JAVASCRIPT

FONDAMENTAUX

DÉBUTANT

SOMMAIRE

INTRODUCTION

JAVASCRIPT

INCONTOURNABLE DU DÉVELOPPEMENT WEB

C'est un langage de programmation de haut niveau

Son typage est dynamique faible

Il s'exécute sur tous les navigateurs (Chrome, Firefox...)

Il est possible de l'exécuter sur un serveur à l'aide de Node.js

https://www.w3schools.com/js/js_es6.asp

ECMASCRIPT

NORME JAVASCRIPT

ECMAScript est un ensemble de normes concernant les langages de programmation de type script et standardisées par Ecma International.

Il s'agit donc d'un standard, dont les spécifications sont mises en œuvre dans différents langages de script, comme JavaScript.

CONTEXTE D'EXÉCUTION

RUNTIME

Les navigateurs disposent d'un moteur d'exécution ou runtime permettant d'exécuter le langage JavaScript.

Ils peuvent être différents en fonction des navigateurs, cela explique dans de rares cas une différence de rendu.

En 2009, Ryan Dahl a créé Node, un programme en C++ permettant d'exécuter du code JavaScript côté serveur. Node à révolutionné le monde du développement web, en donnant à JavaScript la possibilité d'être à la fois un langage frontend et backend.

Moteur V8
Google

SpiderMonkey
Firefox

NodeJS
OpenAI

MISE EN PLACE DE L'ENVIRONNEMENT

CHOISIR UN NAVIGATEUR

OÙ NOTRE CODE VA S'EXÉCUTER ?

Firefox
Mozilla Corp.

Opera
Opera Software

Google Chrome
Google

Microsoft Edge
Microsoft

Veuillez éviter Microsoft Edge...
Les outils développeurs ainsi que le rendu de certains éléments ne sont pas à jour.

CHOISIR UN IDE / ÉDITEUR DE TEXTE

ENVIRONNEMENT DE DÉVELOPPEMENT

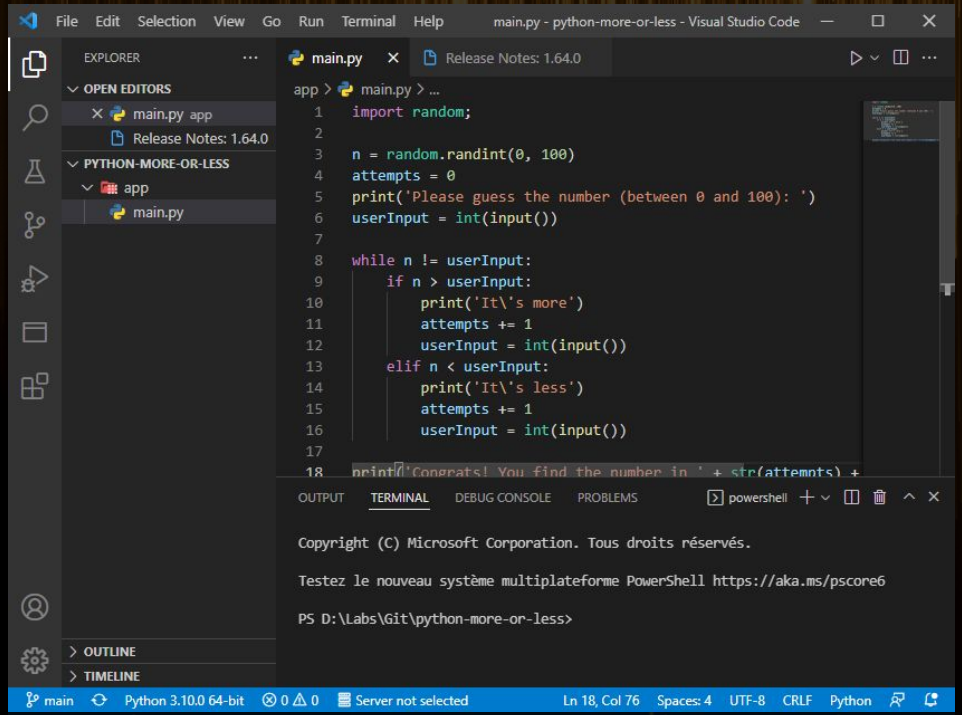
WebStorm
JetBrains

Visual Studio Code
Microsoft

https://code.visualstudio.com/

VISUAL STUDIO CODE

INTERFACE & RACCOURCIS



Gratuit

Système d'extensions

Connecté au système de versionning

Le plus populaire

Et bien plus...

