Utiliser les documentations JavaScript

Table des matières

Introduction

Le métier de développeur nécessite une bonne capacité à apprendre par soi-même et à trouver de l'information sur Internet. En effet les technologies utilisées sont très variées, et elles évoluent rapidement. Une bonne autonomie est donc nécessaire, pour ne pas être bloqué au moindre changement d'une technologie. Heureusement Internet est une mine d'or d'informations, en témoigne le site du Mozilla Developer Network (mozilla.org)[1]. Il héberge une documentation conséquente, entre autre sur JavaScript, que nous allons dérouler dans ce module.

1. https://developer.mozilla.org

I Mozilla Developer Network (MDN)

Objectif

- Se familiariser avec le MDN.

Mise en situation

Le MDN (*Mozilla Developer Network*) est un site communautaire de documentations et tutoriels sur les technologies du Web, comme HTML, CSS, JavaScript ou encore le protocole HTTP.

Fondamental

developer.mozilla.org

Accéder à la documentation JavaScript de MDN

Méthode

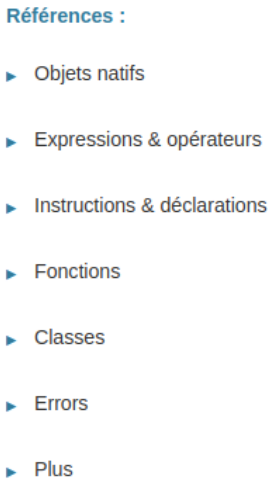
1. Depuis la page d'accueil, le menu déroulant « *Technologies* » contient un lien vers la documentation MDN du JavaScript.

   developer.mozilla.org/fr/docs/Web/JavaScript

2. Cette documentation existe en Français via un menu déroulant en haut à droite. Cependant, les pages anglaises sont les premières mises à jour et il y a un délai avant que la communauté française les traduise.

Structure du MDN

Complément



La page principale de la documentation JavaScript contient des informations sur ce langage et des liens vers des tutoriels et la référence complète. C'est la référence qui est considérée comme la documentation.

Dans la partie gauche de la page, on retrouve les grandes rubriques de cette référence.

- **Objets natifs** : contient l'ensemble des objets natifs standard (`Array`, `Date`, etc.), y compris leurs méthodes et propriétés.
- **Expressions & opérateurs** : présente les mots-clés basiques (`this`, `function`, `class`, etc.), les opérateurs arithmétiques, d'affectation et de manière générale toutes les expressions du langage.
- **Instructions et déclarations** : c'est là que se trouve les `if...else`, ainsi que les déclarations `var`, `let` et `const`.
- **Errors** : contient toutes les erreurs, exceptions et avertissements que l'interpréteur JavaScript peut retourner.

Exemples de recherches

Méthode

Si un développeur a besoin d'informations sur la méthode `map` (qui permet d'appliquer un traitement à tous les éléments d'un tableau), il a deux solutions dans le MDN.

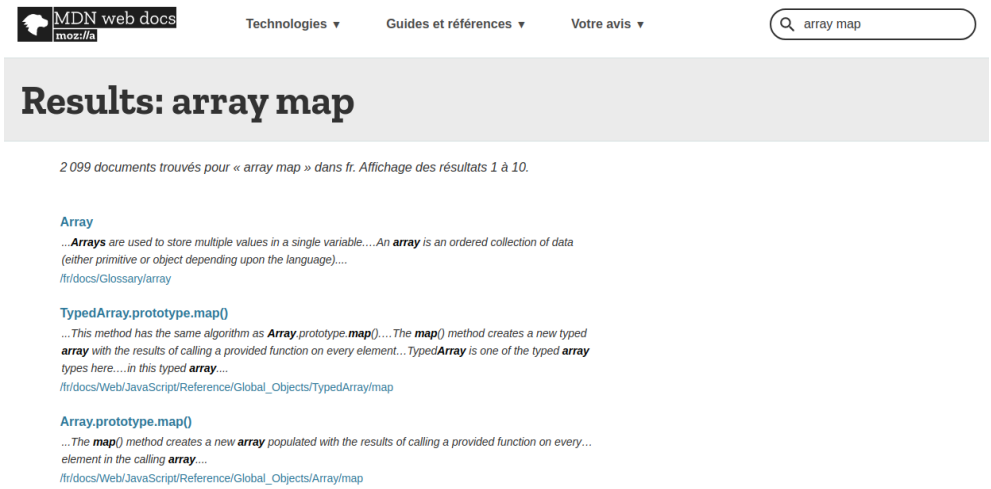
- Depuis le menu latéral :

  Il peut cliquer sur la rubrique « *Objets natifs* » puis `Array` car `map` est une méthode de la classe `Array`.

  Toujours depuis le menu latéral, il peut cliquer sur `Array.prototype.map` (et découvrir toutes les méthodes de `Array`).

- Depuis la barre de recherche :

  Il peut écrire « *array map* » dans la barre de recherche en haut à droite et trouver la page de la documentation contenant les informations qu'il cherche.

