Introduction à JavaScript

Table des matières

Introduction

Le JavaScript est un langage de programmation accessible et très utilisé dans un grand nombre d'applications, voire omniprésent pour les sites web.

Ce module va permettre, en découvrant l'environnement de développement de Repl.it, d'introduire les différents concepts du langage, sa syntaxe et les bonnes pratiques de code pour débuter et coder ses premiers scripts.

I Exécuter des programmes JavaScript avec un navigateur web

Objectif

- Découvrir les fondements de JavaScript.

Mise en situation

JavaScript est un langage de **programmation de script** largement utilisé pour animer les sites web.

- Un langage de script est un langage **interprété**.
- Sa syntaxe est dite de **haut niveau**, plus proche du langage naturel que du langage machine : par conséquent, il est relativement simple à apprendre et est indépendant de l'aspect matériel de la machine sur lequel il est lancé.

Origine

À l'origine de JavaScript se trouve **ECMAScript**. ECMAScript est un **standard** constitué d'un ensemble de règles mises en pratique dans plusieurs langages de script. À ses débuts, le JavaScript est utilisé dans les **navigateurs web** : les navigateurs étant des **clients web** (car ils demandent des services à des serveurs web), JavaScript devient populaire pour le développement web dit orienté client (associé au HTML et CSS pour la réalisation de pages web).

Langage interprété

Az Définition

Un langage interprété est un langage qui est exécuté par un autre programme : l'**interpréteur**.

Celui-ci lit une ligne de code, l'analyse, l'exécute si elle est correcte et passe à la suivante ; il répète ce processus pour chaque instruction du programme. Un programme en langage interprété est donc exécuté au fur et à mesure de sa lecture.

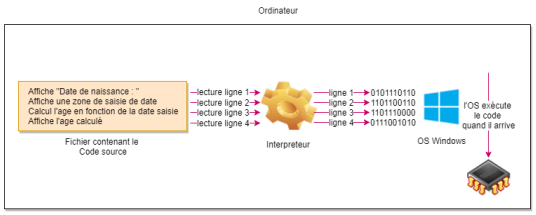


Schéma d'un langage interprété

Interpréteur vs compilateur

⊕ Complément

À la différence d'un langage interprété, un langage **compilé** est **traduit une seule fois** du code source vers le langage machine, et c'est ce code machine qui sera ensuite exécuté. L'étape de traduction n'est pas répétée à chaque exécution ce qui est en général plus performant.

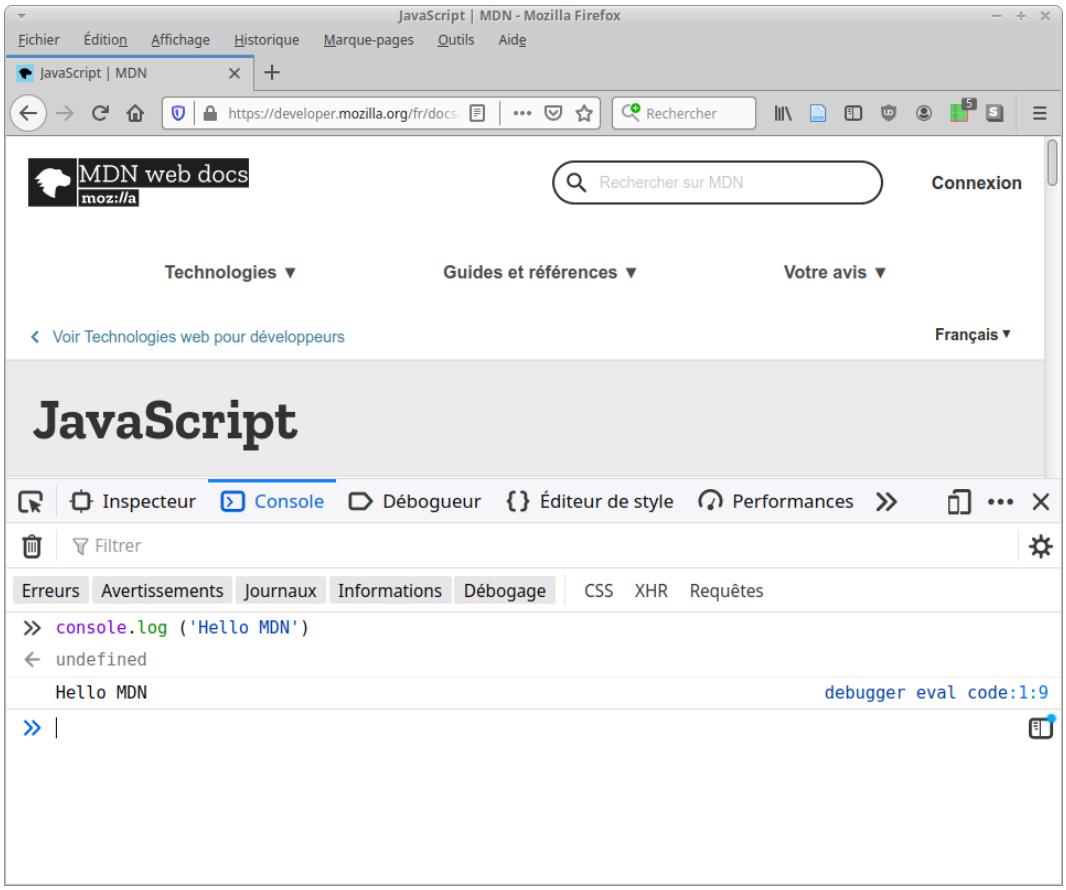
Exécuter du JavaScript dans un navigateur

Méthode

On peut exécuter du JavaScript dans la **console web** du navigateur :

- sur **Chrome** : Clic droit > Inspecter ou `Ctrl+Shift+i`, puis onglet `Console`,
- sur **Firefox** : Clic droit > Examiner l'élément ou `Ctlr+Shift+k`, puis onglet `Console`.

On peut alors exécuter du code dans le champ de la console.



Repl.it

Complément

Il existe des interpréteur JavaScript en ligne, comme Repl.it[1].

Node.js

Complément

Node.js est un environnement logiciel permettant de créer des applications en exécutant du JavaScript du **côté serveur**.

Depuis l'apparition de Node.js, JavaScript est devenu multi-usages et est désormais utilisé comme langage côté serveur, comparable aux langages traditionnels comme le PHP, le Ruby, le C, etc.

À retenir

JavaScript est un langage interprété polyvalent d'abord popularisé par le web et devenu multi-usage depuis Node.js.

1. https://repl.it

II Exercice

On dispose du code suivant, qui calcule le volume d'une sphère de rayon 12 :

```
4 * Math.PI * Math.pow(12, 3) / 3

```

Question 1

[solution n°1 p. 33]

Copier le code dans la console JavaScript d'un navigateur web et l'exécuter. Quelle est la valeur calculée ?

Question 2

[solution n°2 p. 34]

Exécuter à présent une seconde version du code qui utilise la fonction `Math.trunc`. Que fait `Math.trunc` ?

```
Math.trunc(4 * Math.PI * Math.pow(12, 3)) / 3

```

III Exécuter des programmes JavaScript avec Repl.it

Objectifs

- Créer son environnement de travail sur Repl.it ;
- Créer son premier programme JavaScript.

Mise en situation

Lorsque l'on commence à développer, il est nécessaire de mettre en place ce que l'on appelle un environnement de développement. Pour commencer, cela signifie installer les dépendances nécessaires à l'exécution du langage sur la machine. Ensuite, il est préférable d'installer et d'utiliser un IDE, c'est à dire un logiciel permettant d'écrire notre code, généralement adapté au langage utilisé. Celui-ci peut par exemple proposer des fonctionnalités de détection et corrections de fautes dans le code.