https://nodejs.org/fr/

NODE JS

RUNTIME JAVASCRIPT

Node JS
OpenJS Foundation

https://nodejs.org/fr/

⚠ Lors de l'installation de Node.js, veillez à bien ajouter le chemin parmi les variables d'environnement système.

TERMINAL / CMD

```
node -v
v19.0.1
--
```

VÉRIFICATION DE L'INSTALLATION

NodeJS est un runtime JavaScript basé sur le moteur V8 de Google Chrome.
Il va nous être nécessaire pour avoir npm qui sera notre gestionnaire de packages / dépendances.

EXTENSION CODE RUNNER

CÔTÉ PRATIQUE

JavaScript

PHP

Python

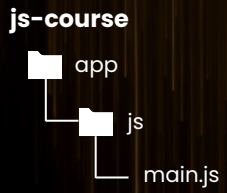
TypeScript

Et bien plus...

Une fois l'extension installée, il suffira de sélectionner dans le fichier JavaScript le code à exécuter, et cliquer sur run code.

CRÉATION DU PROJET

KICK START JAVASCRIPT !



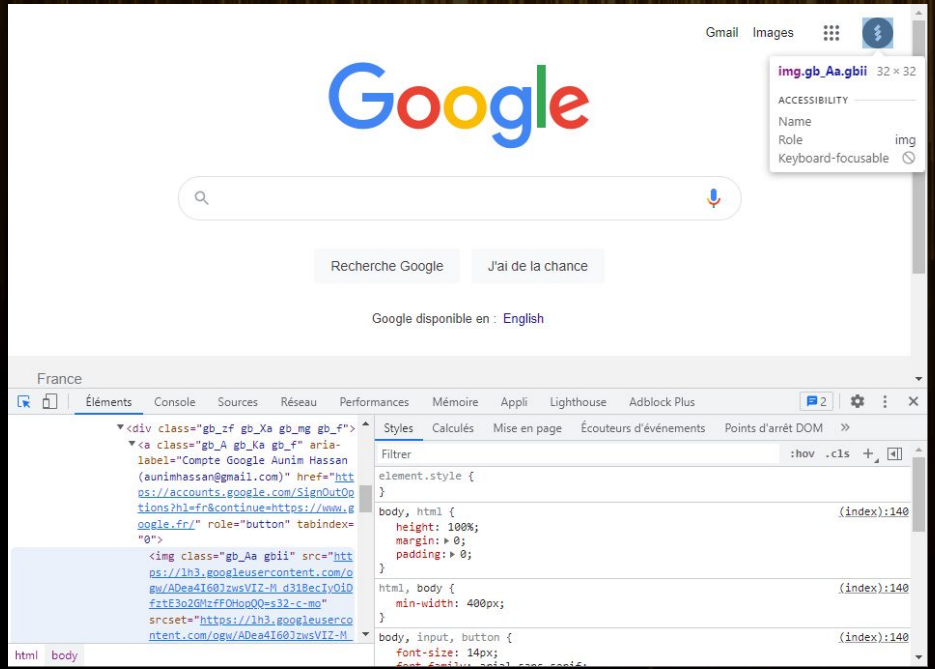
1. Créer et ouvrir le dossier js-course à l'aide de VS. Code.
2. Créer un dossier app.
3. Créer un dossier js dans le dossier app.
4. Créer le fichier main.js dans le dossier js.

LA CONSOLE DÉVELOPPEUR

https://developer.chrome.com/docs/devtools/

CHROME DEVTOOLS

UN ALLIÉ DE TAILLE



Ouverture avec la touche F12

Contexte d'exécution Javascript

Outil de débogage

Et beaucoup d'autres fonctionnalités…

https://developer.mozilla.org/fr/docs/Web/API/Console/log

HELLO WORLD!