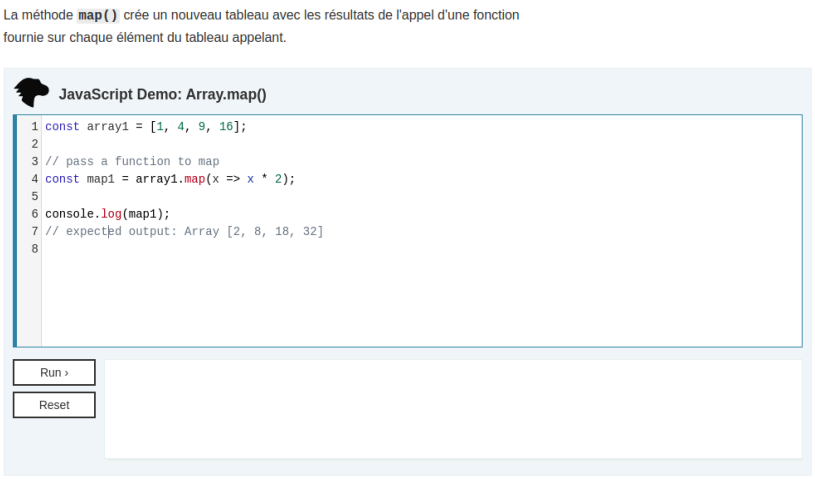


Lire une page du MDN

Méthode

En continuant sur l'exemple de la page de `map`, on y voit plusieurs parties.

Il y a tout d'abord un court descriptif de ce que fait la méthode puis un petit éditeur pour tester la méthode si c'est possible.



Ensuite, la partie *Syntaxe* contient toutes les informations importantes, par exemple :

- Les paramètres pouvant être mis en entrée de la méthode ainsi que sa valeur de retour.
- Les paramètres entre crochets ([ ]) sont optionnels.
- La fonction servant à réaliser le *mapping* doit avoir des argument spécifiques.

Syntaxe

```
var nouveauTableau = arr.map(callback [, thisArg])
```

Paramètres

`callback`

La fonction qui est utilisée pour créer un élément du nouveau tableau. Elle utilise trois arguments :

`valeurCourante`

La valeur de l'élément du tableau à traiter.

`index` Facultatif

L'index de l'élément qui est traité par la fonction.

`tableau` Facultatif

Le tableau sur lequel on a appelé la méthode `map`.

`thisArg` Facultatif

La valeur à utiliser pour `this` lors de l'exécution de `callback`. La valeur par défaut est l'objet global de l'environnement (`Window` pour un navigateur).

Valeur de retour

Un nouveau tableau composé des images de la fonction de rappel.

La suite de la page contient un descriptif plus détaillé pour comprendre comment la méthode fonctionne, de potentiels avertissements, des bonnes pratiques et des exemples. Il y aussi des informations sur l'introduction de la méthode dans la spécification, ce qui peut être utile pour les développeurs travaillant avec d'anciennes versions.

Enfin, il y a généralement un tableau de compatibilité avec les navigateurs pour les développeurs dont le code est destiné au Web.

Compatibilité des navigateurs

Update compatibility data on GitHub

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Webview Android	Chrome pour Android	Firefox pour Android	Opera pour Android	Safari sur iOS	Samsung Internet	Node.js
`map`	1	12	1.5	9	9.5	3	≤37	18	4	10.1	1	1.0	0.1.100

What are we missing?

Support complet

À la toute fin de la page, on trouve une partie *Voir aussi* qui contient généralement des méthodes/fonctions proches qui parfois sont plus adaptées à ce que veut faire le développeur.

Pour `map`, MDN propose aussi de voir `forEach` qui est dans certains cas plus adaptée que `map`.

À retenir

- MDN est un site communautaire de documentations, avec une documentation JavaScript très complète.
- Chaque page contient quasiment la même structure.
- Elle est divisée en grandes catégories pour rendre la recherche d'informations simple.

Complément

- MDN JavaScript (developer.mozilla.org)[2]

2. https://developer.mozilla.org/fr/docs/Web/JavaScript

II Exercice : Appliquer la notion

Vous faites un appel à une API pour recevoir le pseudo du meilleur joueur de votre jeu en ligne pour l'afficher dans la console. Le pseudo arrive dans la variable `bestPlayer`.

Cependant, si aucun joueur n'a joué, l'API retourne l'objet JavaScript `undefined`. Ainsi, lorsque `bestPlayer` est à `undefined`, on affiche qu'`il n'y a eu aucun joueur`.

Votre collègue a écrit ce code.

```
1 const bestPlayer = "Alice" // Un pseudo récupéré depuis l'API
2
3 if(bestPlayer === "undefined") {
4   console.log("Il n'y a eu aucun joueur")
5 } else {
6   console.log(bestPlayer)
7 }
```

Ce code ne fonctionne pas, il affiche quand même `undefined` dans la console.

Question

[solution n°1 p. 29]

Que faut-il modifier ? Voici la page MDN de undefined[3].

Indice :

Il y a un exemple de comment `undefined` est utilisé dans un `if` sur sa page MDN.

3. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/undefined

III Approfondir les bases

Objectif

- Reprendre des parties du cours avec l'aide du MDN.

Mise en situation

Un langage de programmation tel que le JavaScript est simple à prendre en main mais il n'est pas possible de tout connaître dans les moindres détails. De plus il arrive régulièrement d'avoir besoin d'utiliser, pour la première fois, des fonctionnalités très précises du langage, même après plusieurs années de pratique. Pour cela des sites comme MDN sont très intéressants et les ressources mises à disposition permettent d'approfondir ses connaissances et ainsi que de mieux comprendre le langage que l'on utilise.

Conseil

MDN permet d'approfondir certaines notions afin de découvrir de nouveaux cas d'utilisations ou pour répondre à un problème spécifique. Il est recommandé de parcourir la documentation pour compléter ses connaissances.

