Mémo JavaScript

TODO : image

Table des matières

[Comment rechercher dans ce document 2](#_Toc150260336)

[Informations de base 3](#_Toc150260337)

[Voir les erreurs 5](#_Toc150260338)

[Console de développement 5](#_Toc150260339)

[Coder en JavaScript 6](#_Toc150260340)

[Comment lancer un code 6](#_Toc150260341)

[Structure d’un code 7](#_Toc150260342)

[Lexique 8](#_Toc150260343)

[Webographie 9](#_Toc150260344)

# Comment rechercher dans ce document

Faire des recherches par « *TODO* » afin de savoir si le document est complété.

Faire des recherches par « ATTENTION » pour connaître les pièges et autres erreurs typiques du langage.

Faire des recherches par « N.B. » pour les Nota Bene.

Faire des recherches par « B.P. » pour les bonnes pratiques.

Les « Z.B. » veulent signifier zum Beispiel qui signifie par exemple en français.

Si un mot est souligné et écrit en bleu il peut avoir un commentaire faisant référence au Lexique (voir table des matières).

# Informations de base

JavaScript a été créer pour animer les pages web.

Un programme est nommé un script.

Script non compilé, le script est effectué en texte brut.

Le Script n’a pas de définition de variable strict (var/let/rien)

JavaScript **=/=** Java.

Exécution de JavaScript dans :

* Navigateur
* Serveur
* Moteur JavaScript
  + **V8** (Chrome, Edge (Opera))
  + **SpiderMonkey**
  + Chakra
  + JavaScriptCore
  + Nitro
  + SquirrelFish

Exécution d’un moteur :

1. Analyse le script
2. Convertit le script en langage machine (compile)
3. Exécute le code machine

Fonctionnalité de JavaScript grâce à Node.js

* Lire/Écrire des fichiers
* Exécuter des requêtes réseau

Fonctionnalité de JavaScript dans un navigateur :

* Ajouter/Supprimer/Modifier du code HTML/CSS
* Réagir aux actions de l’User (clic souris, hover, etc)
* Envoyer des requêtes à des serveurs distants
* Télécharger des fichiers
* Gérer les Cookies
* Poser des questions au visiteur
* Utilisation d’un stockage local des données pour s’en rappeler
  + Remplir un formulaire => quitter/revenir
* N’a pas accès à tous les fichiers car pourrait être malveillant (possible en dehors d’une page web)

JavaScript est complètement intégré au HTML/CSS et il est pris en charge par tous les navigateurs « principaux ».

D’autres langages se « transpilent » en JavaScript pour amener plus de fonctionnalités :

* CoffeeScript
  + Syntaxe plus courte
* TypeScript
  + Typage strict
* Flow
* Dart
* Brython
  + Utilisation de Python
* Kotlin

Pour rechercher des informations ajouter « MCN » c’est la doc principale.

Attention à bien vérifier la comptabilité entre la version JavaScript et la version du navigateur.

IDE possible :

* Visual Studio Code
* Vim
* Emacs

Possibilité d’utiliser un mode strict qui est plus compatible (TODO : vérifié). Cela oblige par exemple à déclarer les variables.

Le mode stricte rend le code plus sécurisé

Pour des raisons de compatibilité si le code doit être exeécuté en mode strict :

* ‘use strict’
  + Si en haut du code
* ‘’use strict’’
  + Sinon

N.B. : Si le ‘use strict’ est utilisé dans une fonction il ne prend effet que dans cette même fonction et non ailleurs.

N.B. : Le ‘use strict’ est automatiquement utilisé quand le code est dans des classes/modules.

Pour savoir ce qui n’est plus autorisé en strict mode : « <https://www.w3schools.com/js/js_strict.aspom)> ».

# Voir les erreurs

## Console de développement

Impossibilité de voir les erreurs depuis un navigateur donc ils peuvent ajouter des « outils de développement ».

Chrome et Firefox ont les meilleurs outils de développement en 2023.

**Chrome :**

TODO : vérifier « bug.html ».

Aller dans le mode développeur (F12).

Les erreurs s’affichent dans la Console.

# Coder en JavaScript

## Comment lancer un code

**Balise HTML script :**

* <script>
* Mon code
* Alert(‘Hello World’) ;
* </script>

Si on a une balise commentaire dans le script c’est le code qui est très vieux (+15 ans) pour la rétrocomptabilité.

**Script externe :**

* <script src=’’chemin/script.js’’></script>
* <script src=’’https://pleindescript//script1.js’’></script>

On peut mettre plusieurs balises de script, mais une ne peut pas contenir du code dedans ET une source (src).

## Structure d’un code

Il peut y avoir une « infinité » d’instruction dans un script.

Pour finir une instruction on utilise un point virgule ( ;) ou un saut de page.

Une nouvelle ligne signifie SOUVENT un point-virgule mais pas forcément.

Donc faire ATTENTION à la structure du code.

Utilisation des commentaires avec des doubles slash « // » pour une ligne et « /\*\*/ » pour une ou plusieurs lignes.

ATTENTION : si un commentaire imbriqué dans un commentaire imbriqué ça crée une erreur :

* /\*commentaire**/\*erreur\*/**\*/

## Variables

Le mot clef afin de créer une variable est « *let* ».

ATTENTION : ne pas déclarer 2 fois le même nom de variable, cela engendre une erreur.

ATTENTION : la première lettre de la variable ne peut pas être un chiffre.

ATTENTION : pas de trait d’union dans une variable « - ».