QUELQUES MÉTHODES UTILES

JS

```js
// Afficher un élément dans la console
console.log('Hello World!');
console.warn('Hello World!');
console.error('Hello World!');

// Nettoyer la console
console.clear();

/**
 * Commentaire multilignes
 * --
 */
```

js/main.js

⚠

Veuillez utiliser les commentaires avec parcimonie, un code bien écrit doit parler de lui-même.

LES VARIABLES

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

DÉCLARER UNE VARIABLE

Les variables permettent de stocker des données en leur donnant un nom. Ces données peuvent ensuite être utilisées dans notre algorithme, nous verrons comment.

JS

```js
// Variable globale, à éviter...
var config = 'development';

// Variable, on peut réassigner sa valeur
let firstName = 'John';

// Constante, on ne peut pas réassigner sa valeur
const day = 'Monday';
```

js/main.js

https://wiki.c2.com/?LowerCamelCase

RÈGLES À RESPECTER

AVOIR UN CODE PROPRE

- ✗ Ne pas utiliser d'accent, ni de signe de ponctuation ou @.
- ✗ Les chiffres ne doivent pas être utilisés comme premier caractère.
- ✗ Ce qui se trouve à gauche du signe égal doit toujours être un nom de variable, et non une expression.

```
m + 1 = let b
```

- ✓ Respecter la convention lowerCamelCase.
- ✓ Privilégier l'écriture du code en anglais.

CONVENTIONS

JAVASCRIPT STYLE GUIDE

Standard JS
https://standardjs.com/

AirBnb
https://github.com/airbnb/javascript

Google
https://google.github.io/styleguide/jsguide.html

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

TYPES DE DONNÉES

--

NUMBER

```
let age = 20;
typeof age;
// returns number
```

NOMBRE ENTIER OU FLOTTANT

STRING

```
let firstName = 'John';
typeof firstName;
// returns string
```

CHAÎNE DE CARACTÈRES

BOOLEAN

```
let isOpen = false;
typeof isOpen;
// returns boolean
```

VRAI OU FAUX

UNDEFINED

```
let a;
typeof a;
// returns undefined
```

INDÉFINIE

NULL

```
let result = null;
typeof result;
// returns object
```

ABSENCE DE VALEUR / VIDE

OPÉRATIONS ENTRE LES NOMBRES

MANIPULER LES TYPES NUMÉRIQUES

#	EXEMPLES	RÉSULTATS
Addition	2 + 1	3
Soustraction	2 - 1	1
Multiplication	2 * 2	4
Division	17 / 4	4.25
Modulo	17 % 4	1
Exposant	5 ** 2	25

https://www.alsacreations.com/astuce/lire/1764-Les-template-strings-en-JavaScript.html

CONCATÉNATION

JUXTAPOSER DEUX CHAÎNES DE CARACTÈRES OU PLUS...

JS

```js
let firstName = 'John';
let lastName = 'Doe';

// Concaténation
let fullName = firstName + ' ' + lastName;

// ES6 : Template strings
fullName = `${firstName} ${lastName}`;
```

js/main.js

En JavaScript, l'opérateur "+" permet la concaténation de chaînes de caractères.

Avec l'arrivée d'ES6, sont apparus les template strings qui permettent une concaténation plus lisible à l'aide des backquotes [Alt Gr + 7].

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

MANIPULER DES CHAÎNES DE CARACTÈRES

QUELQUES MÉTHODES

```js
let greetings = 'Hello World!';

console.log(greetings.toUpperCase());
console.log(greetings.includes('Hello'));
console.log(greetings.split(' '));
```

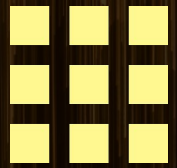
js/main.js

```
HELLO WORLD!
true
['Hello', 'World!']
```

SORTIE CONSOLE

https://www.youtube.com/watch?v=DHjqpvDnNGE

BONUS : FIRESHIP



LES TABLEAUX

LES TABLEAUX

STOCKER PLUSIEURS VALEURS DANS UNE MÊME VARIABLE

L'index d'un tableau est un nombre entier qui commence toujours à 0.