Modification des constantes

Exemple

Il est possible de déclarer des constantes avec le mot-clé `const`.

```
1 const pi = { value: 3.1415, description: "Pi est  le rapport constant de la
  circonférence d'un cercle à son diamètre dans un plan euclidien" }
2 pi = { value: 3.1416, description: "Pi, appelé parfois constante d'Archimède,
  est  le rapport constant de la circonférence d'un cercle à son diamètre dans un
  plan euclidien" }
3
```

Le code ci-dessus renvoie l'erreur *TypeError: Assignment to constant variable* car `pi` est protégé par `const`.

Cependant, la page Types et grammaire du MDN[4] nous apprend que les valeurs d'un objet, même déclaré avec `const` ne sont pas protégées.

```
1 const pi = { value: 3.1415, description: "Pi est  le rapport constant de la
  circonférence d'un cercle à son diamètre dans un plan euclidien" }
2 pi.value = 3.1416
3
4
```

Ainsi, ce code ne renvoie pas d'erreur et une valeur présente dans un objet déclaré avec `const` a été modifiée.

4. https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Types_et_grammaire

Paramètre par défaut des fonctions

Exemple

En JavaScript, il faut préciser quand une fonction a besoin de paramètres. Cette page du MDN[5] montre que l'on peut donner des valeurs par défaut aux paramètres de fonction.

```
/** Cette fonction ajoute un utilisateur à la base de données
  si cette fonction est appelée sans argument "nickname",
  c'est la chaîne de caractère "JohnDoe" qui est donnée par défaut
*/
function addUser (database, id, nickname = 'JohnDoe') {
  database.push([id, nickname])
  console.log('Ajout de ' + id + ' à la base de données.')
  return database
}

let db = [[1, 'Alice']] // Pour l'exemple, "db" remplace une base de données distante.
db = addUser(db, 2, 'Bob')
db = addUser(db, 3)
console.log(db)

```

Heure UTC

Exemple

Un développeur qui souhaite retenir les heures d'inscriptions à son site web va vite se retrouver face à des problèmes d'incohérences de fuseaux horaires si ses utilisateurs sont dans différentes régions du monde.

Il pourrait être intéressant de stocker, dans la base de données, uniquement l'heure UTC et le fuseau horaire de l'utilisateur.

Pour cela, un nouveau tour sur la page des dates du MDN[6] nous indique qu'il existe une méthode `getTimezoneOffset` qui retourne la différence en minutes entre l'heure locale de l'utilisateur et l'heure UTC.

```
/** Cette fonction ajoute un message à une base de données */
function logMessage (database, msg, user, time) {
  database.push([msg, user, time.getTime() + time.getTimezoneOffset() * 60, time.getTimezoneOffset()])
  console.log('Message ajouté à la base de données')
  return database
}

let db = [] // Pour l'exemple, "db" remplace une base de données distante.
db = logMessage(db, 'Un message de Alice', 'Alice', new Date())
console.log(db)

```

Dans l'exemple ci-dessus, on imagine un service de messagerie qui aurait besoin de stocker les messages d'un utilisateur à une base de données.

La fonction `database.push()` prend en argument un tableau contenant le message, l'utilisateur, l'heure d'envoi sommé à la différence de minutes à l'heure UTC pour obtenir l'heure UTC et en dernier argument le fuseau horaire.

Ici, `getTimezoneOffset()` est multiplié par 600 pour obtenir des secondes car `time.getTime()` est en millisecondes.

5. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Default_parameters
6. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Date

Les BigInt

Exemple

JavaScript, contrairement à un grand nombre de langages de programmation, possède un type unique pour les entiers et les flottants. Le MDN nous apprend sur la page Structures de données[7] que le type `Number` permet en réalité de représenter les nombres de `-(2^53 - 1)` à `2^53 - 1`.

En dehors de cet intervalle, JavaScript ne renvoie pas d'erreur mais ne représente plus les nombres de manière sure. C'est-à-dire qu'ils deviennent des approximations qui peuvent créer des bugs difficiles à corriger.

Sur cette même page, on apprend qu'il existe le type BigInt qui permet justement de représenter les entiers en dehors de l'intervalle de Number. Un BigInt se reconnaît par un « *n* » à la fin du nombre.

```
/** Cette variable contient le nombre de lignes
  d'une table de base de données.
  Ici, on imagine 1000000.
*/
let numberOfRows = 2 ** 50

/** On souhaite stocker ce nombre de ligne
  avec le bon type (number/bigint) afin
  de le manipuler ailleurs dans le code
  sans avoir de problèmes.
*/
if (numberOfRows >= Number.MAX_SAFE_INTEGER) {
  numberOfRows = BigInt(numberOfRows)
}

/** On peut désormais continuer ce code sans problème
  de type entre Number et BigInt.
*/

console.log(typeof (numberOfRows))

```

À retenir

- Il est primordial de s'instruire sur les pages du MDN pour approfondir ses connaissances et éviter les erreurs que nous avons vu.
- Le modificateur `const` ne protège pas les valeurs des objets et des tableaux.
- `BigInt` est à privilégier avec de très grands entiers.

7. https://developer.mozilla.org/fr/docs/Web/JavaScript/Structures_de_donn%C3%A9es

IV Exercice : Appliquer la notion

Vous êtes chargé de traiter les données reçues d'un capteur. Ce capteur retourne des valeur, et pour chaque nombre reçu, il faut lui appliquer un certain traitement. Le nombre se trouve dans la variable `numberReceived`.

