What is JavaScript?

JavaScript was initially created to “make web pages alive”.

The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called Java.

The “script” tag

JavaScript programs can be inserted almost anywhere into an HTML document using the `<script>` tag.

```
<!DOCTYPE HTML>
<html>
<body>
     <p>Hello Javascript</p>
 <script> alert( 'Hello, world!' ); </script>
</body>
</html>
```

Modern markup

The `<script>` tag has a few attributes that are rarely used nowadays but can still be found in old code:

The `type` attribute: `<script type=…>`

```
<script type="text/javascript"><!-- ... //-→</script>
```

External scripts

If we have a lot of JavaScript code, we can put it into a separate file.

Script files are attached to HTML with the `src` attribute:

```
<script src="/path/script.js"></script>
We can give a full URL as well. For instance:
```

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

To attach several scripts, use multiple tags:

```
<script src="/js/script1.js"></script>
<script src="/js/script2.js"></script>



…
```

Semicolons

A semicolon may be omitted in most cases when a line break exists.

This would also work:

```
alert('Hello')
alert('World')
```

```
Here, JavaScript interprets the line break as an “implicit” semicolon.
This is called an automatic semicolon insertion.
```

```
There will be error
alert("There will be an error")[1, 2].forEach(alert)
```

```
Semicolons need
alert("All fine now");
[1, 2].forEach(alert)
```

Comments

Comments can be put into any place of a script. They don’t affect its execution because the engine simply ignores them.

One-line comments start with two forward slash characters `//`.

Like this:

```
// This comment occupies a line of its own

alert('Hello');
```

Multiline comments start with a forward slash and an asterisk `/*` and end with an asterisk and a forward slash `*/`.

Like this:

```
/* An example with two messages.
This is a multiline comment.
*/
alert('Hello');
alert('World');
```

Variables

A variable is a “named storage” for data. We can use variables to store data.

1.let

2.const

3.var

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

```
var message = "Hi";
alert(message); // Hi
```

But internally var is a very different beast, that originates from very old times. It’s generally not used in modern scripts, but still lurks in the old ones.

“var” has no block scope

Variables, declared with var, are either function-scoped or global-scoped. They are visible through blocks.

For instance:

```
if (true) {
  var test = true; // use "var" instead of "let"
}

alert(test); // true, the variable lives after if
```

As var ignores code blocks, we’ve got a global variable test.

If we used let test instead of var test, then the variable would only be visible inside if:

```
if (true) {
  let test = true; // use "let"
}

alert(test); // ReferenceError: test is not defined
```

var num = 10;
Equal သင်္ကေတကို ပရိုဂရမ်းမင်းမှာ အများအားဖြင့်
တန်ဖိုးသတ်မှတ်ဖို့ သုံးကြပါတယ်။
Assignment Operator လို့ခေါ်ပါတယ်။ ညီတယ်ဆိုတဲ့ အဓိပ္ပါယ်မဟုတ်ပါဘူး။
var num1 = 3;
var num2 = 4;
var num = num1 + num2;
Variable အမည်ပေးတဲ့အခါ —

• abc စာလုံးအကြီး/အသေးတွေ ပါလို့ရတယ်၊
• 123 ဂဏန်းတွေ ပါလို့ရတယ် (ဂဏန်းနဲ့ မစရဘူး)၊
• Space တွေပါလို့ မရဘူး၊
• လိုအပ်ရင် Underscore (_)ကို ထည့်သုံးနိုင်တယ်၊
• +,-.*&#@ %^အပါအဝင် Special Character သင်္ကေတတွေ ပါလို့မရဘူး။

Examples of valid names:

```
let userName;
let test123;
```

What’s interesting – the dollar sign `'$'` and the underscore `'_'` can also 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; // hyphens '-' aren't allowed in the name
```

We can declare variables to store data by using the `var`, `let`, or `const` keywords.

•`let` – is a modern variable declaration.

•`var` – is an old-school variable declaration. Normally we don’t use it at all, but we’ll cover subtle differences from `let` in the chapter The old "var", just in case you need them.

•`const` – is like `let`, but the value of the variable can’t be changed.

Javascript Data Types

That are primitives data types

- undefined
- Boolean
- Number
- String
- BigInt
- Symbol

String

A string in JavaScript must be surrounded by quotes.

```
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed another ${str}`;
```

In JavaScript, there are 3 types of quotes.

1.Double quotes: `"Hello"`.

2.Single quotes: `'Hello'`.

3.Backticks: `` `Hello` ``.

```
let name = "mg mg"; // embed a variable
alert( `Hello, ${name}!` ); // Hello, mg mg! // embed an expression
alert( `the result is ${1 + 2}` ); // the result is 3
```

Special Data Types

JavaScript Data Type မှာ အမျိုးအစား (၃) မျိုးရှိပါသေး

တယ်။ undefined, null နဲ့ NaN တို့ဖြစ်ပါတယ်။

Type အမျိုးအစား သတ်မှတ်ထားခြင်း မရှိသေးတဲ့ အခြေအနေကို undefined လို့ခေါ်တာပါ။

```
let age;
alert(age); // shows "undefined"
```

null

ကတော့ တန်ဖိုးမရှိတဲ့ အခြေအနေကို ပြောတာပါ။

```
let age = null;
```

In JavaScript, `null` is not a “reference to a non-existing object” or a “null pointer” like in some other languages.

It’s just a special value which represents “nothing”, “empty” or “value unknown”.

The code above states that `age` is unknown.

Boolean (logical type)
The boolean type has only two values: true and false.

This type is commonly used to store yes/no values: true means “yes, correct”, and false means “no, incorrect”.
Boolean values also come as a result of comparisons:

```
let isGreater = 4 > 1;

alert( isGreater ); // true (the comparison result is "yes")
```

Boolean မှာ မူလတန်ဖိုးအနေနဲ့ true နဲ့ false နှစ်မျိုးပဲရှိပါတယ်။

```
0
""
null
undefined
```

NaN
ဒီ (၅) မျိုးကို Boolean ပြောင်းရင် false ဖြစ်ပါတယ်။ Zero, Empty String, null, undefined နဲ့
NaN တို့ပါ။ ဒီတန်ဖိုးတွေကို Falsy Value လို့ခေါ်ပါတယ်။
ကျန်တန်ဖိုးတွေ Boolean ပြောင်း
ရင် true ဖြစ်တယ်လို့ မှတ်နိုင်ပါတယ်။ Truthy Value လို့ခေါ်ပါတယ်။

Interaction: alert, prompt, confirm

As we'll be using the browser as our demo environment, let's see a couple of functions to interact with the user: `alert`, `prompt` and `confirm`.

`alert`

shows a message.

`prompt`

shows a message asking the user to input text. It returns the text or, if Cancel button or `Esc` is clicked, `null`.

`confirm`

shows a message and waits for the user to press "OK" or "Cancel". It returns `true` for OK and `false` for Cancel/`Esc`.

```
confirm("are u there?");
```

```
let age = prompt("what is your age?");
alert(`you are age is ${age}`);
```