JS

```js
let fruits = ['banana', 'apple', 'orange', 'mandarin'];
console.log(fruits[2]);

// .isArray()
let age = 22;
console.log(Array.isArray(age));
console.log(Array.isArray(fruits));
```

js/main.js

```
orange

false
true
```

SORTIE CONSOLE

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

MANIPULER DES TABLEAUX

--

#	EXEMPLES	RÉSULTATS
Ajouter un élément	`[3, 26].push(14);`	`[3, 26, 14]`

Modifier un élément	`let numbers = [5, 14, 7, 6];`
	`numbers[2] = 21;`
	`console.log(numbers);`
		`[5, 14, 21, 6]`

Supprimer un élément	`let numbers = [5, 14, 7, 6];`
	`numbers.splice(2, 1);`
	`console.log(numbers);`
		`[5, 14, 6]`

Trier	`[3, 9, 7].sort();`	`[3, 7, 9]`
Taille	`["Bonjour", "Bonsoir", "Bye"].length;`	`3`
Concaténer	`[2, 1].concat([5, 7]);`	`[2, 1, 5, 7]`
Élément depuis l'index	`["Bonjour", "Bonsoir", "Bye"].at(1);`	`"Bonsoir"`

LES OBJETS

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

LES OBJETS

UN ENSEMBLE DE PAIRES CLEF / VALEUR

JS

```
let car = {
    brand: "Kia",
    model: "Sportage",
    color: "Black",
    year: 2010,
    start: () => console.log("Vrouuum!")
}

console.log(car.model);
console.log(car.year);
car.start();
```

js/main.js

```
Sportage
2010
Vrouuum!
```

SORTIE CONSOLE

En JavaScript, l'opérateur "." permet d'accéder aux propriétés et méthodes de l'objet.

Les objets sont très utiles pour représenter les éléments du monde réel. En outre, le format de données JSON [JavaScript Object Notation] se rapproche beaucoup de cette syntaxe.

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

MANIPULER DES OBJETS

--

```
let car = {
    brand: "Kia",
    model: "Rio",
    color: "Black",
    year: 2013,
    start: () => console.log("Vrouuum!")
};
```

#	EXEMPLES	RÉSULTATS
Tableau des clefs	`Object.keys(car);`	`['brand', 'model', 'color', 'year', 'start']`
Tableau des valeurs	`Object.values(car);`	`['Kia', 'Rio', 'black', 2013, 'Function: start']`
Ajouter une propriété	`car.doors = 3;`	--
Modifier une propriété	`car.model = "Sportage";`	--
Supprimer une propriété	`delete car.color;`	--

TRAVAUX PRATIQUES

ÉCHAUFFEMENT

TABLEAUX & OBJETS

Client 1
Nom : Doe
Prénom : John
Age : 21
Email : john.doe@xyz.com
Hobbies : Karaté, Tennis

Client 2
Nom : Stewart
Prénom : Jane
Age : 26
Email : [Email non renseigné]
Hobbies : Danse, Peinture, Chant

Client 3
Nom : Tardieu
Prénom : Olivier
Age : 32
Email : olivier.tardieu@xyz.com
Hobbies : [Hobbies non renseignés]

A l'aide de tableaux et d'objets, trouver le meilleur moyen de stocker toutes les données ci-contre dans une seule et unique variable.

CAS D'ENTREPRISE PART 1.

PARSING

L'entreprise Eofy a stocké uniquement les emails de ses salariés en respectant la nomenclature suivante :

stephane.lanier.a@xyz.com

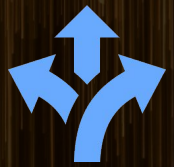
firstName lastName : departments

A l'aide de vos recherches, écrire un programme qui parsera la variable email afin d'obtenir chacune des informations séparément pour créer l'objet person ci-contre.

```
let email = 'stephane.lanier.a@xyz.com';
```

Votre programme

```
let person = {
    email: 'stephane.lanier.a@xyz.com',
    firstName: 'stephane',
    lastName: 'lanier',
    departments: ['A']
};
```



LES CONDITIONS

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

IF... ELSE IF... ELSE...