Cependant, le capteur connaît parfois des dysfonctionnements et renvoie `NaN`, une valeur utilisée pour représenter une quantité qui n'est pas un nombre (*Not a Number*).

Le code suivant a été écrit par quelqu'un d'autre et il ne fonctionne pas. En effet, la clause `else` est systématiquement exécutée.

```
1 const numberReceived = Number.NaN
2
3 if (numberReceived === Number.NaN) {
4   console.log('Mauvais nombre, à ne pas traiter')
5 } else {
6   console.log('Nombre à traiter')
7   // Instructions pour traiter le nombre...
8 }
9
```

Question

[solution n°2 p. 29]

Sur la page MDN du type Number[8], trouvez un moyen de savoir si un nombre est `NaN`.

Corriger le bug.

Indice :

Il faut regarder du côté des méthodes.

8. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Number

V Chercher des fonctions

Objectif

- Apprendre à chercher des fonctions sur le MDN.

Mise en situation

Une bonne partie du travail d'un développeur consiste à s'approprier des fonctions, objets, méthodes pour résoudre un problème au lieu de développer lui-même des fonctionnalités, de ré-inventer la roue.

Il est utile de se renseigner sur les fonctions disponibles en rapport avec les chaînes de caractères. La page MDN des *« Strings »*[9] est faite pour ça.

Chaque méthode, fonction ou objet est cliquable et a sa propre page avec un exemple, voici quelques méthodes intéressantes :

Exemple

- `concat` : pour concaténer au moins deux chaînes ensemble.

```
1 const hello = "Bonjour"
2 const world = "tout le monde"
3
4 console.log(hello.concat(' ', world))
5 // affiche "Bonjour tout le monde"
```

Exemple

- `includes` : renvoie `true` si une chaîne contient la chaîne donnée comme argument.

```
1 const names = ['Alice', 'Jean-Marie', 'Charlie']
2
3 for (const name of names) {
4   console.log(`Le prénom ${name} ${name.includes('-') ? 'est' : "n'est pas"}
  composé`)
5 }
6 // Pour chaque prénom, affiche qu'il est composé si il inclut le caractère -
  (trait d'union).
7
```

La page de la méthode `includes` indique aussi que cette dernière est sensible à la casse.

9. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/String

Exemple

- `split` : divise une chaîne selon un séparateur et renvoie un tableau avec les différents éléments.

```
let receivedInfo = 'Luke:Skywalker:luke.sky@rebels.net:Jedi Master'
receivedInfo = receivedInfo.split(':')

const luke = {
  firstname: receivedInfo[0],
  lastname: receivedInfo[1],
  email: receivedInfo[2],
  title: receivedInfo[3]
}

console.log(luke)
```

Chaînes de caractères

Exemple

- `toUpperCase` : retourne la chaîne en majuscule, l'exemple précédent pourrait donner :

```
let receivedInfo = 'Luke:Skywalker:luke.sky@rebels.net:Jedi Master'
receivedInfo = receivedInfo.split(':')

const luke = {
  firstname: receivedInfo[0],
  lastname: receivedInfo[1].toUpperCase(),
  email: receivedInfo[2],
  title: receivedInfo[3]
}

console.log(luke)
```

À retenir

- Il est très simple de trouver toutes les fonctions/méthodes autour d'un objet JavaScript.
- Chaque page contient aussi des informations sur la manière qu'a JavaScript d'interpréter le code.

VI Exercice : Appliquer la notion

Dans un programme JavaScript, il est commun d'appeler un programme externe, par exemple pour effectuer une requête à un serveur web.

Ce serveur peut connaître des dysfonctionnements, ou ne dispose pas forcément des ressources que vous lui demandez.

Cependant, il renvoie toujours une chaîne de caractères pour vous informer de la situation.

Cette chaîne commence par 200 lorsque tout va bien, 400 lorsque le serveur n'a pas ce que vous demandez et 500 lorsqu'il connaît une panne interne.

Question

[solution n°3 p. 29]

En cherchant sur la page MDN des chaînes de caractères, trouvez deux moyens de récupérer le code situé en début de chaîne.

La chaîne est contenue dans la variable `apiResponse`.

Indice :

Les méthodes `slice` et `substring` permettent de découper la chaîne de caractères.

VII Les erreurs en JavaScript

Objectif

- Apprendre à gérer les erreurs en JavaScript.

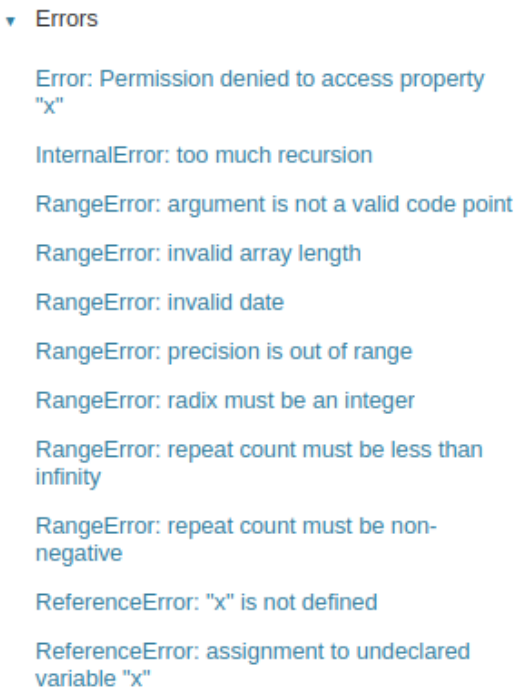
Mise en situation

Le JavaScript, à l'instar de beaucoup de langages, possède un système de gestion d'erreur.

