YASMINE BOUAMRA

Contact : yasmine.trainings@gmail.com.

01 INITIATION À JAVASCRIPT

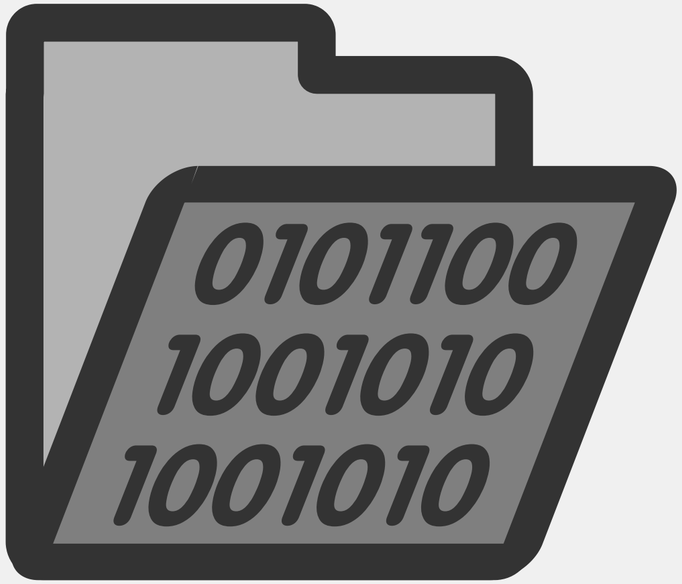
INTRODUCTION À L'HISTOIRE DE JAVASCRIPT

- 02

Javascript est un langage de programmation de scripts. Il est utilisé pour rendre les pages web *dynamiques / interactives.*

Javascript est un langage orienté objet à prototype. L’orienté objet est un paradigme de programmation sur lequel on s'attardera plus tard.

MAIS AVANT, QU'EST-CE QU'UN LANGAGE DE PROGRAMMATION ?



UN OUTIL DE RÉSOLUTION DE PROBLÈMES

C’est la manière que nous, humains, utilisons pour traduire des instructions à un ordinateur.

JavaScript est un langage interprété.

Il existe deux types de langage de programmation.
Les langages de programmations compilés et les langages de programmations interprétés.

INTERPRÉTÉ VS COMPILÉ

Dans le cas d'un langage interprété, toutes les commandes sont analysées et exécutées une à une.
Dans le cas langage compilé, le code est analysé et traduit en code source qui sera exécuté par l'OS directement.

06

DU COUP, OÙ TROUVER L'INTERPRÉTEUR DE JAVASCRIPT?

Les navigateurs (Google Chrome, Mozilla, Safari etc...) ont un interpréteur de JS intégré.

02

COMMENCER À CODER EN JS

Les outils et notions de base

Les outils pour coder en JavaScript

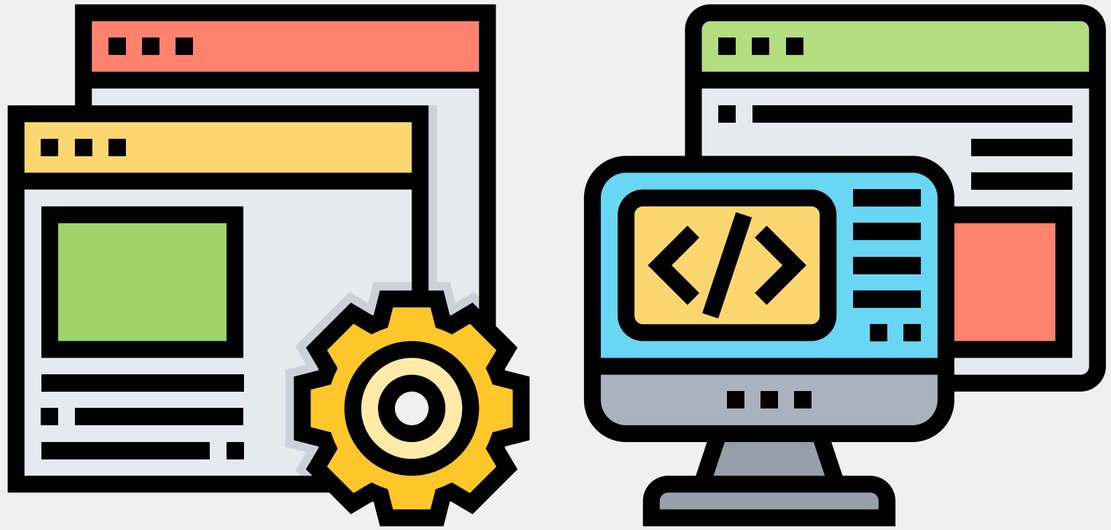
Vous aurez besoin de deux outils :
- Un éditeur de texte
- Un navigateur web

