

# Storing data in variables

A “variable” is a container, common to every scripting and programming language, in which data can be stored and retrieved later. Unlike the “strongly typed” variables in most other languages, which must declare a particular data type they may contain, JavaScript variables are much easier to use because they are “loosely typed” – so they may contain any type of data:

## Don't forget



A variable name is an alias for the value it contains – using the name in script references its stored value.

Data Type	Example	Description
boolean	<b>true</b>	A true (1) or false (0) value
number	<b>100</b> <b>3.25</b>	An integer or A floating-point number
string	<b>"M"</b> <b>"Hello World!"</b>	A single character or A string of characters, with spaces
function	<b>init</b> <b>fido.bark</b>	A user-defined function or A user-defined object method
object	<b>fido</b> <b>document</b>	A user-defined object or A built-in object

A JavaScript variable is declared using the **var** keyword followed by a space and a name of your choosing, within certain naming conventions. The variable name may comprise letters, numbers, and underscore characters, but may not contain spaces or begin with a number. Additionally you must avoid the JavaScript keywords, reserved words, and object names listed in the tables on page 9. The declaration of a variable in a script may simply create a variable to which a value can be assigned later, or may include an assignation to instantly “initialize” the variable with a value:

```
var myNumber; // Declare a variable.
myNumber = 10; // Initialize a variable.
var myString = "Hello World!"; // Declare and initialize a variable.
```

Multiple variables may be declared on a single line too:

```
var i, j, k; // Declare 3 variables.
var num=10, char="C"; // Declare and initialize 2 variables.
```

Upon initialization JavaScript automatically sets the variable type for the value assigned. Subsequent assignation of a different data type later in the script can be made to change the variable type. The current variable type can be revealed by the **typeof** keyword.

## Hot tip



Choose meaningful names for your variables to make the script easier to understand later.

## ...cont'd

- 1 Create a HTML document that embeds an external JavaScript file and has a "panel" element  

```
<script type="text/javascript" src="variable.js"> </script>  
<div id="panel"><noscript>  
  <div>! JavaScript is Not Enabled.</div> </noscript>  
</div>
```



variable.html

- 2 Open a plain text editor, like Windows Notepad, and add a function to execute after the document has loaded  

```
function init()  
{  
  
}  
window.onload=init;
```



variable.js

- 3 In the function block, declare and initialize variables of different data types  

```
var str="Text Content in JavaScript";  
var num=100;  
var bln=true;  
var fcn=init;  
var obj=document.getElementById( "panel");
```

- 4 Now insert statements to write the variable values and data types into the panel  

```
obj.innerHTML=str + " : "+typeof str;  
obj.innerHTML+="<br>" +num+ " : "+typeof num;  
obj.innerHTML+="<br>" +bln+ " : "+typeof bln;  
obj.innerHTML+="<br>init() : "+typeof fcn;  
obj.innerHTML+="<br>" +obj+ " : "+typeof obj;
```

- 5 Save the script alongside the HTML document then open the page in your browser to see the variable data



### Hot tip



The **typeof** returns a value of "undefined" for uninitialized variables.

### Hot tip



Notice how the + operator is used here to join (concatenate) parts of a string and with += to append strings onto existing strings.