**Variables**

Most of the time, a script needs to work with information. If it’s an online-shop – that’s going to be the goods and a shopping cart. If it’s a chat – users, messages and so on.

Variables are used to store the information.

**[A variable](https://javascript.info/variables" \l "a-variable)**

A [variable](https://en.wikipedia.org/wiki/Variable_(computer_science)) is a “named storage” for data. We can use variables to store goodies, visitors and other data.

To create a variable in JavaScript, we need to use the **let** keyword.

The statement below creates (in other words: *declares* or *defines*) a variable with the name “message”:

let message;

Now we can put some data into it by using the assignment operator =:

let message;

message = 'Hello'; // store the string

The **string** is now saved into the memory area associated with the variable. We can access it using the variable name:

let message;

message = 'Hello!';

alert(message); // shows the variable content

To be concise we can merge the variable declaration and assignment into a single line:

let message = 'Hello!'; // define the variable and assign the value

alert(message); // Hello!

**var instead of let**

In older scripts you may also find another keyword: var instead of let:

var message = 'Hello';

The var keyword is *almost* the same as let. It also declares a variable, but in a slightly different, “old-school” fashion.

There are subtle differences between let and var, but they do not matter for us yet.

[**A real-life analogy**](https://javascript.info/variables#a-real-life-analogy)

We can easily grasp the concept of a “variable” if we imagine it as a “box” for data, with a unique-named sticker on it.

For instance, the variable message can be imagined as a box labelled "message" with the value "Hello!" in it:



We can put any value into the box.

Also we can change it. The value can be changed as many times as needed:

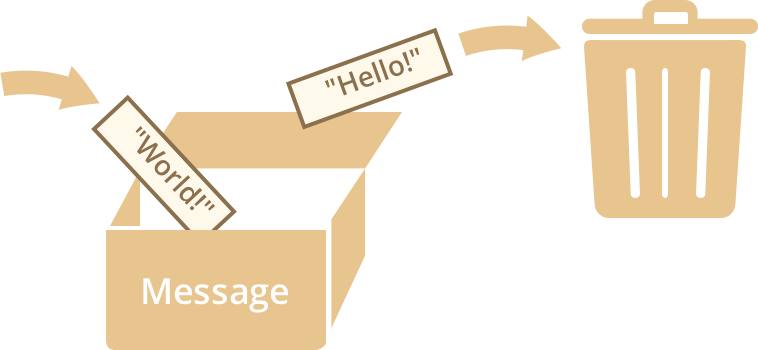
let message;

message = 'Hello!';

message = 'World!'; // value changed

alert(message);

When the value is changed, the old data is removed from the variable:



We can also declare two variables and copy data from one into the other.

let hello = 'Hello world!';

let message;

// copy 'Hello world' from hello into message

message = hello;

// now two variables hold the same data

alert(hello); // Hello world!

alert(message); // Hello world!

[**Variable naming**](https://javascript.info/variables#variable-naming)

There are two limitations for a variable name in JavaScript:

1. The name must contain only letters, digits, symbols $ and \_.
2. The first character must not be a digit.

Valid names, for instance:

let userName;

let test123;

When the name contains multiple words, [camelCase](https://en.wikipedia.org/wiki/CamelCase) is commonly used. That is: words go one after another, each word starts with a capital letter: myVeryLongName.

What’s interesting – the dollar sign '$' and the underscore '\_' also can be used in names. They are regular symbols, just like letters, without any special meaning.

These names are valid:

let $ = 1; // declared a variable with the name "$"

let \_ = 2; // and now a variable with the name "\_"

alert($ + \_); // 3

Examples of incorrect variable names:

let 1a; // cannot start with a digit

let my-name; // a hyphen '-' is not allowed in the name

**Case matters**

Variables named apple and AppLE – are two different variables.

**Reserved names**

There is a list of reserved words, which cannot be used as variable names, because they are used by the language itself.

For example, words let, class, return, function are reserved.

The code below gives a syntax error:

let let = 5; // can't name a variable "let", error!

let return = 5; // also can't name it "return", error!

**An assignment without use strict**

Normally, we need to define a variable before using it. But in the old times, it was technically possible to create a variable by a mere assignment of the value, without let. This still works now if we don’t put **use strict,** the behavior is kept for compatibility with old scripts.

// note: no "use strict" in this example

num = 5; // the variable "num" is created if didn't exist

alert(num); // 5

**That’s a bad practice, it gives an error in the strict mode:**

"use strict";//always include this statement at the start of your JS code

num = 5; // error: num is not defined

**[Constants](https://javascript.info/variables" \l "constants)**

To declare a constant (unchanging) variable, one can use const instead of let:

const myBirthday = '18.04.1982';

The variable declared using const are called “constants”. They cannot be changed. An attempt to do it would cause an error:

const myBirthday = '18.04.1982';

myBirthday = '01.01.2001'; // error, can't reassign the constant!

