are:

Numbers are written with or without decimals:

```
10.50

1001
```

Try it Yourself »

Strings are text, written within double or single quotes:

```
"John Doe"

'John Doe'
```

Try it Yourself »

JavaScript Variables

In a programming language, **variables** are used to **store** data values.

JavaScript uses the **var** keyword to **declare** variables.

An **equal sign** is used to **assign values** to variables.

In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

```
var x;

x = 6;
```

Try it Yourself »

JavaScript Operators

JavaScript uses **arithmetic operators** (+ - * /) to **compute** values:

```
(5 + 6) * 10
```

Try it Yourself »

JavaScript uses an **assignment operator** (=) to **assign** values to variables:

```
var x, y;
x = 5;
y = 6;
```

Try it Yourself »

JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value.

The computation is called an evaluation.

For example, 5 * 10 evaluates to 50:

```
5 * 10
```

Try it Yourself »

Expressions can also contain variable values:

```
x * 10
```

Try it Yourself »

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

```
"John" + " " + "Doe"
```

Try it Yourself »

JavaScript Keywords

JavaScript **keywords** are used to identify actions to be performed.

The **var** keyword tells the browser to create variables:

```
var x, y;
x = 5 + 6;
y = x * 10;
```

Try it Yourself »

JavaScript Comments

Not all JavaScript statements are "executed".

Code after double slashes **//** or between **/*** and ***/** is treated as a **comment**.

Comments are ignored, and will not be executed:

```
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

Try it Yourself »

> You will learn more about comments in a later chapter.

JavaScript Identifiers

Identifiers are names.

In JavaScript, identifiers are used to name variables (and keywords, and functions, and labels).

The rules for legal names are much the same in most programming languages.

In JavaScript, the first character must be a letter, or an underscore (_), or a dollar sign ($).

Subsequent characters may be letters, digits, underscores, or dollar signs.

> Numbers are not allowed as the first character.
> This way JavaScript can easily distinguish identifiers from numbers.

JavaScript is Case Sensitive

All JavaScript identifiers are **case sensitive**.

The variables **lastName** and **lastname**, are two different variables.

```
var lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
```

Try it Yourself »

JavaScript does not interpret **VAR** or **Var** as the keyword **var**.

JavaScript and Camel Case

Historically, programmers have used different ways of joining multiple words into one variable name:

Hyphens:

first-name, last-name, master-card, inter-city.

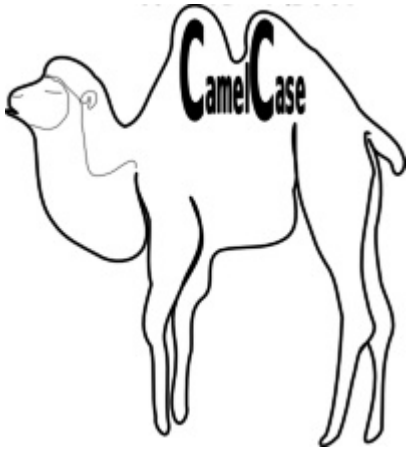
> Hyphens are not allowed in JavaScript. They are reserved for subtractions.

Underscore:

first_name, last_name, master_card, inter_city.

Upper Camel Case (Pascal Case):

FirstName, LastName, MasterCard, InterCity.



Lower Camel Case:

JavaScript programmers tend to use camel case that starts with a lowercase letter:

firstName, lastName, masterCard, interCity.

JavaScript Character Set

JavaScript uses the **Unicode** character set.

Unicode covers (almost) all the characters, punctuations, and symbols in the world.

For a closer look, please study our Complete Unicode Reference.

‹ Previous Next ›