ÉDITEUR DE TEXTE

NAVIGATEUR WEB

08

LE DÉVELOPPEMENT WEB FRONT ET BACK END



Le front end c’est l’interface avec laquelle l’utilisateur interagit.

Le back end est la partie qui gère les données, interagit avec le serveur et produit des résultats.

Javascript est l’un des trois outils principaux du développement web frontal aux côtés d’HTML et CSS.

- HTML sert de structure à une page web
- CSS permet de gérer l’aspect et le design d’une page web
- JS permet de gérer le comportement des composants d’une page web

Il est temps de découvrir les principes de programmation de base et leur syntaxe en JS.

03

A VOS CLAVIERS !

Affichages

Pour afficher des valeurs et des résultats, on peut les afficher au niveau de la console. Pour cela, on utilise console.log()

```
<script>
    console.log("Hello World!");
</script>
```

Commentaires

Un commentaire est une ligne de code qui est ignorée par l'interpréteur.

```
<script>
    //  ceci est un commentaire

    /* ceci est églament
    un commentaire */
</script>
```

VARIABLES

Les variables sont utilisées pour manipuler des données
Il existe 5types de variables :

- number (nombres entiers ou flottants)
- boolean (booléens)
- string (chaîne de caractères)
- undefined (variable indéfinie)
- null (variable définie, mais sans valeur)

VARIABLES

Les variables sont typés dynamiquement par JavaScript.
Pour déclarer une variable, il faut utiliser le mot clé var.

```
<script>
    var a = 0;
    var b = "bonjour";
    var c = true;
</script>
```

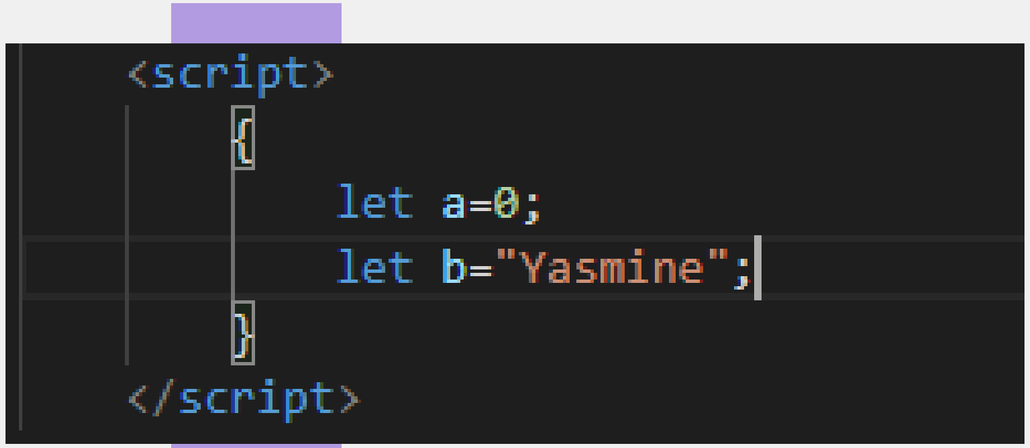
VARIABLES

Il est possible de voir le type d'une variable en utilisant la fonction typeof(). Pour l'afficher, on utilise console.log().

```
<script>
    var a = 0;
    console.log(typeof(a))
</script>
```

VARIABLES

Il existe une deuxième manière de déclarer les variables, qui est : let.
let permet de déclarer des variables dont la portée réduite. Elle est limitée à un bloc d'instructions. Un bloc d'instructions est défini par deux acolades.
Par exemple, on utilise des blocs d'instructions pour les branchements conditionnels.