ATTENTION : Les variables sont sensibles à la casse.

N.B. : une variable peut se nommer « *$* » ou encore « *\_* ».

Plusieurs variables peuvent être déclaré sur une seule ligne grâce à des virgules.

ATTENTION : ne pas FAIRE cela pour la lisibilité du code mais possibilité de se retrouver avec un code comme cela :

let user = 'John', age = 25, message = 'Hello';

Dans d’ancien script le mot clef « *var* » peut être utilisé. Il est QUASIMENT identique au mot clef « *let* ».

Une constante est défini par le mot clef « *const* »

B.P. : Avoir la constante en majuscule du début à la fin si c’est une constante « codées en dur » par exemple const NOM = ‘’LEO’’.

Certaines constante peuvent être calculé pendant l’exécution du programme, par exemple :

const pageLoadTime = /\* temps pris par une page Web pour charger \*/;

Il y a 8 types de données différentes :

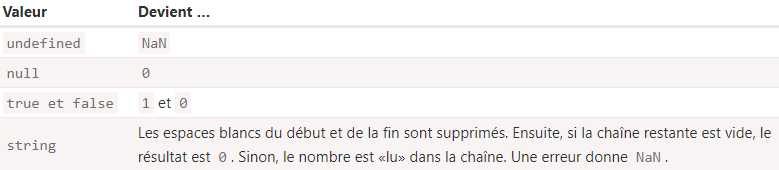
* Nombres
  + Si des calculs sont effectué cela peut rendre « Infinity », « -Infinity » et « NaN » le dernier s’affiche dans le cas d’une erreur de calcul. ATTENTION : pas de crash juste NaN
* BigInt (rarement utile)
  + Dans le cas ou le nombre est très grand, on met un n à la fin de l’initialisation
  + const bigInt=1n ;
* String
  + 3 possibilités de guillemets
  + let str=« Hi » ;
  + let str1= ‘hi’ ;
  + let str2= `hey ${str}, ${str1}` ;
    - Même chose que pour les base de données
* Booléen
* Null
  + let age = null ;
* Undefined
  + let age ;
* Object
  + Crée un objet, plusieurs valeurs possible
* Symbol
  + ID unique d’un objet
* typeof
  + Renvoie le type de ce qui suit

## Conversion Variables

Possibilité de convertir une variable avec le nom du type puis la variable entre parenthèse Z.B. :

let str = true ; //Crée un une variable string booléen.

str = String(value) ; //La variable est maintenant un string qui contient « true ».



1 – Cas de conversion spécifique

Une variable Booléenne peut être créé selon le résultat d’une comparaison

Z.B. : let isFalse = 5<10 ; //isFalse = false

## Opérateur

Il y a plusieurs opérateurs :

* +
* -
* \*
* /

Un opérateur est dit unaire s’il a 1 opérand : x = -x ; //un seul opérand

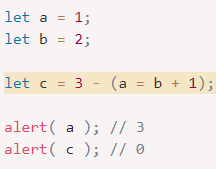
Un opérateur est dit binaire s’il y a 2 opérands : x = 3-1 ; //2 opérands

Opérations mathématique prise en charge :

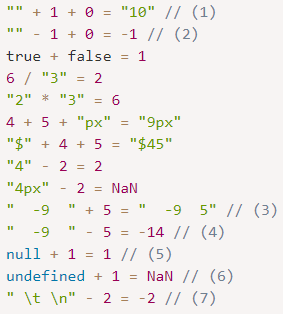
* Addition (+)
* Soustraction (-)
* Multiplication (\*)
* Division (/)
* Reste (%)
* Exponentiation (\*\*) //C’est la puissance

On peut concaténer du texte : let fullName = ‘’John’’ + ‘’Doe’’ ;

On peut assigner une variable dans l’assignation d’une autre variable :



2 – Bout de code et son rendu

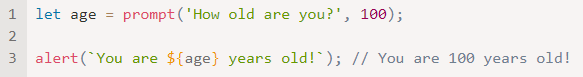


3 – Bout de code et son rendu

## Interactions

Quelques interactions :

* Alert(‘’Hello’’) ;
  + Alert renvoie un message avec un bouton ok
* prompt(title, [default]) ;
  + Fenêtre avec champ de saisie, la valeur default est égal au texte écrit par défaut.
  + N.B. : Renvoie « null » si on appuie sur Annuler.
  + Les crochets indiquent que c’est facultatif.
    - ATTENTION : Il faut quand même écrire [default] ou autre chose sinon le résultat est « undefined ».



4 –Demander l’âge puis l’afficher

* confirm
  + Fenêtre avec un oui ou un non

## Boucles

Pour sortir de plusieurs boucle on peut créer un label à une boucle afin de pouvoir effectuer Z.B. :

break label ;

# Lexique

**Transpilation**

* Convertir un code d’un langage à un autre

**Instruction**

* Commande qui effectue une action
* alert(‘’Virus détecté’’) ;

**Interaction**

* Fonction qui intéragit avec l’utilisateur

# Webographie

Définit le langage :

* [ECMA-262 - Ecma International (ecma-international.org)](https://ecma-international.org/publications-and-standards/standards/ecma-262/)

(Pas forcément à jour) :

* <https://tc39.es/ecma262/#sec-executable-code-and-execution-contexts>

Documentation (pas forcément à jour, check la date de la Version):

* <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference>

Mots Clefs :

* <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Lexical_grammar#Mots-cl%C3%A9s>

Documentation utilisé pour le mémo :

* <https://fr.javascript.info/operators>