# JavaScript

Let's learn the JavaScript language

this document was built thanks to,
https://javascript.info/

# {} - Summary

## Introduction

## {} 1.1 - Introduction/ What is JavaScript ?

JavaScript is a Programing Language that was initialiy designed to add interactivity on Web page.
Nowaday it has much more uses but we will not cover them in this course.

---

JavaScript is an interpreted language.
It means that the code is translated by an interpreter line by line in executable bytecode.
In contrast, compiled language such as C, C++ need to be compiled as an executable before running.

In JavaScript the interpreter is called the JavaScript Engine.
Every browser has an embedded engine.
( take note that some of them share the same, like Edge and Opera that use the V8 engine from Chrome )

Chrome use the V8 Engine

Firefox use the SpiderMonkey engine

---

Because of the variety of JavaScript engines, some features might not work on every browser

## {} 1.2 - Introduction/ What can in-browser JavaScript do ?

JavaScript is a high level programming language. It means that it provide you strong abstraction from your computer.
You have no low-level access on Memory or CPU like C or C++ provide.

---

JavaScript capabilities depend on the environnment it's running in. For instance
Node.js grant JavaScript support outside of the browser and allow you to access OS
API like FileSystem, Network ...
In your browser you have no such access.

In a browser environnement JavaScript is capable of,
- Dynamicaly add/edit HTML content and CSS on the Page
- React on User's inputs. Like mouse clicks, mouvements, key presses ...
- Send requests over the network ( AJAX )
- Store data localy ( see "local storage" )

## {} 1.3 - Introduction/ What can't in-browser JavaScript do ?

JavaScript abilities browser side are limited for the sake of user's safety.
We need to avoid evil Web Pages from accessing private information from the users.

---

As a consequence here is the list of restrictions that are applied,
- JavaScript browser side has no access on your disk. It cannot read/write any file within your computer.
- Tabs/Windows has no direct access between them. It is possible for a window to know an other one if it was oppened by it. But even so you will have no access on the newly oppened window.
- You can easily send requests over the network to a server but the ability to receive data is limited. It requires to explicitly add on the remote side information about the web page. ( see "CORS" )

## Data Types

# {} 2.1 - Data Types/ Object

Object are used to store keyed collection.

We can imagine an object as a cabinet with signed files. Each data is stored in a file by an unique key.

---

```
const user = new Object(); // use the "object constructor" syntax
const dog = {}; // use the "literal" syntax
```

---

We can add properties as "key:value" pair.

```
const user = {                 // the user object using the "literal" syntax
  firstName: "Dylan",          // using the key "firstName" store the value "Dylan"
  lastName: "DE SOUSA",        // using the key "lastName" store the value "DE SOUSA"
  age: 23,                     // using the key "age" store the value 23
}
```

---

We can add, remove, read keys at any time.

```
alert( user.firstName ); // display an alert with the message "Dylan"
user.isTeacher = true; // add a new key with a boolean value
delete user.age          // remove the age key from the user object
```

# {} 2.1 - Data Types/ Number

Number in JavaScript are stored in 64-bit format IEEE-754, also known as "double precision floating point numbers".

---

```
const billlion = 1000000000;
const billlion = 1_000_000_000;  // It's only "syntactic sugar" JavaScript engine will ignore the _
const billlion = 1e9;            // by appending with "e" we can specify the zeroes count.
```

When coding we try to avoid long sequence, the "e" format is from far the greatest here.

---

You can use the Hexadecimal, Binary or Octal notation,

```
const hexadecimal = 0xFF;    // hexadecimal form of 255
const binary = 0b11111111; // binary form of 255
const octal = 0o377;         // octal form of 255
```

---

You can use the toString(base) method,

```
const number = 255;
const hexadecimal = number.toString(16);  // hexadecimal form of 255
const binary = number.toString(2);        // binary form of 255
const octal = number.toString(8);         // octal form of 255
```

## {} 2.1 - Data Types/ String

String in JavaScript is used to store any textual data. Like C++ there is no separate type for single character. Strings in JavaScript are always UTF-16 encoded, it is not dependent from the page encoding.