STRUCTURE CONDITIONNELLE

```
let person = {
    username: 'john.doe',
    bankAccount: 90,
};  state: 'poor'
};
```

L'instruction if exécute un bloc d'instruction si une condition donnée est vraie ou équivalente à vrai. Si la condition n'est pas vérifiée, il est possible d'utiliser une autre instruction.

Les conditions permettent d'ajouter une première forme d'intelligence à notre programme.

JS

```
if (person.bankAccount < 100) {
    person.state = 'poor';
} else if (person.bankAccount >= 100 && person.bankAccount < 200) {
    person.state = 'average';
} else {
    person.state = 'rich';
}
```

js/main.js

SWITCH

STRUCTURE CONDITIONNELLE

L'instruction switch évalue une expression et, selon le résultat obtenu et le cas associé, exécute les instructions correspondantes.

```
let fruit = 'Bananes';
// Bananes : 0.48 € le kilo.
```

JS

```
switch (fruit) {
    case "Oranges":
        console.log("Oranges : 0.59 € le kilo.");
        break;
    case "Pommes":
        console.log("Pommes : 0.32 € le kilo.");
        break;
    case "Bananes":
        console.log("Bananes : 0.48 € le kilo.");
        break;
    case "Cerises":
        console.log("Cerises : 3.00 € le kilo.");
        break;
    case "Mangues":
    case "Papayes":
        console.log("Mangues et papayes : 2.79 € le kilo.");
        break;
    default:
        console.log("Désolé, nous n'avons plus de " + fruit + ".");
}
```

js/main.js

OPÉRATEURS DE COMPARAISON

OPÉRATEURS	DESCRIPTIONS
==	Égal à
!=	Différent de
>	Strictement supérieur à
>=	Supérieur ou égal à
<	Strictement inférieur à
<=	Inférieur ou égal à

===	Égalité stricte	https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Operators/Strict_equality
!==	Inégalité stricte	https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Operators/Strict_inequality

OPÉRATEURS LOGIQUES

L'ALGÈBRE DE BOOLE

OPÉRATEURS	DESCRIPTIONS
!	NON Logique
&&	ET Logique
||	OU Logique

```
let age = 19;
let height = 1.68;
// Chouette, vous avez accès au manège.
```

JS

```
if (age >= 18 && height > 1.50)
    console.log('Chouette, vous avez accès au manège.');
else
    console.log('Accès au manège refusé.');
```

js/main.js

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Operators/Logical_AND

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Operators/Logical_OR

LE TYPAGE DYNAMIQUE FAIBLE

MIEUX COMPRENDRE L'ÉGALITÉ / L'INÉGALITÉ STRICTE

```js
// Typage dynamique
let email = 'john.doe@aeth.fr';

// Typage faible
console.log((2 == '2'));
console.log((2 === '2'));
```

js/main.js

```
true
false
```

SORTIE CONSOLE

JavaScript possède un typage dynamique, c'est à dire qu'il n'est pas nécessaire de préciser le type de la variable à sa déclaration.

De plus son typage est faible, ce qui peut être sujet à erreur comme on peut le voir dans l'exemple ci-contre.

```
console.log(2 + '2');
```

22

LES TERNAIRES

UNE SYNTAXE PLUS CONCISE TOUT EN RESTANT LISIBLE

```
let person = { username: 'john.doe' };
let bankAccount = 90;
```

JS

```
if (bankAccount < 100)
    person.state = 'poor';
else
    person.state = 'rich';

console.log(person.state);
```

js/main.js

JS

```
person.state = (bankAccount < 100) ? 'poor' : 'rich';

console.log(person.state);
```

js/main.js

```
poor
```

SORTIE CONSOLE

LES BOUCLES

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

LES BOUCLES WHILE / FOR

RÉPÉTER X FOIS DES INSTRUCTIONS

```js
// Boucle : While
let i = 0;
while (i < 5) {
    console.log(i)
    i++;
}
```

js/main.js

```js
// Boucle : For
for (let i = 0 ; i < 5 ; i++) {
    console.log(i);
}
```

js/main.js

```
0
1
2
3
4
```

SORTIE CONSOLE

Les boucles permettent de répéter des actions simplement et rapidement.