NUMBER

Le type Number permet de stocker des nombres allant jusqu'à 2^53.
Il permet de stocker les nombres naturels et flottants.
Il possède trois signes :

- +Infinity
- -Infinity
- NaN

NUMBER

```
let a = 1;
let b = 1.75;
let c = +Infinity;
let d = -Infinity;
let e = NaN;
```

EXERCICE

Affichez dans votre console l'addition de deux nombres entiers.
PS : l'addition se fait avec l'opérateur +.

BOOLEAN

Le type boolean permet de donner une valeur de vérité à une variable.
Il a deux symboles spéciaux :

- True
- False

BOOLEAN

```
let x = true;
let y = false;
```

STRING

Le type string permet de stocker des chaines de caractères. En d'autres termes, du texte.

- Concaténation à l’aide de l’opérateur +.
- Longueur de la chaîne à l’aide de la propriété length.
- Récupération d’un caractère à l’aide de [n] ou charAt().
- Caractère d’échappement : \

String


```
let test = "true";
let x = 'jazz';
```

EXERCICE

Quelles sont les types des variables suivantes ?

```
let a = "Javascript";
let b = "true";
let c = true;
let d = 100;
```

EXERCICE

Que valent les valeurs suivantes ?

```
let a = "Javascript";
let b = "true";
let c = a + b;
let d = 100;
let e = a + d;
```

EXERCICE

Affichez les chaines de caractères et leur longueur sur la console.

Affichez le caractère à l'indice 5 pour chaque chaine. Que remarquez-vous ?

```
<script>
    {
        var a="Bonjour";
        var b="Je m'appelle";
        var c= "Yasmine";
    }
</script>
```

UNDEFINED & NULL

En plus des nombres, des types primitifs vu auparavant, nous avons undefined et null.

- undefined est le type qui est attribué implicitement à une variable déclarée sans valeur.
- null est le type que l'on attribue explicitement à une variable lorsque l'on souhaite la définir sans valeur.
Typiquement, null est utilisée pour la gestion d'erreurs.

UNDEFINED & NULL

En plus des nombres, des types primitifs vu auparavant, nous avons undefined et null.

```
<script>
    {
        var a; // type undefined
        var b=null; // type null
    }
</script>
```

CONSTANTES

Les constants sont des variables déclarées en utilisant le mot clé **CONST**, comme leur noms l'indique, la valeur d'une constante ne peut pas être modifié après leur avoir attribué explicitement une valeur.

```
<script>
    const pi = 3.14;
    const avogadro  = 6.02214
</script>
```