Quand l'interpréteur traite du code problématique, il peut retourner une erreur standard à destination du développeur, qu'il est essentiel de savoir interpréter et corriger adéquatement.

Les erreurs sur le MDN

Le MDN dispose d'une liste exhaustive des erreurs natives du JavaScript.



ReferenceError: "x" is not defined Exemple

Cette erreur est fréquente et facilement compréhensible. Sa page sur le MDN[10] nous indique qu'elle apparaît lorsqu'une variable « *x* » est référencée mais n'existe pas (pas déclarée) dans la portée utilisée. On y retrouve aussi des exemples pour reproduire cette erreur avec des variables non déclarées ou hors de portée.

10. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Erreurs/Not_defined

Message

```
ReferenceError: "x" is not defined
```

Type d'erreur

`ReferenceError`.

Quel est le problème ?

Une variable qui n'existe pas est référencée quelque part. Cette variable doit être déclarée ou il faut vérifier qu'elle est disponible dans le script concerné ou dans la portée utilisée.

Note : Lors du chargement d'une bibliothèque comme jQuery, assurez-vous de bien charger la bibliothèque avant d'accéder aux variables comme `$`. La balise `<script>` utilisée pour charger la bibliothèque doit être présente avant le code qui l'utilise.

TypeError: invalid 'in' operand "x"

Exemple

```
1 const hello = 'hello world'
2 if ('hello' in hello) {
3   console.log('hello se trouve dans la chaîne de caractère ' + hello)
4 }
5
```

Le code ci-dessus pourrait paraître normal pour un développeur Python qui s'est habitué à utiliser le mot-clé `in` pour savoir si une chaîne se trouve dans une autre.

En JavaScript, ce code renvoie une erreur.

Une consultation de la page MDN de cette erreur[11] explique que `in` ne fonctionne que sur les objets (pour vérifier si une propriété existe), et non sur les chaînes de caractères.

SyntaxError: missing = in const declaration

Exemple

Une erreur de syntaxe est levée lorsqu'une constante n'est pas initialisée. En effet, les constantes doivent être initialisée à la déclaration contrairement aux variables déclarées avec `let` ou `var`, même s'il s'agit d'une bonne pratique.

Là-aussi, la page de cette erreur sur le MDN[12] donne des conseils au développeur dans la partie *Résoudre le problème*, par exemple en suggérant que ce n'est probablement pas une constante qu'il faut déclarer si cette erreur surgit. En effet, une constante est généralement connue au moment de la déclaration..

À retenir

- Il faut être capable d'analyser les erreurs en les cherchant sur le MDN pour résoudre efficacement un problème.

11. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Erreurs/in_operator_no_object
12. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Erreurs/Missing_initializer_in_const

VIII Exercice : Appliquer la notion

En fonction de votre projet, vous utilisez différents *debuggers* (outil pour exécuter le code ligne par ligne pour dénicher de mauvais effets et ainsi trouver des bugs).

Au début d'un programme, vous voulez savoir quel debugger est disponible sur votre machine.

Pour ce faire, vous disposez d'une fonction `myDebugger()` qui retourne renvoie le debugger actif.

```
/** Pour l'exercice, on retourne "gdb" mais ça
  pourrait être n'importe quoi.
*/
function myDebugger () {
  return 'gdb'
}

const DEBUG_MODE = true

if (DEBUG_MODE) {
  const debugger = myDebugger()
  // Instructions suivantes...
}

```

Mais ce code lève une erreur. Sur Firefox :

```
SyntaxError: missing variable name
```

Et sur Chromium :

```
SyntaxError: Unexpected token 'debugger'
```

Question

[solution n°4 p. 29]

Corriger l'erreur en vous aidant du MDN.

Indice :

On pourra par exemple chercher le mot-clé `debugger` dans le MDN, ou chercher l'erreur elle-même.

IX DevDocs.io

Objectif

- Découvrir DevDocs.io.

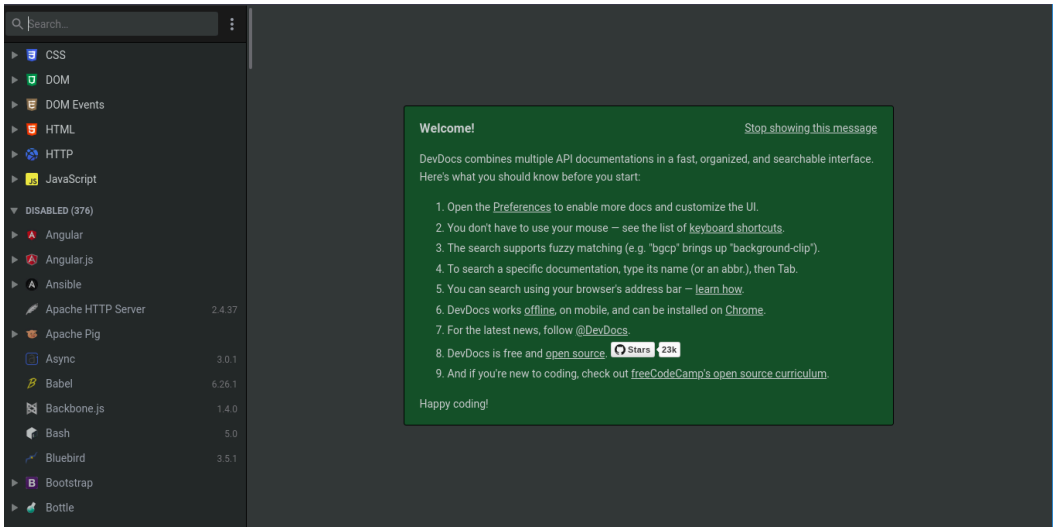
Mise en situation

Ce qui est intéressant avec les langages de programmation répandus, c'est la forte activité de la communauté qui permet de le faire évoluer et de le documenter. MDN n'est pas le seul site à proposer des ressources intéressantes pour les développeurs, il en existe d'autres, certains maintenus par la communauté. Selon les besoins, il peut s'agir de documentation, de forums pour demander de l'aide, ou de listes de discussions pour faire remonter des anomalies dans le langage.