Une boucle peut être vue comme une version informatique de « copier N lignes » ou de « faire X fois quelque chose ».

L'INSTRUCTION BREAK

SORTIE PRÉMATURÉE D'UNE BOUCLE OU D'UNE CONDITION SWITCH

```js
// Boucle : For
for (let i = 0 ; i < 100 ; i++) {
    if (i > 5) {
        console.log("Stop");
        break;
    }
    console.log(i);
}
```

js/main.js

```
0
1
2
3
4
5
Stop
```

SORTIE CONSOLE

L'instruction break peut s'avérer utile lorsqu'on souhaite sortir prématurément d'une boucle ou d'une condition switch.

Elle ne fonctionne pas dans une condition if/else.

L'INSTRUCTION CONTINUE

SORTIE DE L'ITÉRATION EN COURS

```js
// Boucle : For
for (let i = 0 ; i < 5 ; i++) {
    if (i == 3) {
        // Saut de la valeur 3
        console.log("-");
        continue;
    }
    console.log(i);
}
```

js/main.js

```
0
1
2
-
4
```

SORTIE CONSOLE

L'instruction continue arrête l'exécution des instructions pour l'itération de la boucle. L'exécution est reprise à l'itération suivante.

LA MÉTHODE .forEach

PARCOURIR UN TABLEAU FACILEMENT

```js
let users = ['John', 'Bob', 'Aunim'];

// .forEach
users.forEach((user) => {
    console.log(user);
});

// .forEach, avec l'indice d'itération
users.forEach((user, key) => {
    console.log(key, user);
});
```

js/main.js

```
John
Bob
Aunim

0 John
1 Bob
2 Aunim
```

SORTIE CONSOLE

À travers cet exemple, nous remarquons une syntaxe assez particulière propre à JavaScript.

Pas de panique, le prochain chapitre sur les fonctions permettra de mieux comprendre de quoi il s'agit.

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

LA BOUCLE FOR... OF...

LA MEILLEUR SOLUTION POUR PARCOURIR UN TABLEAU

```js
let users = ['John', 'Bob', 'Aunim'];

// for... of...
for (const user of users) {
    console.log(user);
}

// for... of..., avec l'indice du tableau
for (const [key, user] of users.entries()) {
    console.log(key, user);
}
```

js/main.js

```
John
Bob
Aunim

0 John
1 Bob
2 Aunim
```

SORTIE CONSOLE

Afin de parcourir un tableau, je vous recommande plutôt cette méthode qui permet d'éviter la syntaxe un peu particulière vue précédemment.

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

LA BOUCLE FOR... IN

PARCOURIR UN OBJET FACILEMENT

```
let car = {
    brand: "Kia",
    model: "Rio",
    color: "Black"
};
```

JS

```
for (const property in car) {
    console.log(`${property}: ${car[property]}`);
}
```

js/main.js

```
brand: Kia
model: Rio
color: Black
```

SORTIE CONSOLE

LES FONCTIONS

LES FONCTIONS

BIEN COMPRENDRE LA DIFFÉRENCE ENTRE LA DÉFINITION ET L'APPEL D'UNE FONCTION

JS

```
// Définition
function addition() {
    // ...
}

// Première appel
addition();

// Second appel
addition();
```

js/main.js

L'utilisation d'une fonction permet d'implémenter un comportement une seule fois puis de le réutiliser autant de fois que nécessaire. Cela permet d'organiser son code et en faciliter sa maintenance.

Une fonction est déclarée via le mot-clef function et est suivi de son nom.

La définition de la fonction ne sera pas exécutée, c'est lors de l'appel à cette fonction que celui-ci réalisera son exécution. L'appel se fera via le nom de votre fonction.

L'INSTRUCTION RETURN

RETOURNER UNE VALEUR

```js
// Définition
function addition() {
    let result = 2 + 2;
    return result;
}

// Appel / Affichage
console.log(addition());
```

js/main.js

```
4
```

SORTIE CONSOLE

Suite au traitement réalisé par une fonction, il est possible d'en récupérer le résultat en dehors de celle-ci. Ce résultat est la valeur de retour de la fonction.