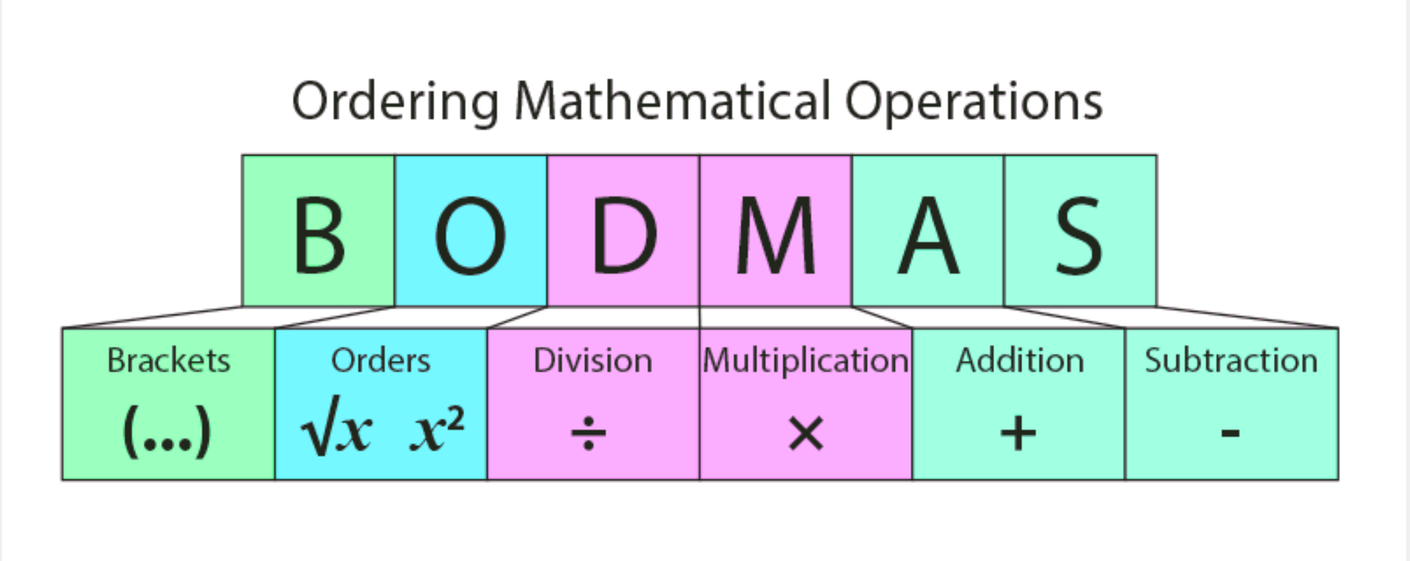
EXERCICE

Transcrivez les informations suivantes à l'aide de variable ou de constantes.

```
1   Le système solaire est consituté de 9 planètes.
2   L'hydrogène a un électron.
3   En ce moment, il y a 12 personnes dans la salle d'attentes.
4   Au jour d'aujourd'hui, il y a 1420 satellites starling opérationnels.
```

OPÉRATEURS ARITHMÉTIQUES

Passons à présent aux opérateurs arithmétiques.
le plus +, le moins -, la multiplication * et la division /.
l'ordre de priorités est :
* et / d'abord, puis + et le - , de gauche a droite, les parenthèses les plus internes en premier.



EXERCICE

Donner le résultat des opérations suivantes

```
<script>
    const pi = 3.14;
    var r = 4;
    let res = pi * r * r - 5;
    // res = ?
    const res2 = 5 * 4 - 2 * (12 - (7 * 2));
    // res2 = ?
</script>
```

LES CONVERSIONS IMPLICITE

La conversion implicite se fait lors de l'utilisation d'opérateurs arithmétiques entre variables de types différents.

- La concaténation d'une chaine de caractères avec un nombre donne une chaine de caractères
- Dans le cas de nombres stockés sous forme de chaines de caractères, ils sont traités comme nombre avec les autres opérateurs arithmétiques.

```
<script>
    {
        x="11"+2; // 112
        x="11"-2; // 9
    }
</script>
```

LES OPÉRATEURS LOGIQUES

Les opérateurs logiques nous permettent de faire des comparaisons entre des variables et avoir un résultat sous forme d'un boolean.

&& ET	! NON	> SUPERIEUR
|| OU	< INFERIEUR	!= INEGALITÉ

== VS ===

== et === sert tous les deux à étudier l'égalité entre deux opérandes. Néanmoins, ils ne sont pas équivalent.
== est utilisée pour l'égalité des valeurs, et === est utilisé pour légalité des valeurs et du type.

x	y	==	===
undefined	undefined	true	true
null	null	true	true
true	true	true	true
false	false	true	true
'foo'	'foo'	true	true
0	0	true	true
+0	-0	true	true
+0	0	true	true
-0	0	true	true
0	false	true	false
""	false	true	false
""	0	true	false
'0'	0	true	false
'17'	17	true	false
[1, 2]	'1,2'	true	false
new String('foo')	'foo'	true	false
null	undefined	true	false

EXERCICE

Donner le résultat des opérations suivantes

```
<script>
    var a = true && false;
    let b = false || true;
    var z = 0 && false;
    var x = 10 && true;
    const cond = 5 === 6;
    let cond2 = 5 == '5';
</script>
```

Passons aux structures de contrôles. Les structures qui permettent de répéter des instructions, gérer des conditions, etc...