---

```
const single = 'Hello World';
const double = "Hello World";
const backsticks = `Hello World`;
```

---

Single and double quotes are essentialy the same. Backsticks however allow us to embed expression within the string, it is called a Template String.

```
function sum(a, b) {
  return a + b;
}
alert(`1 + 2 = ${sum(1, 2)}.`); // It display an alert with the message, "1 + 2 = 3."
```

---

You can access a character using the square brackets or a method called charAt()

```
const string = 'Hello World';
alert( str[0] );                // h
alert( str.charAt(999)); // undefined
alert( str.charAt(str.length - 1) ); // Access the last character, d
```

# {} 2.1 - Data Types/ Boolean

Boolean in JavaScript is used to store true / false value.

---

```
const isTrue = true;
const isFalse = false;
```

---

Logical Operators

- || ( OR )
- && ( AND )
- ! ( NOT )
- ?? ( Nullish Coaslescing )

## || ( OR )

```
alert( true || true );   // true
alert( false || true );  // true
alert( true || false );  // true
alert( false || false ); // false
```

## && ( AND )

```
alert( true || true );   // true
alert( false || true );  // false
alert( true || false );  // false
alert( false || false ); // false
```

## ! ( NOT )

```
alert( !true );                       // false
alert( !0 );                          // true
alert( !!'non empty string' )         // true
alert( !!null );                      // false
```

## {} 2.1 - Data Types/ Array

Array in JavaScript is used to ordered collection.

---

```
const users = new Array();
const dogs = [];
```

---

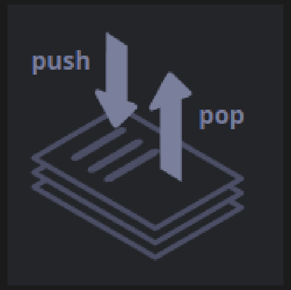
```
const users = [
  'We can mix values',
  10,
  { firstName: 'Dylan', lastName: 'DE SOUSA' },
  { firstName: 'Jhon', lastName: - },
];
```

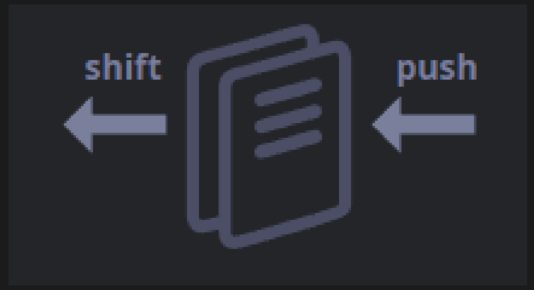
To access any data we need to use it's index ( array's index start at 0 ),

```
alert( users[2].firstName ); // Dylan
```

---

Methods pop/push, shift / unshift





## {} 2.1 - Data Types/ Null

Null in JavaScript is used to represent "nothing", "empty" or "unknown value".

```
const age = null;
```

Please take note that NULL doesn't refer to this statement:

"reference to a non-existing object"

"null pointer"

like  it does in other languages.

## {} 2.1 - Data Types/ Undefined

Undefined in JavaScript is used to represent "value is not assigned".

---

```
const age;
const name = undefined;
```

Explictly using undefined is not recommended, because we have the Null value that is used when we need to specify an empty value.
The Undefined is a default value when no value have been provided.

## Building Blocks

# {} 3.1 - Building Blocks / Conditional Branching

You may need to perform differents actions based on conditions. To do that, we can use the If statement.

```
const condition = true; // It can be a comparison using the logical operators seen before
if ( condition ) {
  alert( 'It is true.' );
}

const message = 'hello';
if ( message == 'hello' ) {
  alert( 'It is the same message' );
} else if ( message == 'world' ) { // we can several condition using the else if
  alert( "It's not 'hello' but it is 'world' " );
} else {  // this is the default case if none of the previous conditions were true
  alert( 'wrong message' );
}
```

## {} 3.2 - Building Blocks / Loops

Loops allow you to repeat the same code multiple times.

### While

```
while ( condition ) {
  // The code is executed while the condition is true
}
```

```
let i = 0;
while ( i < 3 ) { // It will loop until i reach the value 3
  alert( i ); // shows 0, then 1, then 2
}
```

A single execution of a loop is called an iteration.
Here the above while use three iterations.

## {} 3.2 - Building Blocks / Loops

### do … while

```
do {
  // loop body
} while ( condition )
```

It will first execute one iteration and then check the condition, and then It start behaving like a while.