Pour faciliter cette mise en place, le site **Repl.it** permet d'avoir un environnement de développement complet en ligne directement dans son navigateur, pour une large variété de langages.

Rappel

Repl.it (cf. p.43)

Créer un programme JavaScript sur Repl.it

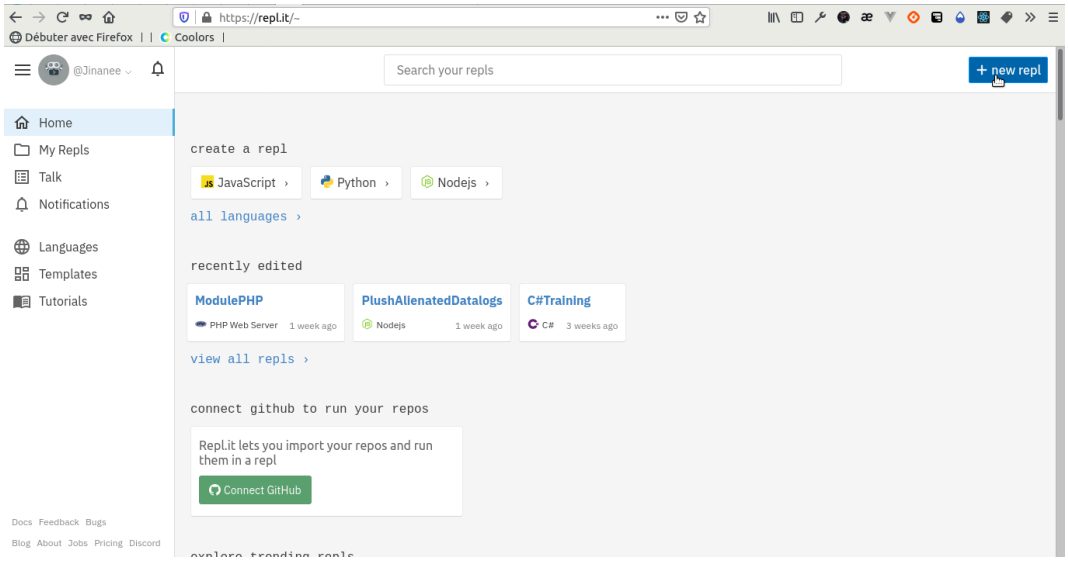
Méthode

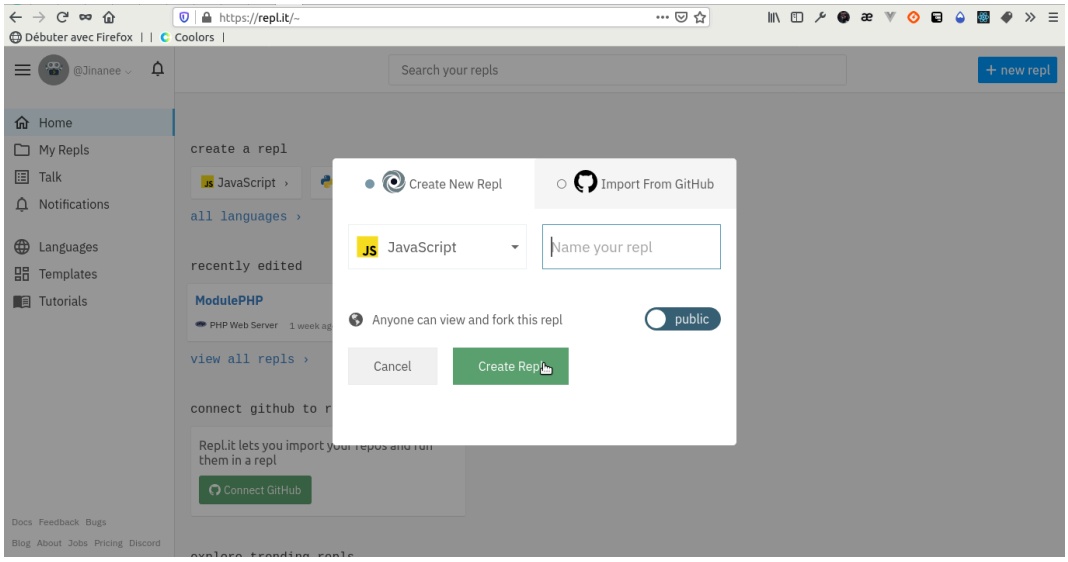
Grâce à Repl.it, il est possible de créer des environnements de développement selon le langage que l'on veut utiliser.

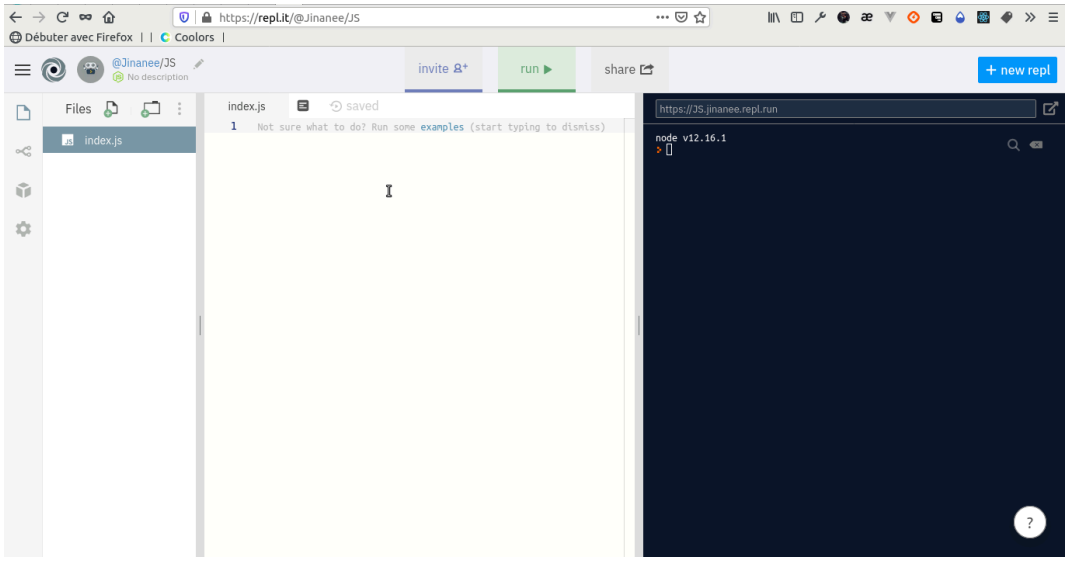
Sur Repl.it[2]:

1. Cliquer sur *« + new repl »* pour ouvrir la fenêtre de création.
2. Dans *« Language »*, sélectionner *« JavaScript »* et donner un nom au Repl.

2. https://repl.it/


https://repl.it/~
Search your repls
+ new repl
Home
My Repls
Talk
Notifications
Languages
Templates
Tutorials
create a repl
JavaScript
Python
Nodejs
all languages
recently edited
ModulePHP
PlushAlienatedDatalogs
C#Training
view all repls
connect github to run your repos
Repl.it lets you import your repos and run them in a repl
Connect GitHub

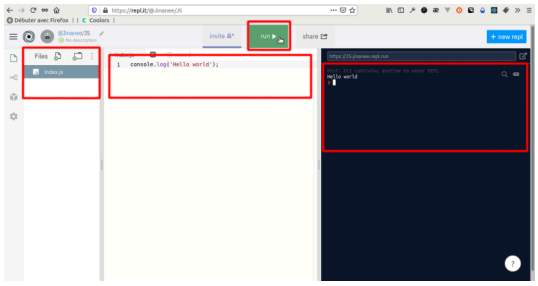

https://repl.it/~
Search your repls
+ new repl
Create New Repl
Import From GitHub
JavaScript
Name your repl
Anyone can view and fork this repl
public
Cancel
Create Repl


https://repl.it/@Jinanee/JS
@Jinanee/JS
invite
run
share
+ new repl
Files
index.js
https://JS.jinanee.repl.run
node v12.16.1

Écrire et lancer du code

Le code JavaScript s'écrit dans des fichiers dont le nom se termine par **.js** (c'est l'extension du fichier).