DevDocs.io

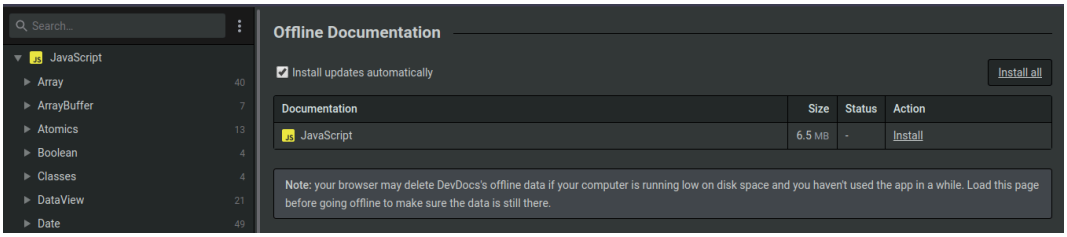
DevDocs[13] est un site web qui, à l'instar du MDN, propose une documentation des technologies du Web (HTML/CSS/HTTP/JS) mais aussi de librairies et frameworks (jQuery, Angular, etc.).

Dans les préférences (à droite de la barre de recherche), il est possible de ne garder que la documentation JavaScript.



Toujours dans les paramètres, le bouton *Offline Data* mène à une page qui propose de télécharger la documentation JavaScript en cliquant sur le bouton *Install* à droite de la page.

Une fois le téléchargement terminé, le site est accessible même sans connexion à Internet.



13. https://devdocs.io/

X Exercice : Appliquer la notion

Question

[solution n°5 p. 30]

Donner les pages relatives aux méthodes suivantes sur le MDN (mozilla.org) et DevDocs.io :

- Retirer le dernier élément d'un tableau ;
- Générer un nombre aléatoire ;
- Concaténer une chaîne de caractères avec elle-même un certain nombre de fois.

XI Quiz

Exercice 1 : Quiz - Culture

[solution n°6 p. 30]

Exercice

Les réponses à ces questions se trouvent sur cette page du MDN (mozilla.org)[14].

Quel est le standard de JavaScript ?

- **A** Le même que le Java
- **B** Les PEP
- **C** ECMAScript
- **D** Aucune de ces réponses

Exercice

Pourquoi le JavaScript a t-il été conçu ?

- **A** Pour exécuter des scripts sur les pages web
- **B** Pour faire tourner des serveurs (via Node.js)
- **C** Pour développer des applications de bureau (via Electron)
- **D** Pour prendre le relais du Java

Exercice

Lequel de ces éléments n'est pas un objet standard du JavaScript ?

- **A** Number
- **B** Error
- **C** Array
- **D** Integer

14. https://developer.mozilla.org/fr/docs/Web/JavaScript

Exercice 5 : Quiz - Méthode

[solution n°7 p. 31]

Pour répondre à ces questions, il faut chercher dans le MDN (mozilla.org)[15].

Exercice

Comment savoir si au moins un élément d'un tableau `myArray`, de type `Array`, remplit une certaine condition ?

- **A** `myArray.condition()`
- **B** `myArray.some()`
- **C** `myArray.verify()`
- **D** Il est nécessaire d'écrire une telle fonction soi-même

Exercice

Est-il possible d'utiliser des fonctions fléchées (Arrow Functions) dans la version 20 de Firefox ?

- **A** Oui
- **B** Non

Exercice

JavaScript ne fait pas la différence, en termes de types, entre nombre entier et nombre flottant.

Quelle fonction permet de vérifier si une variable `age` de type `Number` représente un nombre entier ?

- **A** Number.isInteger(age)
- **B** age.isInteger
- **C** age.isInteger()
- **D** isInteger(age)

Exercice 9 : Quiz - Code

[solution n°8 p. 32]

Le but ici est de chercher sur le MDN (mozilla.org)[16] les fonctions manquantes (___) des questions.

15. https://developer.mozilla.org/fr/docs/Web/JavaScript
16. https://developer.mozilla.org/fr/docs/Web/JavaScript

Exercice

La fonction suivante remplace `.com` par `.fr` dans la chaîne `domain`

```
function goToFrench(domain) {
  return domain.___(/.com/, ".fr")
}

console.log(goToFrench("mozilla.com"))
// outputs: mozilla.fr
```

- **A** `change`
- **B** `match`
- **C** `endsWith`
- **D** `replace`

Exercice

`db` est une variable qui permet d'interagir avec une base de données. Le développeur veut stocker les noms de familles en **majuscules**.

```
function storeUserToDatabase(email, firstname, lastname) {
  db.addUser(email, firstname, lastname.___()
}
```

- **A** `toUpper`
- **B** `toUpperCase`
- **C** `goBig`
- **D** C'est une fonction à écrire soi-même.