⚠ Attention si la fonction rencontre l'instruction return, elle retournera la valeur et s'arrêtera.

```js
function addition() {
    let result = 2 + 2;
    return result;

    // Ne sera jamais exécuté
    console.log('Coucou !');
}
```

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

PARAMÈTRES / ARGUMENTS

DONNÉES EN ENTRÉE D'UNE FONCTION

```js
// Définition - (paramètres)
function addition(a, b) {
    let result = a + b;
    return result;
}

// Appel / Affichage - (arguments)
console.log(addition(2, 3));
```

js/main.js

```
5
```

SORTIE CONSOLE

Il est possible de faire passer des paramètres à votre fonction afin de la rendre plus générique.

Il faudra également donner à votre fonction des valeurs pour chaque paramètre lors de son appel.

TRAVAUX PRATIQUES

LA COULEUR DU FRUIT

```
let fruits = [
    ['banana', 'yellow'],
    ['apple', 'green'],
    ['orange', 'orange'],
    ['strawberry', 'red']
];

let fruitName = 'apple';

/**
 * Écrire une fonction qui retourne la couleur du fruit
 * --
 *
 */
function getFruitColor(fruits, fruitName) {
    // Votre code...
}

console.log(getFruitColor(fruits, fruitName)); // green
```

https://www.lipsum.com/

```
let text = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.';

/**
 * Écrire une fonction qui supprime les voyelles d'un texte.
 * --
 *
 */
function removeVowels(text) {
    // Votre code
}

console.log(removeVowels(text));
```

SUPPRIMER LES VOYELLES

CAS D'ENTREPRISE PART 2.

PARSING

L'entreprise Eofy a stocké uniquement les emails de ses salariés en respectant la nomenclature suivante :

stephane.lanier.a_e@xyz.com

firstName lastName departments

A l'aide de vos recherches, écrire une fonction qui parsera la variable emails en retournant le tableau d'objets persons ci-contre.

```
let emails = ['stephane.lanier.a_e@xyz.com', 'john.doe.b_c@yahoo.fr'];
```

parseEmails(...)

```
let persons = [{
    email: 'stephane.lanier.a_e@xyz.com',
    firstName: 'stephane',
    lastName: 'lanier',
    departments: ['A', 'E']
}, {
    // Informations de John...
}];
```

https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage

LE CHIFFRE DE CÉSAR

CRYPTOGRAPHIE

ABCDEFGHIJKLMNOPQRSTUVWXYZ

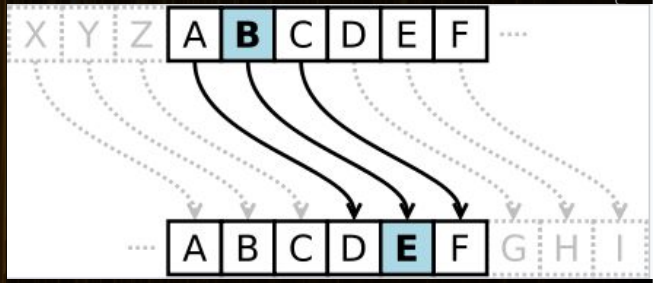
caesarCipher(...)

DEFGHIJKLMNOPQRSTUVWXYZABC

EXEMPLE POUR UN DÉCALAGE DE 3

Écrire la fonction qui permet de réaliser le chiffre de César avec un décalage de 1 dans un premier temps.

Dans un second temps, ajouter le décalage en paramètre de la fonction.



Le chiffre de César fonctionne par décalage des lettres de l'alphabet.

Par exemple, dans l'image ci-dessus, il y a une distance de 3 caractères, donc B devient E dans le texte codé.

Exemple de texte à chiffrer

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth.

ALLER PLUS LOIN

ALLER PLUS LOIN

CONTINUER D'APPRENDRE

- Manipuler le DOM
- Le format de données JSON
- La POO en JavaScript
- L'asynchrone en JavaScript

MERCI DE VOTRE ATTENTION