- À gauche, on voit la liste des fichiers existants. Les deux icônes permettent de créer des fichiers et des dossiers. Un fichier `index.js` a déjà été créé.
- Le bouton « *Run* » permet de lancer le programme.
- La partie principale au centre est l'éditeur de texte où est tapé le code.
- La partie de droite est la console où le résultat s'affiche quand le programme est lancé par « *Run* ».



Hello World Exemple

Copier le code ci-après dans le fichier `index.js` et cliquer sur « *Run* ».

Ce premier programme affiche dans la console le texte entre parenthèses.

```
1 console.log('Hello World')
```

Complément

Il en existe d'autres interpréteurs JavaScript en ligne :

- CodePen (https://codepen.io)
- JSFiddle (https://jsfiddle.net)
- JS Bin (https://jsbin.com)

À retenir

Un repl est créé pour un langage donné, le code écrit dans l'éditeur de texte peut être exécuté grâce au bouton *Run*. Le résultat s'affiche dans la console située à droite de l'écran.

IV Exercice

Question 1

[solution n°3 p. 34]

Exécuter ce programme avec Repl.it. Quel est le résultat obtenu ?

```
const languageJs = '...'
const languageC = '...'

console.log(languageC + ' est un langage compilé.')
console.log(languageJs + ' est un langage interprété.')
```

Question 2

[solution n°4 p. 34]

Modifier le code pour qu'il affiche :

```
C est un langage compilé.
JavaScript est un langage interprété.
```

Indice :

Il faut modifier la valeur des variables `languageJS` et `languageC`, située entre apostrophes ou *simple quote*.

V Syntaxe du JavaScript

Objectifs

- Apprendre les bases de la syntaxe du JavaScript.

Mise en situation

La syntaxe d'un langage est l'ensemble des **règles d'écritures** à respecter pour rédiger un code source valide. La manière de définir des variables ou des fonctions, le format d'écriture d'une opération, ou encore le respect ou non de la casse sont des exemples de règles de syntaxe dans un langage.

JavaScript est un langage dit de **haut niveau**, c'est à dire que sa syntaxe est plus proche du langage naturel que du langage machine. C'est entre autres ce qui le rend simple à prendre en main.

Instruction

Az Définition

Une instruction est une ligne de code qui effectue une action : un affichage, un calcul, etc.

Les instructions sont exécutées **séquentiellement**, c'est-à-dire dans l'ordre et une par une.

Variables et constantes

Az Définition

Une instruction peut utiliser des **variables**, des objets auxquels on peut attribuer une valeur (un nom, une date, un nombre, etc.) et que l'on déclare avec le mot clé `let`. Sa valeur peut changer au cours du programme.

Une **constante** ressemble à une variable sauf qu'elle doit forcément contenir une valeur dès sa création et que celle-ci ne peut plus être changée par la suite. On la déclare avec le mot clé `const`.

Déclaration de variables en JavaScript

Syntaxe

```
1 // Déclaration et affectation d'une variable nommée month
2 let month = 'Avril'
3 // Changement de la valeur de la variable month
4 month = 'Mai'
5
6 // Déclaration et affectation d'une variable constante nommée NB_MONTH
7 const NB_MONTH = 12
```

Expression

Az Définition

Une expression est un morceau de code qui produit un **résultat**, par exemple une addition. Le résultat d'une expression peut être stockée dans une variable.

Instruction avec expression

Exemple

La ligne représente une instruction. La partie de droite, `5 + 6`, est une expression.

```
let sum = 5 + 6
```

Fonctions

Définition

En JavaScript, certaines actions nécessitent de faire appel à des **fonctions** prédéfinies. Une fonction est un morceau de code ré-utilisable.

Les fonctions servent notamment à ne pas réécrire le code si on l'utilise plusieurs fois dans le programme : à la place, il suffit d'appeler la fonction en utilisant son nom, et les instructions qu'elle contient s'exécuteront.

Fonction d'affichage

Exemple

Pour afficher des choses dans la console, on utilise la fonction prédéfinie `console.log()`.

Entre les parenthèses, il faut indiquer la ou les informations à afficher.

```
// Affiche Hello World dans la console
console.log('Hello World')
// Affiche 4 dans la console
console.log(4)
```

Règles de syntaxe

Fondamental

Pour que les instructions soient exécutées, elles doivent être écrites dans une syntaxe correcte. Le JavaScript impose :

- De respecter la **casse**, c'est à dire les majuscules et minuscules. `console.Log()` n'équivaut pas à `console.log()`.
- Les **espaces** doivent être respectées : le nom d'une variable ne peut pas en contenir, en revanche elles sont nécessaires après des mots clés comme `let`.
- Il faut aller à la ligne entre deux instructions.

Exemple

```
const hello = 'Hello'
const world = 'World'
console.log(hello, world)
```

Standard ⚠ Attention

Dans des versions antérieures, les instructions JavaScript devaient se terminer par un point-virgule. Avec l'évolution du langage, le point-virgule n'est plus utile. On peut tout de même encore voir les deux écritures.

À retenir

Le JavaScript permet d'utiliser des variables, des fonctions, des opérations et les instructions. Leur écriture doit respecter des règles de syntaxe prédéfinies.

VI Exercice

On dispose du programme suivant, qui devrait afficher deux phrases :

```
1 console.log('Bogota est la capitale de la Colombie') console.log('Séoul est la capitale de la Corée du Sud')
```

Question 1

[solution n°5 p. 35]

Tester le programme. Que se passe-t-il ?

Question 2

[solution n°6 p. 35]

Trouver et corriger l'erreur du programme.

Indice :

Les instructions doivent être séparées correctement par un retour à la ligne.

Question 3

[solution n°7 p. 35]

Ajouter une instruction permettant d'afficher la capitale de la Norvège.

VII La casse

Objectif

- Passer en revue les différentes conventions de casses.

Mise en situation

Lorsque l'on développe, on est amené à nommer différents éléments de notre code, comme les variables ou les fonctions.

Selon les langages, la différenciation des noms en fonction des majuscules et des minuscules, c'est à dire **la casse**, n'est pas toujours la même.

Comme les espaces ne sont pas utilisables dans des noms, différentes conventions se sont mises en place pour séparer différents mots dans un même nom.

Sensibilité Fondamental

JavaScript est **sensible à la casse** ; il faut donc faire attention aux majuscules/minuscules (mots clés, variables, fonctions, etc.).

Par exemple, le mot clé `let` (qui crée une variable) ne peut pas être remplacé par `Let` (qui n'existe pas).

Attention

On peut créer deux variables contenant les mêmes caractères mais avec une casse différente.

Casse Syntaxe

Les espaces servant à séparer les mots du langages, on ne peut pas les utiliser pour les variables. Pour les remplacer, on accole plusieurs mots et on utiliser une convention pour marquer leur séparation :

- ***camel case*** : première lettre du premier mot en minuscule, les premières lettres des mots suivants en majuscule (ex : `albumName, parseInput`).
- ***pascal case*** : identique à la casse camel, sauf que la toute première lettre est en majuscule aussi (ex : `AlbumName`, `ParseInput`).
- ***snake case*** : les espaces sont remplacés par des underscores « _ » (ex : `album_name`, `parse_input`).
- ***kebab case*** : les espaces sont remplacés par des tirets - (ex : `album-name`, `parse-input`).

Le choix d'une convention doit être fixée pour le programme entier : on choisira un type de convention pour le nommage des variables et fonctions, une convention pour le nom des fichiers, etc.

Variations des conventions Remarque

Certaines conventions sont plus populaires pour certains langages : snake case en Python, camel case en JavaScript, etc.