Exercice

La fonction ci-dessous cherche à surveiller le trafic sur un site web.

Si le nombre de connections dépasse un certain nombre, le type `Number` ne pourra pas représenter adéquatement ce nombre ; il faudra utiliser une variable de type `BigInt`.

```
function monitorTraffic() {
  let numberOfConnections = getNumberOfConnections()

  if(numberOfConnections >= Number.___)
    numberOfConnections = BigInt(numberOfConnections)
  }

  ...
}
```

- **A** `MAX_SAFE_INTEGER`

B `MAX_INTEGER`

C `MAX_SAFE`

D `MAX_VALUE`

XII Exercice : Défi

Le but de ce défi est de construire un programme JavaScript pas à pas en s'aidant de la documentation MDN[17].

Ce programme a pour objectif de créer une petite base de données de films.

Les films sont stockés dans un ensemble (`Set`) qui est une structure de données qui ressemble aux tableaux.

Question 1

[solution n°9 p. 33]

Trouvez la documentation des ensembles sur MDN.

Indice :

Les `Set` sont des objets natifs.

Question 2

[solution n°10 p. 33]

Créez un programme JavaScript et déclarez l'ensemble vide nommé `movies`.

Question 3

[solution n°11 p. 34]

Trouvez la documentation de la fonction permettant d'ajouter des éléments dans un ensemble.

Question 4

[solution n°12 p. 34]

Ajoutez à l'ensemble les films suivants :

- Matrix
- The Artist

Question 5

[solution n°13 p. 34]

Trouvez la documentation de la boucle de type `for..of`.

Question 6

[solution n°14 p. 34]

Utilisez la structure `for..of` pour afficher tous les films de l'ensemble `movies`.

On fera précéder chaque titre d'un tiret.

Question 7

[solution n°15 p. 34]

Trouvez la documentation de la fonction permettant de supprimer un élément d'un ensemble.

Que renvoie cette fonction ?

17. https://developer.mozilla.org/fr/docs/Web/JavaScript

Question 8

[solution n°16 p. 34]

Supprimer le film The Artist ; vous testerez si la suppression a réussi ou non.

Question 9

[solution n°17 p. 35]

Ajoutez à votre programme un menu permettant de :

- ajouter des films
- supprimer des films
- sortir du menu

Conclusion

Savoir lire, mais aussi écrire, de la documentation est une compétence importante du développeur. Sans que cela ne soit très compliqué, il convient tout de même de savoir chercher l'information au bon endroit. Dans ce module nous avons beaucoup parlé de MDN, qui est une référence en terme de documentation pour les technologies du Web, mais il en existe d'autres.

Le plus important reste de ne pas réinventer la roue à chaque nouveau programme, mais de comprendre tout de même ce que l'on peut trouver dans la documentation, et ne pas se contenter de simplement copier du code.

Solutions des exercices

Solution n°1

[exercice p. 9]

Il suffit de retirer les guillemets (") autour de "undefined" comme le montre l'exemple sur la page du MDN.

```
const bestPlayer = "Alice" // Un pseudo récupéré depuis l'API

if(bestPlayer === undefined) {
  console.log("Il n'y a eu aucun joueur")
} else {
  console.log(bestPlayer)
}
```

Solution n°2

[exercice p. 13]

Pour savoir si un nombre est considéré comme *Not a Number*, il faut utiliser la méthode `Number.isNaN()`.

```
if (Number.isNaN(numberReceived)) {
  console.log('Mauvais nombre, à ne pas traiter')
} else {
  console.log('Nombre à traiter')
  // Instructions pour traiter le nombre...
}
```

Solution n°3

[exercice p. 16]

```
// Méthode 1
const status = apiResponse.slice(0, 3)

// Méthode 2
const status2 = apiResponse.substring(0, 3)

```

Pour cet exemple, ces deux fonctions font la même chose. Leurs différences sont explicitées sur leurs pages MDN respectives.

Solution n°4

[exercice p. 19]

Le mot `debugger` est un mot-clé réservé par l'interpréteur JavaScript, il ne peut donc pas être utilisé comme nom de variable ou fonction. On pourra corriger le code comme suit :

```
const debuggerName = myDebugger()
```

Solution n°5

[exercice p. 21]

- Retirer le dernier élément d'un tableau :
  MDN[18] / DevDocs[19]
- Générer un nombre aléatoire :
  MDN[20] / DevDocs[21]
- Concaténer une chaîne de caractères avec elle-même un certain nombre de fois :
  MDN[22] / DevDocs[23]

Solution n°6

[exercice p. 22]

Exercice

Les réponses à ces questions se trouvent sur cette page du MDN (mozilla.org)[24].
Quel est le standard de JavaScript ?

A Le même que le Java

B Les PEP

C ECMAScript

D Aucune de ces réponses

Exercice

Pourquoi le JavaScript a t-il été conçu ?

A Pour exécuter des scripts sur les pages web

B Pour faire tourner des serveurs (via Node.js)

C Pour développer des applications de bureau (via Electron)

D Pour prendre le relais du Java

Node.js et Electron sont des évolutions de l'écosystème JavaScript.

18. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop
19. https://devdocs.io/javascript/global_objects/array/pop
20. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random
21. https://devdocs.io/javascript/global_objects/math/random
22. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/String/repeat
23. https://devdocs.io/javascript/global_objects/string/repeat
24. https://developer.mozilla.org/fr/docs/Web/JavaScript

Exercice

Lequel de ces éléments n'est pas un objet standard du JavaScript ?