When a programmer is sure that the variable should never change, he can use const to guarantee it, and also to clearly show that fact to everyone.

**[Uppercase constants](https://javascript.info/variables" \l "uppercase-constants)**

There is a widespread practice to use constants as aliases for difficult-to-remember values that are known prior to execution.

Such constants are named using capitals and underscores.

Like this:

const COLOR\_RED = "#F00";

const COLOR\_GREEN = "#0F0";

const COLOR\_BLUE = "#00F";

const COLOR\_ORANGE = "#FF7F00";

// ...when we need to pick a color

let color = COLOR\_ORANGE;

alert(color); // #FF7F00

Benefits:

* COLOR\_ORANGE is much easier to remember than "#FF7F00".
* It is much easier to mistype in "#FF7F00" than in COLOR\_ORANGE.
* When reading the code – COLOR\_ORANGE is much more meaningful than #FF7F00.

When should we use capitals for a constant, and when – name them normally? Let’s make that clear.

Being a “constant” just means that the value never changes. But there are constants that are known prior to execution (like a hexadecimal value for red), and there are those that are *calculated* in run-time, during the execution, but do not change after the assignment.

For instance:

const pageLoadTime = /\* time taken by a webpage to load \*/;

The value of pageLoadTime is not known prior to the page load, so it’s named normally. But it’s still a constant, because it doesn’t change after assignment.

In other words, capital-named constants are only used as aliases for “hard-coded” values.

**[Name things right](https://javascript.info/variables" \l "name-things-right)**

Talking about variables, there’s one more extremely important thing.

Please name the variables sensibly. Take time to think if needed.

Variable naming is one of the most important and complex skills in programming. A quick glance at variable names can reveal which code is written by a beginner and which by an experienced developer.

In a real project, most of the time is spent on modifying and extending the existing code base, rather than writing something completely separate from the scratch. And when we return to the code after some time of doing something else, it’s much easier to find the information that is well-labelled. Or, in other words, when the variables have good names.

Please spend some time thinking about the right name for a variable before declaring it. That will repay you a lot.

Some good-to-follow rules are:

* Use human-readable names like userName or shoppingCart.
* Stay away from abbreviations or short names like a, b, c, unless you really know what you’re doing.
* Make the name maximally descriptive and concise. Examples of bad names are data and value. Such a name says nothing. It is only ok to use them if it’s exceptionally obvious from the context which data or value is meant.
* Agree on terms within the team and in your own mind. If a site visitor is called a “user” then we should name related variables like currentUser or newUser, but not currentVisitor or a newManInTown.

Sounds simple? Indeed it is, but creating good descriptive-and-concise names in practice is not. Go for it.

**Reuse or create?**

And the last note. There are some lazy programmers who, instead of declaring a new variable, tend to reuse the existing ones.

As the result, the variable is like a box where people throw different things without changing the sticker. What is inside it now? Who knows… We need to come closer and check.

Such a programmer saves a little bit on variable declaration, but loses ten times more on debugging the code.

**An extra variable is good, not evil.**

Modern JavaScript minifiers and browsers optimize code well enough, so it won’t create performance issues. Using different variables for different values can even help the engine to optimize.

[**Summary**](https://javascript.info/variables#summary)

We can declare variables to store data. That can be done using var or let or const.

* let – is a modern variable declaration. The code must be in strict mode to use let in Chrome (V8).
* var – is an old-school variable declaration. Normally we don’t use it at all.
* const – is like let, but the value of variable can’t be changed.

Variables should be named in a way that allows to easily understand what’s inside.

**Questions and Exercises**

1. What is a variable? How is different from a constant?

Constant is set value, it cannot be changed.

1. What is the format for creating a variable and a constant?

let, const

1. What are some rules for naming variables?

Camel case, no spaces, use \_

1. What is a string?

Content written in quotes.

1. Write a new program that does the following:

Declare two variables: admin and name. Assign the value "John" to name. Copy the value from name to admin. Show the value of admin using alert (must output “John”).

1. Create the variable with the name of our planet. How would you name such a variable?

Let planet = “”;

1. Create the variable to store the name of the current visitor. How would you name that variable?

Let visitor = “”;

1. Examine the following code:

const birthday = '18.04.1982';

const age = someCode(birthday);

Here we have a constant birthday date and the age is calculated from birthday with the help of some code (it is not provided for shortness, and because details don’t matter here).

Would it be right to use upper case for birthday? For age? Or even for both?

const BIRTHDAY = '18.04.1982'; // make uppercase?

const AGE = someCode(BIRTHDAY); // make uppercase?

If it’s the same birthday, it should be constant.

Age changes.