Camel case en JavaScript Exemple

```
1 let firstPlayer = 'Leo'
```

Snake case en Python Exemple

```
1 first_player = 'Leo'
```

À retenir

Le choix de la convention de casse permet d'uniformiser les programmes ainsi que d'écrire des noms plus détaillés et lisibles.

VIII Exercice

On dispose du programme suivant :

```
1 const goodnight_french = 'Bonne nuit'
2 const goodnight_japanese = 'Oyasuminasai'
3 console.log(goodnight_french, goodnight_japanese)
4
```

Question 1

[solution n°8 p. 35]

Quel est la convention de casse utilisée ?

Question 2

[solution n°9 p. 35]

Modifier le code pour qu'il utilise la convention *camelCase*.

IX Indentation du code

Objectif

- Connaître les conventions d'indentation du code.

Mise en situation

Le métier de développeur ne consiste pas simplement à écrire du code. Il arrive régulièrement qu'il soit nécessaire d'en lire, par exemple pour comprendre le fonctionnement d'un programme. Il est donc important de donner une bonne lisibilité à son code, pour permettre aux autres ou à soi-même de le relire facilement. Pour cela différentes règles doivent être respectées, comme par exemple l'**indentation**. Elle consiste à ajouter des espaces au début des lignes, pour identifier clairement les différents **blocs** de code.

Indentation

Fondamental

La lecture d'un code doit être facilitée visuellement par une bonne **indentation** : c'est une manière de formater les lignes et d'arranger les blocs de code en utilisant un nombre spécifique d'espaces et de tabulations. Le résultat doit permettre d'identifier d'un simple coup d'œil les différents niveaux du code, en particulier lorsque l'on imbrique des blocs dans d'autres.

L'indentation est la plupart du temps **esthétique** : elle ne fait que mettre en forme le code.

Néanmoins, dans certains langage comme Python elle **modifie** le sens du contenu.

Code non-indenté VS code indenté

Exemple

```
// Code mal indenté
let surname = 'First'
let firstname = 'Alexander'

if (surname && firstname) {
console.log('Name :')
console.log(firstname, surname)
}

```

```
// Code correctement indenté
let surname = 'First'
let firstname = 'Alexander'

if (surname && firstname) {
  console.log('Name :')
  console.log(firstname, surname)
}

```

Grâce à l'indentation on perçoit mieux que les deux affichages `console.log` dépendent de la condition (`if`).

Règles générales

Méthode

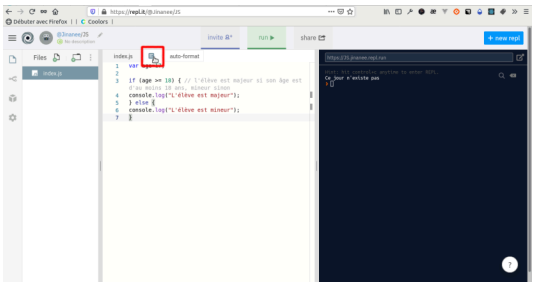
Bien que le style d'indentation puisse légèrement varier, il existe des conventions populaires :

- Utiliser des espaces (qui espaceront de manière identique partout) plutôt que des tabulations (qui pourront rendre un résultat différent selon les environnements).
- Utiliser deux espaces pour l'indentation.
- À chaque fois qu'un bloc est imbriqué dans un autre, l'indenter avec les 2 espaces supplémentaires.
- Ne pas sauter plusieurs lignes entre des instructions.

Auto-format

Méthode

Sur Repl.it, une option permet rapidement de corriger l'indentation du code. Pour cela, cliquer sur l'icône « *auto-format* » à droite du nom du fichier.



À retenir

L'indentation ne doit pas être négligée car, sans elle, un code peut rapidement devenir difficile à lire et donc à comprendre.

X Exercice

On dispose du programme suivant, qui est peu lisible du fait d'une mauvaise indentation :

```
const number = 10
if (number > 0) {
console.log('Le numéro est positif')
if (number < 9) {
console.log('Le numéro est un chiffre')}
}

```

Question 1

[solution n°10 p. 35]

Corriger l'indentation du programme.

Indice :

Remarquer que le code contient deux blocs imbriqués, et non deux blocs l'un à la suite de l'autre. Il y a aura donc **deux niveaux** d'indentation.

Question 2

[solution n°11 p. 36]

On utilise des espaces pour l'indentation. Si la convention classique est respectée, combien d'espaces précèdent la deuxième instruction *« console.log »* ?

XI Les commentaires

Objectif

- Comprendre l'utilité des commentaires.

Mise en situation

Un programme qui s'allonge peut rapidement devenir difficile à relire, à comprendre et à modifier.

Commentaires

Fondamental

Commenter son code est une bonne pratique à adopter pour conserver un **code explicite** : préciser le rôle d'une fonction, expliquer une opération réalisée sur plusieurs instructions, etc.

Les commentaires sont indispensables en particulier pour des programmes complexes, destinés à être réutilisés ou modifiés par plusieurs personnes ; ils permettent de rendre un programme ou un bloc de code plus clair en utilisant la langue naturelle (français, anglais, etc.).

Il existe trois types de commentaires en JavaScript :

- Les commentaires en bloc ;
- Les commentaires en ligne ;
- Les commentaires de fin de ligne.

Commentaires en JavaScript

Syntaxe

```
/*
  Commentaire en bloc
*/

// Commentaires en ligne

console.log('Hello World') // Commentaire de fin de ligne
```

Méthode

Pour être efficaces, les commentaires doivent donner suffisamment d'informations sans être trop détaillés. Pour cela :

- Ne pas commenter chaque ligne mais plutôt un ensemble d'instructions qui effectuent une opération spécifique ou plutôt complexe.
- Ne pas expliciter tous les détails comme le rôle des variables : pour cela, il est préférable de judicieusement choisir leur nom. Cela aidera à la compréhension sans alourdir de commentaires.

- Les instructions commentées doivent en revanche être supprimées si elles ne sont pas vouées à être utilisées : les laisser ajoute inutilement des lignes et nuit à la compréhension.

Supprimer temporairement des instructions ⊕ Complément

Les commentaires servent aussi à masquer temporairement des lignes de code que l'on souhaite temporairement ignorer, par exemple à des fins de tests ou pour isoler un problème. Cette méthode permet d'éviter de les supprimer de les réécrire ces même lignes.

À retenir

Les commentaires doivent expliciter ce qui ne l'est pas dans le code et ne doivent pas alourdir le code.

XII Exercice

On dispose d'un programme, qui ne contient aucun commentaire, et d'une liste de commentaires désordonnés.

```
const day = 4
const month = 'Mai'
const year = '2020'

console.log('Jour J: ' + day + ' ' + month + ' ' + year)

const nextDay = day + 1
console.log('Jour J+1: ' + day + ' ' + month + ' ' + year)

```

```
// Affichage de la date du lendemain
// Incrémentation d'un jour
// Affichage de la date
// Définition des constantes
```

Question 1

[solution n°12 p. 36]

Insérer les commentaires à leur place dans le programme.

Question 2

[solution n°13 p. 36]

De quel type sont ces commentaires ?

XIII Bonnes pratiques

Objectif

- Connaître les bonnes pratiques d'écriture de code JavaScript.

Mise en situation

Il existe des **bonnes pratiques** pour chaque langage de programmation. Les bonnes pratiques sont des règles d'écriture du code qui ne sont pas imposées par le langage. Elles sont considérées comme importantes par la communauté. Il est fortement recommandé de se renseigner et de suivre les bonnes pratiques des langages qu'on utilise, qui servent généralement à améliorer la lisibilité du code source. De plus, cela permet d'avoir un style de code commun au sein d'une équipe, ou même d'une communauté, et ainsi de faciliter la collaboration.