A Number

B Error

C Array

D Integer

Le typage du JavaScript englobe les entiers dans le type `Number`.

Solution n°7

[exercice p. 23]

Exercice

Comment savoir si au moins un élément d'un tableau `myArray`, de type `Array`, remplit une certaine condition ?

A `myArray.condition()`

B `myArray.some()`

C `myArray.verify()`

D Il est nécessaire d'écrire une telle fonction soi-même

La page (mozilla.org)[25] dédiée aux variables `Array` nous indique que `some()` permet de vérifier automatiquement si un élément d'un tableau remplit une condition.

Exercice

Est-il possible d'utiliser des fonctions fléchées (Arrow Functions) dans la version 20 de Firefox ?

A Oui

B Non

25. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array

La page (mozilla.org) des fonctions fléchées[26] indique que le support arrive à la version 22.

Exercice

JavaScript ne fait pas la différence, en termes de types, entre nombre entier et nombre flottant.

Quelle fonction permet de vérifier si une variable `age` de type `Number` représente un nombre entier ?

- **A** Number.isInteger(age)
- **B** age.isInteger
- **C** age.isInteger()
- **D** isInteger(age)

Solution n°8

[exercice p. 23]

Exercice

La fonction suivante remplace `.com` par `.fr` dans la chaîne `domain`

```
function goToFrench(domain) {
  return domain.___(/.com/, ".fr")
}

console.log(goToFrench("mozilla.com"))
// outputs: mozilla.fr
```

- **A** `change`
- **B** `match`
- **C** `endsWith`
- **D** `replace`

La méthode `replace()` permet de remplacer un morceau de chaîne de caractères par un autre.

Exercice

db est une variable qui permet d'interagir avec une base de données. Le développeur veut stocker les noms de familles en **majuscules**.

26. https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Fonctions/Fonctions_fl%C3%A9ch%C3%A9es

```
function storeUserToDatabase(email, firstname, lastname) {
  db.addUser(email, firstname, lastname.___()
}
```

A `toUpper`

B `toUpperCase`

C `goBig`

D C'est une fonction à écrire soi-même.

Exercice

La fonction ci-dessous cherche à surveiller le trafic sur un site web.

Si le nombre de connections dépasse un certain nombre, le type `Number` ne pourra pas représenter adéquatement ce nombre ; il faudra utiliser une variable de type `BigInt`.

```
function monitorTraffic() {
  let numberOfConnections = getNumberOfConnections()

  if(numberOfConnections >= Number.___)
    numberOfConnections = BigInt(numberOfConnections)
  }

  ...
}
```

A `MAX_SAFE_INTEGER`

B `MAX_INTEGER`

C `MAX_SAFE`

D `MAX_VALUE`

`Number.MAX_SAFE_INTEGER` contient le nombre le plus élevé que `Number` peut contenir sans risque de perte d'information.

Solution n°9

[exercice p. 26]

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Set

Solution n°10

[exercice p. 26]

```
// Empty set
let movies = new Set()
```

Solution n°11

[exercice p. 26]

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Set/add

Solution n°12

[exercice p. 26]

```
// Empty set
let movies = new Set()

// Add two movies
movies.add('Matrix')
movies.add('The Artist')
```

Solution n°13

[exercice p. 26]

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Statements/for...of

Solution n°14

[exercice p. 26]

```
// Empty set
let movies = new Set()

// Add two movies
movies.add('Matrix')
movies.add('The Artist')

// Display the movies
for (let movie of movies) {
  console.log('-', movie)
}
```

Solution n°15

[exercice p. 26]

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Set/delete

La fonction renvoie : *« true si un élément de l'objet Set a été retiré lors de l'opération, false sinon. »*

Solution n°16

[exercice p. 27]

```
// Empty set
let movies = new Set()

// Add two movies
movies.add('Matrix')
movies.add('The Artist')

// Display the movies
for (let movie of movies) {
  console.log('-', movie)
}

// Delete The Artist
if (movies.delete('The Artist')) {
  console.log('Suppression de The Artist effectuée.')
```

```
}
else {
  console.log('Suppression de The Artist impossible, le film n\'existe pas.')
}
```

Solution n°17

[exercice p. 27]

```
// Empty set
let movies = new Set()

// Add two movies
movies.add('Matrix')
movies.add('The Artist')

// Display the movies
for (let movie of movies) {
  console.log('-', movie)
}

// Delete The Artist
if (movies.delete('The Artist')) {
  console.log('Suppression de The Artist effectuée.')
}
else {
  console.log('Suppression de The Artist impossible, le film n\'existe pas.')
}

let choice
while (choice !== 0) {
  // Movie list
  console.log()
  console.log('Liste des films :')
  for (let movie of movies) {
    console.log('-', movie)
  }
  // Menu
  console.log()
  console.log('Menu :')
  console.log('1. Ajouter un film')
  console.log('2. Supprimer un film')
  console.log('0. Quitter')
  choice = Number(prompt('Choix '))

  // Add or delete
  let movie
  if (choice === 1 ) {
    movie = prompt('Titre du film à ajouter ')
    movies.add(movie)
    console.log(movie, 'ajouté.')
  }
  else if (choice === 2) {
    movie = prompt('Titre du film à supprimer ')
    if (movies.delete(movie)) {
      console.log('Suppression de', movie, 'effectuée.')
    }
    else {
      console.log('Suppression de', movie, 'impossible, le film n\'existe pas.')
    }
  }
}
```

Crédits des ressources

p. 4, 5, 5, 6, 20, 20
Licence : Domaine Public