### for

```
for ( begin; condition; step) {
  // loop body
}
```

The for loop works like this:

```
 Run Begin
   → if condition → run body → run step
   → if condition → run body → run step
   … // until it reach the condition false
```

```
const users = [];
for ( let i = 0; i < users.length; i++) {
  alert( users[i].firstName );
}
```

The continue keyword is used to skip an iteration of any loop.
The break keyword is used to stop the loop before all the iterations.

# {} 3.3 - Building Blocks / Functions

```
function sayHi() {
  alert( 'Hello World' );
}
```

Whenever we need to say hi we can now call the function sayHi()

```
sayHi() // this statement will execute the function. It will display an alert with the "Hello World" message
```

We can pass arguments to a function so we can retrieve them within the function and do what we want.

```
function sayHi( firstname ) {
  alert(`hello ${firstName}`);
}
```

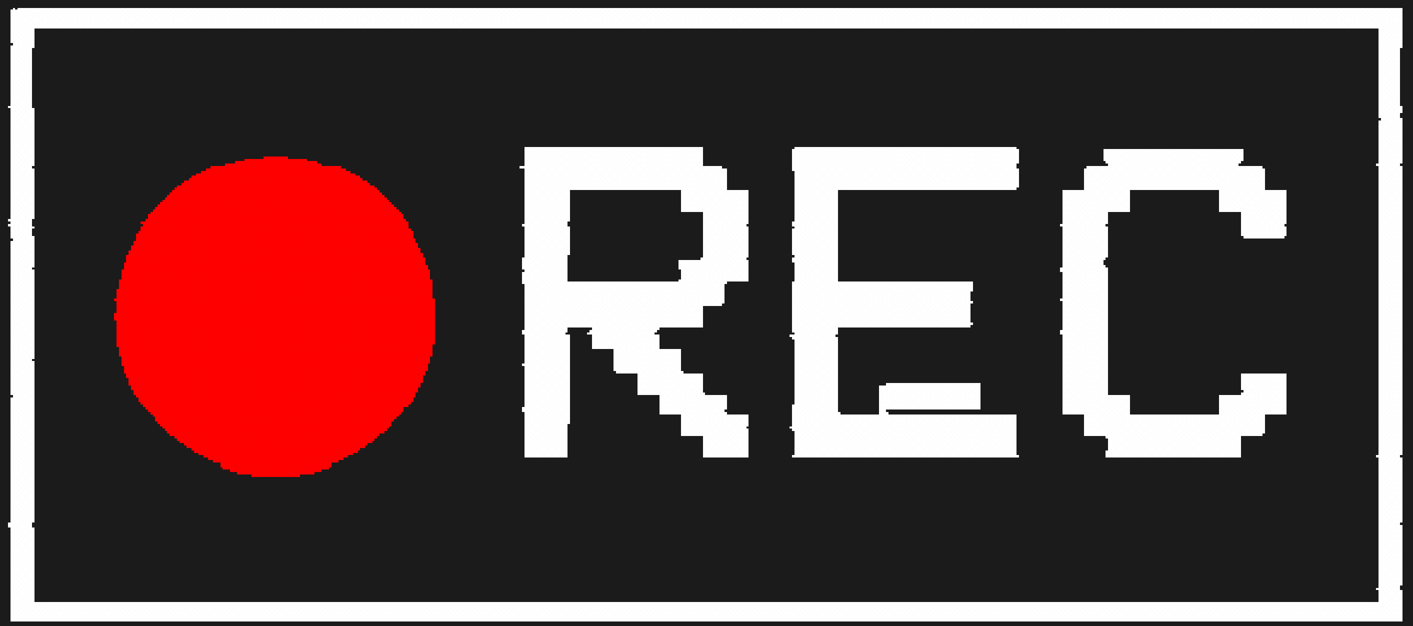
## {} 3.3 - Building Blocks / Functions

In JavaScript functions are first-class citizen. It means that we can provide a function as an argument, you may know them as callback function

```
function ask(question, yesCallback, noCallback) {
   if ( confirm(question) ) {
     yesCallback();
     return;
   }
   noCallback();
}

function showYes() {
  alert( 'Yes I am !' );
}

function showNo() {
 alert( "No I'm not !" );
}

ask( 'Are you the queen ?', showYes, showNo );
```

# Live Coding

*As always there is no stupid questions.*
*So please, feel free to ask anything.*



Every live coding session can be consulted on the repository efrei-courses/learn-nodejs