Référence

Fondamental

Des bonnes pratiques populaire de JavaScript sont regroupées ici : https://standardjs.com/rules-fr.html#javascript-standard-style.

Chaînes de caractère

Méthode

Les chaînes de caractères doivent être entourées d'apostrophes. Les guillemets ne sont utilisés que si la chaîne contient elle-même une apostrophe.

Exemple

```
1 console.log("Hello World") // ✗ non
2
3 console.log('Hello World') // ✓ ok
4 console.log("Sans l'apostrophe") // ✓ ok
```

Variables

Méthode

- Toutes les variables doivent être déclarées.
- On privilégie *const* à *let* si la variable est une constante.
- Toutes les variables qui sont déclarées doivent forcément être utilisées à un moment dans le programme.

Exemple

```
const useless = 'Null' // ✗ non

const useful = 'Hi' // ✓ ok
console.log(useful)

```

Espacements Méthode

Les mots clés, les opérateurs, les accolades, les virgules, etc., doivent être espacés.

Exemple

```
// ✗ mauvaises pratiques
const color1='blanc'
let flower1='jasmin'
if(color1==='rose'){flower1='rose'}
console.log(flower1)

// ✓ bonnes pratiques
const color2 = 'rouge'
let flower2 = 'tulipe'
if (color2 === 'jaune') {
  flower2 = 'mimosa'
}
console.log(flower2)

```

Casse Méthode

La convention *camelCase* est privilégiée en JavaScript.

Anglais Méthode

- Le nom des variables, des fonctions, etc., doit être en anglais.
- Les commentaires doivent de préférence être en anglais (sauf si tous les développeurs préfèrent une autre langue).
- Les valeurs (chaînes de caractères) sont bien entendu dans la langue requise par le programme.

À retenir

Il est important de se référer aux bonnes pratiques qui permettent de produire un code propre et lisible par tous.

XIV Exercice : Appliquer la notion

On dispose du programme suivant :

```
const year =2020
const days_in_week =7
const hours_in_day =24

console.log('Une semaine contient ' +days_in_week+ " jours")
console.log('Un jour contient ' +hours_in_day+ " heures")
```

Question

[solution n°14 p. 36]

Corriger le code pour qu'il suive les bonnes pratiques du JavaScript.

Indice :

Il faut corriger la casse des variables, l'espacement entre les opérateurs, les variables inutilisées et les délimiteurs de chaînes de caractères.

XV Quiz

Exercice 1 : Quiz - Culture

[solution n°15 p. 36]

Exercice

JavaScript est un langage :

- **A** Notamment utilisé pour le Web
- **B** Uniquement utilisé pour le Web
- **C** Compilé
- **D** Interprété

Exercice

ECMAScript est :

- **A** Une variante du JavaScript
- **B** Une convention de l'usage de JavaScript
- **C** Une spécification qui standardise JavaScript
- **D** L'ancêtre de JavaScript

Exercice

Qu'est-ce qu'un IDE ?

- **A** Un ensemble d'outils pour le développement
- **B** Une plate-forme en ligne pour partager du code
- **C** Un logiciel dédié au développement JavaScript

Exercice

Quel nom est écrit en *pascal case* ?

- **A** `nomFamille`

- **B** `Nom_Famille`
- **C** `nom_famille`
- **D** `nom-famille`
- **E** `NomFamille`

Exercice 6 : Quiz - Méthode

[solution n°16 p. 37]

Exercice

Quelles affirmations sont vraies ?

- **A** Une expression est une instruction en JavaScript.
- **B** La valeur d'une constante ne peut pas être modifiée dans le programme.
- **C** ECMAScript est un langage qui descend du JavaScript.
- **D** Le point-virgule à la fin des instructions est issu d'une ancienne syntaxe et il n'est plus utile de le mettre.

Exercice

Il peut être utile de commenter son code pour :

- **A** Expliquer le rôle de chaque variable
- **B** Expliciter la logique globale du programme
- **C** Désactiver temporairement des instructions
- **D** Expliquer une instruction complexe

Exercice

Quelles règles sont des bonnes pratiques ?

- **A** Ne jamais utiliser de majuscule dans le nom d'une variable
- **B** Utiliser une casse différente pour les fonctions et pour les variables

C Déclarer les variables sur une même ligne

D Insérer un espace après une virgule

E Utiliser en priorité des apostrophes plutôt que des guillemets

Exercice 10 : Quiz - Code

[solution n°17 p. 38]

Exercice

Quel(s) code(s) produi(sen)t une erreur ?

A
```
variable age = 17 ; console.log(age) ;
```

B
```
const AGE = "17" ; console.log(AGE) ;
```

C
```
const AGE = 17 ; console.log(AGE) ;
```

D
```
age = 17 ; console.log(age) ;
```

Exercice

Quel(s) code(s) respecte(nt) les bonnes pratiques syntaxiques du JavaScript ?

A
```
const nomFamille="Charlie";
```

B
```
const nom_deFamille = 'Charlie';
```

C
```
const nomFamille = 'Charlie'
```

D
```
let nom = 'Charlie';
```

Exercice

Quelle fonction permet d'afficher des informations dans la console JavaScript ?

A
```
screen.print()
```

B
```
console.print()
```

C
```
console.log()
```

D
```
window.log()
```

XVI Exercice : Défi

On dispose d'un programme permettant de décider, pour une personne donnée, si elle peut faire une attraction à sensations fortes. Pour cela, il effectue une suite de tests et affiche un résultat en fonction des conditions remplies.

La structure `if/else` permet de tester si une expression est vraie ou fausse et d'exécuter des blocs de code différents en fonction de ce résultat.

```
const age=17
const taille = 152
let prixbillet
let entree_autorise = false

if (age>8) {
  if (taille > 150) {
  entree_autorise = true
  prixbillet = 40 + age
  } else {
    console.log('Vous êtes trop petit.')
  }
} else {
console.log('Vous êtes trop jeune.')
}


if (entree_autorise) {
if (age<18) {
  prixbillet = prixbillet-10
}
  console.log('Vous pouvez avoir une place pour ' + prixbillet + '$')
}
```

Question 1

[solution n°18 p. 39]

Tester le code. Quel est le résultat obtenu ?

Question 2

[solution n°19 p. 40]

Ajuster la casse du code pour respecter les bonnes pratiques JavaScript.

Indice :

La convention de casse utilisée en JavaScript est le `camelCase`.

Question 3

[solution n°20 p. 40]

Corriger l'indentation et les espaces dans le code pour respecter les bonnes pratiques JavaScript.

Question 4

[solution n°21 p. 40]

Placer les commentaires, là où ils sont pertinents, en remplissant les parties manquantes si besoin.

```
// Déclaration des constantes

// Déclaration des ...

// Tests des critères d'admission selon l'age et la taille

// Application de la réduction pour les moins de ... ans
```

Question 5

[solution n°22 p. 41]

Traduire le nom des variables en anglais.

Indice :

En faisant un clic-droit sur le nom d'une variable dans Repl.it, il est possible de modifier d'un coup toutes les occurrences de cette variable (*Change All Occurrences*).

Conclusion

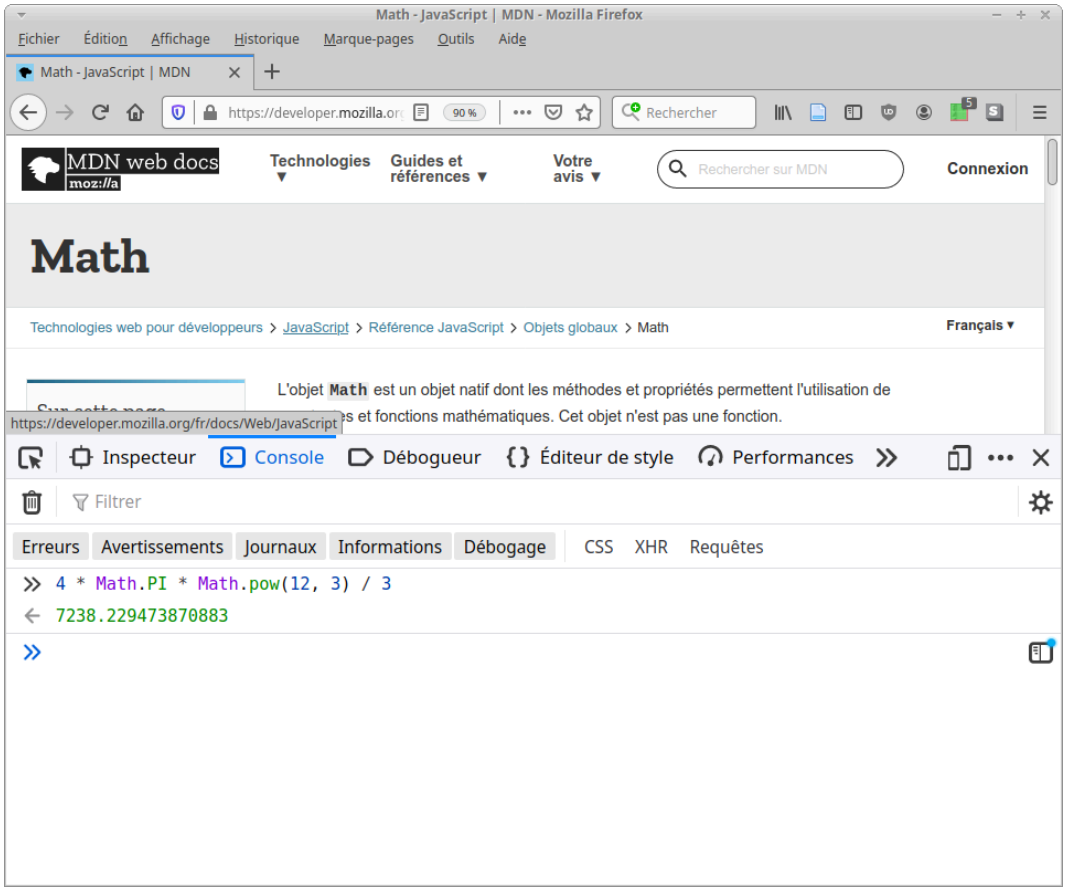
JavaScript est un langage simple et polyvalent. Historiquement très utilisé dans le Web, il permet de dynamiser les sites depuis des années. Plus récemment, ses caractéristiques ont favorisé son adoption importante pour tous types d'usages et sa popularité pour l'apprentissage de la programmation. Comme tous les langages, il nécessite qu'une syntaxe soit respectée, et il est recommandé de suivre différentes bonnes pratiques de programmation.

Solutions des exercices

Solution n°1

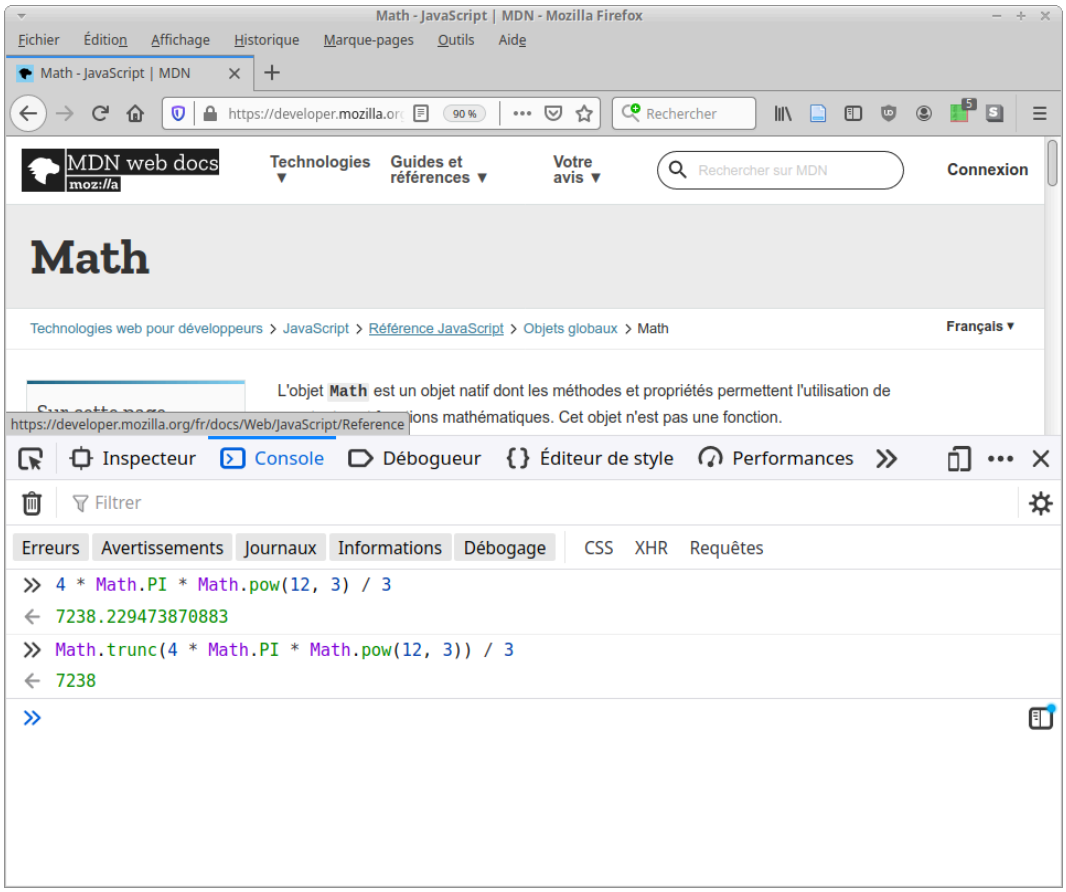
[exercice p. 6]

7238



Solution n°2

[exercice p. 6]



`Math.trunc` réalise une troncature d'un nombre : elle coupe la partie décimale (*trunc* vient de l'anglais *truncate*).

Solution n°3

[exercice p. 10]

```
... est un langage compilé.
... est un langage interprété.
```

Solution n°4

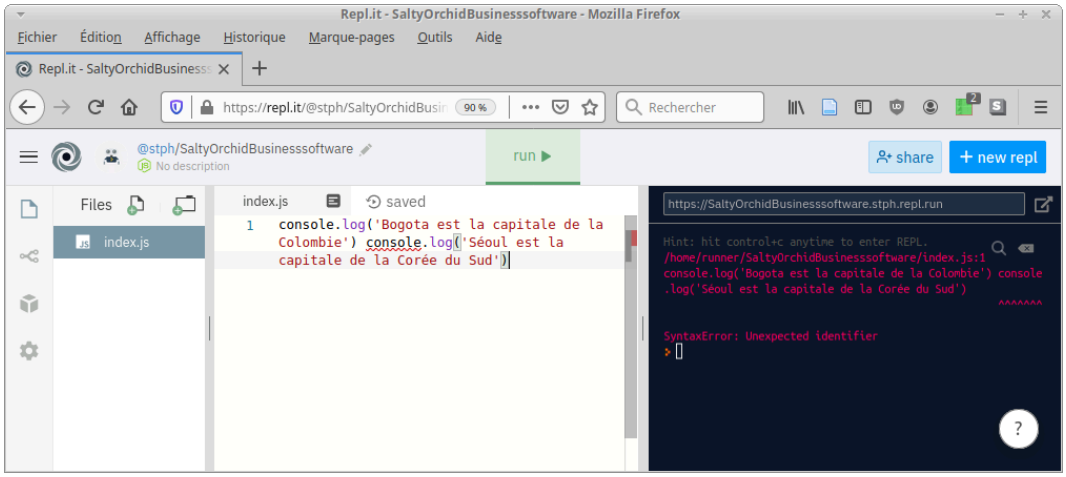
[exercice p. 10]

```
1 const languageJs = 'JavaScript'
2 const languageC = 'C'
3
4 console.log(languageC + ' est un langage compilé.')
5 console.log(languageJs + ' est un langage interprété.')
```

Solution n°5

[exercice p. 14]

Le programme ne fonctionne pas : une erreur est retournée, affichée en rouge.



Solution n°6

[exercice p. 14]

```
1 console.log('Bogota est la capitale de la Colombie.')
2 console.log('Séoul est la capitale de la Corée du Sud.')
```

Solution n°7

[exercice p. 14]

```
1 console.log('Bogota est la capitale de la Colombie.')
2 console.log('Séoul est la capitale de la Corée du Sud.')
3 console.log('Oslo est la capitale de la Norvège.')
```

Solution n°8

[exercice p. 17]

La convention *snake_case* est utilisée.

Solution n°9

[exercice p. 17]

```
1 const goodnightFrench = 'Bonne nuit'
2 const goodnightJapanese = 'Oyasuminasai'
3 console.log(goodnightFrench, goodnightJapanese)
4
```

Solution n°10

[exercice p. 20]

```
1 const number = 10
2 if (number > 0) {
3   console.log('Le numéro est positif')
4   if (number < 9) {
5     console.log('Le numéro est un chiffre')
6   }
7 }
8
```

Solution n°11

[exercice p. 20]

4 espaces : 2 espaces pour le premier bloc, 2 autres espaces pour le deuxième bloc.

Solution n°12

[exercice p. 23]

```
1  // Définition des constantes
2  const day = 4
3  const month = 'Mai'
4  const year = '2020'
5  // Affichage de la date
6  console.log('Jour J : ' + day + ' ' + month + ' ' + year)
7
8  // Incrémentation d'un jour
9  const nextDay = day + 1
10 // Affichage de la date du lendemain
11 console.log('Jour J+1 : ' + nextDay + ' ' + month + ' ' + year)
12
```

Solution n°13

[exercice p. 23]

Ce sont tous des commentaires en ligne.

Solution n°14

[exercice p. 26]

```
1  const daysInWeek = 7
2  const hoursInDay = 24
3
4  console.log('Une semaine contient ' + daysInWeek + ' jours')
5  console.log('Un jour contient ' + hoursInDay + ' heures')
6
```

Solution n°15

[exercice p. 27]

Exercice

JavaScript est un langage :

- **A** Notamment utilisé pour le Web
- **B** Uniquement utilisé pour le Web
- **C** Compilé
- **D** Interprété

Exercice

ECMAScript est :

- **A** Une variante du JavaScript
- **B** Une convention de l'usage de JavaScript
- **C** Une spécification qui standardise JavaScript
- **D** L'ancêtre de JavaScript

Exercice

Qu'est-ce qu'un IDE ?

- **A** Un ensemble d'outils pour le développement
- **B** Une plate-forme en ligne pour partager du code
- **C** Un logiciel dédié au développement JavaScript

Exercice

Quel nom est écrit en *pascal case* ?

- **A** `nomFamille`
- **B** `Nom_Famille`
- **C** `nom_famille`
- **D** `nom-famille`
- **E** `NomFamille`

Solution n°16

[exercice p. 28]

Exercice

Quelles affirmations sont vraies ?

- **A** Une expression est une instruction en JavaScript.
- **B** La valeur d'une constante ne peut pas être modifiée dans le programme.

C ECMAScript est un langage qui descend du JavaScript.

D ☐ Le point-virgule à la fin des instructions est issu d'une ancienne syntaxe et il n'est plus utile de le mettre.

Exercice

Il peut être utile de commenter son code pour :

A

Expliquer le rôle de chaque variable	Il est préférable que le nom de la variable explique directement son rôle.

B ☐

Expliciter la logique globale du programme	Il est courant de rencontre un bloc de commentaires détaillant le rôle global d'un programme.

C Désactiver temporairement des instructions

D Expliquer une instruction complexe

Exercice

Quelles règles sont des bonnes pratiques ?

A Ne jamais utiliser de majuscule dans le nom d'une variable

B Utiliser une casse différente pour les fonctions et pour les variables

C Déclarer les variables sur une même ligne

D Insérer un espace après une virgule

E Utiliser en priorité des apostrophes plutôt que des guillemets

Solution n°17

[exercice p. 29]

Exercice

Quel(s) code(s) produi(sen)t une erreur ?

A ☐

`variable age = console.log(age) ; 17 ;`	Le bon mot clé pour déclarer une variable est `var`.

B `const AGE = "17" ; console.log(AGE) ;`

C `const AGE = 17 ; console.log(AGE) ;`

D

`age = console.log(age) ; 17 ;` Même si c'est une bonne pratique, `age` n'a pas besoin d'être créé avec `let` ou `const` pour lui donner une valeur.

Exercice

Quel(s) code(s) respecte(nt) les bonnes pratiques syntaxiques du JavaScript ?

A

`const nomFamille="Charlie";` Le symbole = devrait être espacé et la chaîne de caractères être entre apostrophes.

B

`const nom_deFamille = 'Charlie';` La variable ne respecte pas la convention *camelCase*.

C `const nomFamille = 'Charlie'`

D

`let nom = 'Charlie';` Les points-virgules ne sont plus utilisés à la fin des instructions.

Exercice

Quelle fonction permet d'afficher des informations dans la console JavaScript ?

A `screen.print()`

B `console.print()`

C `console.log()`

D `window.log()`

Solution n°18

[exercice p. 30]

« Vous pouvez avoir une place pour 47 $. »

Solution n°19

[exercice p. 30]

```
const age=17
const taille = 152
let prixBillet
let entreeAutorise = false

if (age>8) {
  if (taille > 150) {
  entreeAutorise = true
  prixBillet = 40 + age
  } else {
    console.log('Vous êtes trop petit.')
  }
} else {
console.log('Vous êtes trop jeune.')
}


if (entreeAutorise) {
if (age<18) {
  prixBillet = prixBillet-10
}
  console.log('Vous pouvez avoir une place pour ' + prixBillet + '$')
}
```

Solution n°20

[exercice p. 30]

```
const age = 17
const taille = 152
let prixBillet
let entreeAutorise = false

if (age > 8) {
  if (taille > 150) {
    entreeAutorise = true
    prixBillet = 40 + age
  } else {
    console.log('Vous êtes trop petit.')
  }
} else {
  console.log('Vous êtes trop jeune.')
}

if (entreeAutorise) {
  if (age < 18) {
    prixBillet = prixBillet - 10
  }
  console.log('Vous pouvez avoir une place pour ' + prixBillet + '$')
}
```

Solution n°21

[exercice p. 31]

```
// Déclaration des constantes
const age = 17
const taille = 152

// Déclaration des variables
let prixBillet
```

```
let entreeAutorise = false

// Tests des critères d'admission selon l'age et la taille
if (age > 8) {
  if (taille > 150) {
    entreeAutorise = true
    prixBillet = 40 + age
  } else {
    console.log('Vous êtes trop petit.')
  }
} else {
  console.log('Vous êtes trop jeune.')
}

if (entreeAutorise) {
  // Application de la réduction pour les moins de 18 ans
  if (age < 18) {
    prixBillet = prixBillet - 10
  }
  console.log('Vous pouvez avoir une place pour ' + prixBillet + '$')
}

```

Solution n°22

[exercice p. 31]

```
// Déclaration des constantes
const age = 17
const size = 152

// Déclaration des variables
let ticketPrice
let entranceAllowed = false

// Tests des critères d'admission selon l'age et la size
if (age > 8) {
  if (size > 150) {
    entranceAllowed = true
    ticketPrice = 40 + age
  } else {
    console.log('Vous êtes trop petit.')
  }
} else {
  console.log('Vous êtes trop jeune.')
}

if (entranceAllowed) {
  // Application de la réduction pour les moins de 18 ans
  if (age < 18) {
    ticketPrice = ticketPrice - 10
  }
  console.log('Vous pouvez avoir une place pour ' + ticketPrice + '$')
}

```

Crédits des ressources

Schéma d'un langage interprété p. 4

Universel - Transfert dans le Domaine Public - Culture Informatique - https://www.culture-informatique.net/cest-quoi-langage-de-programmation/

Contenus annexes

1. Repl.it

Objectifs

- Savoir exécuter un programme avec Repl.it.

Mise en situation

On pourrait choisir d'écrire ses programmes avec un éditeur de texte basique, cela suffit et dans le passé on procédait ainsi. Néanmoins on dispose aujourd'hui d'outils plus confortables et plus efficaces.

Le minimum est de disposer d'un **éditeur colorisant** qui met en évidence la syntaxe du langage et évite les erreurs bêtes. Mais il existe des outils encore plus puissants que les éditeurs de base qui automatisent de nombreuses tâches, les IDE ou **environnements de développement intégrés**. Ils permettent de plus facilement exécuter et déboguer le code que l'on écrit.

Ce module présente l'IDE en ligne **Repl.it** qui a l'avantage d'être rapide à prendre en main et qui prend en charge de très nombreux langages de programmation, dont les principaux langages du web, comme HTML, CSS, JavaScript, PHP, Python, Java, C, etc.

IDE Az Définition

Un **IDE** (Environnement de Développement Intégré ou *Integrated Development Environment* en anglais) est un logiciel qui regroupe plusieurs outils utiles au développement de programmes informatiques. Très souvent, il comporte :

- Un **éditeur de texte colorisant** pour écrire les code source et mettre certaines parties en valeur.
- Un **débogueur** pour analyser le fonctionnement du programme et trouver des bogues.
- Un **compilateur** ou un **interpréteur** pour transformer le code en instructions exécutables par l'ordinateur.
- Des fonctions de recherche, d'optimisation du code ou de détection des erreurs, qui facilitent la vie du développeur.

Repl.it Az Définition

Repl.it est un **IDE** en ligne qui permet d'écrire et d'exécuter des programmes depuis un navigateur web. Il a l'avantage de ne pas nécessiter d'installation sur le poste de travail.

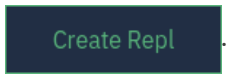
Repl.it permet :

- De créer des espaces de stockage et d'exécution de programme, appelés **repls**.
- De travailler avec de nombreux langages, comme Python, JavaScript, C, Java, PHP, etc.

Créer et exécuter un programme sur Repl.it

Méthode

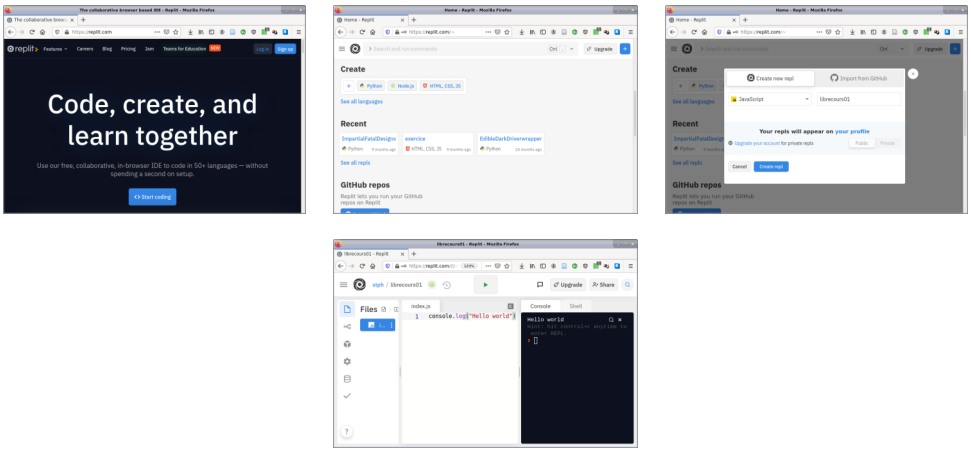
1. Se rendre sur le site Repl.it[3].
2. Se connecter (ou créer un nouveau compte).
3. Cliquer sur 

pour créer un nouveau repl.
4. Choisir le langage et cliquer sur 

.
5. Écrire le programme dans l'espace au milieu.
6. Cliquer sur  dans la barre supérieure pour exécuter le programme.
7. Visualiser le résultat dans l'espace à droite.

Créer et exécuter un programme avec Repl.it

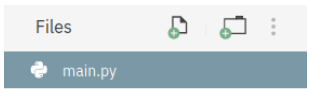
Exemple



Exécuter un programme avec Repl.it

Ajouter un fichier sur Repl.it

Méthode



Lors de la création d'un repl, Repl.it crée automatiquement un fichier dans lequel on peut écrire du code. Il est visible dans l'arborescence de fichiers qui est affichée à gauche de l'interface.

3. https://repl.it/

Un programme informatique est rarement composé d'un unique fichier, qui deviendrait trop gros. Plusieurs solutions existent pour créer de nouveaux fichiers :

- Créer un fichier directement depuis l'interface, avec le bouton  .
- Importer un fichier existant sur votre ordinateur en sélectionnant *Upload File* après avoir cliqué sur le bouton  .

Autres IDE en ligne

Complément

- Il existe d'autres IDE en ligne, comme par exemple CodeChef[4] ou GeeksForGeeks[5]. Le principe est exactement le même, seules les fonctionnalités diffèrent.
- Certains IDE sont spécialisés et proposent des fonctionnalités spécifiques à un langage. Par exemple, JSFiddle[6] ou PlayCode[7] permettent de tester des pages web en proposant des outils avancés pour le développement web.

IDE locaux

Complément

Les IDE existent aussi comme des logiciels à installer sur son poste de travail. Ils présentent plusieurs avantages :

- pas besoin de connexion Internet,
- meilleure performance (pas d'usage du réseau et pas de partage de serveur),
- débogage plus facile car l'exécution du programme se fait *« en local »*,
- confidentialité du code qui reste sur l'ordinateur.

La plupart de ces IDE sont spécialisés dans un langage de programmation.

- On peut citer Code::Blocks[8] pour les langages C et C++ ou PyCharm[9] pour le langage Python.
- Il existe aussi des IDE généralistes, comme Atom[10], Visual Studio Code[11] ou Eclipse[12], qui prennent en charge une grande variété de langages à travers un système d'extensions.

Conseil

Pour la réalisation de programmes en situation réelle on conseille l'usage d'un IDE local.

4. CodeChef - https://www.codechef.com
5. GeeksForGeeks - https://ide.geeksforgeeks.org/
6. JSFiddle - https://jsfiddle.net/
7. PlayCode - https://playcode.io/
8. CodeBlocks - http://www.codeblocks.org/
9. PyCharms - https://www.jetbrains.com/pycharm/
10. Atom - https://atom.io/
11. Visual Studio Code - https://code.visualstudio.com/
12. Eclipse - https://www.eclipse.org/downloads/

Repl Bash ⊕ Complément

Pour travailler en Bash sous Windows sans avoir installé le sous-système Linux, il est possible d'utiliser Bash dans Repl.it.

À retenir

- Un IDE est un programme qui facilite la tâche des développeurs en proposant diverses fonctionnalités utiles (débogage, coloration, analyse d'erreurs, etc.).
- Repl.it est un IDE en ligne qui prend en charge une très grande variété de langages.
- Il existe des IDE locaux qui n'ont pas besoin de connexion Internet pour fonctionner.