04 LES STRUCTURES DE CONTROLE

LES CONDITIONS

Les conditions sont des instructions qui nous permettent d’exécuter un bout de code seulement et seulement si la condition est vérifiée, ou d'exécuter un autre bout de code dans le cas contraire.

```
<script>
    if (condition) { // la condition doit avoir une valeu booléene
        //intructions
    }
    else { // le else est optionnel
        //instructions
    }
</script>
```

EXERCICE

Donner les scripts qui permettent :
- d'afficher "possibilité d'avoir le permis de conduire" si la variable âge est supérieure ou égale à 18.
- d'afficher le signe de la variable x (négatif ou positif)

fg

LE SWITCH CASE

Pour éviter d'imbriquer beaucoup de conditions, on peut utiliser le switch case

```
switch (expression) {
    case x:
        // instructions
        break;
    case y:
        // instructions
        break;
    default:
    // instructions

}
```

EXERCICE

Donner les scripts qui permettent :
- d'afficher le mois si son nombre est donné

LES BOUCLES

Avant de définir les boucles, imaginons qu'on ait besoin d'afficher tous les nombres entre 1 et 100, ou que vous vouliez exécuter des instructions similaires n fois.
Comment allez-vous procéder ?

LES BOUCLES

Les boucles remédient à cette problématique.
Une boucle vous permet d'exécuter des instruction n fois, sans pour autant les re-écrire plusieurs fois.
Pour le moment, on va voir uniquement deux types de boucle, for et while. Plus tard nous verons aussi la boucle foreach.

LA BOUCLE FOR

```
<script>
    for (let index = 0; condition; index++) {
        //initialisation de l'index
        // la condition d'aret( par rapport a l'index)
        // l'incrementation ou decremntation de l'index


        //instructions
    }
</script>
```

LA BOUCLE WHILE

```
<script>
    while (conditions) {
        //boucle tantque la condition est vrai


        //instructions
    }
</script>
```

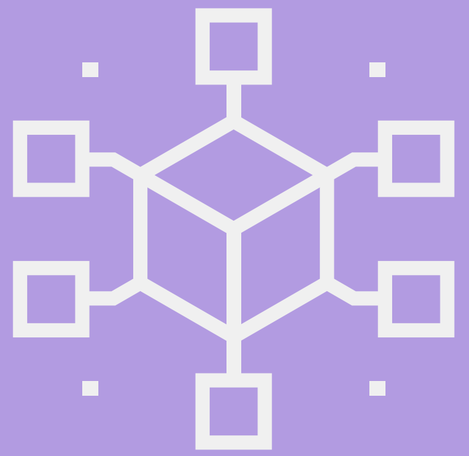
EXERCICE

Donner le script qui font le traitement suivant:

- affiche "itération x" sur la console, avec x allant de 0 à 99.
- affiche tous les nombres premiers existanst entre 0 et 400.
- affiche les nombres dans un ordre décroissant (120, 119.... 3, 2, 1, 0)

LES STRUCTURES DE DONNÉES

Une structure de données, comme son nom l'indique, est une structure qui nous permet de sauvegarder des données d'une manière un peu plus complexe et pratique qu'avec des variables. Un exemple d'une structure qui est largement utilisée est les vecteurs.
Javascript dispose de plusieurs structures de données prédéfinies, néanmoins, nous pouvons nous aussi définir des structures de données adaptées aux problématiques qu'on aborde.



LES VECTEURS

Imaginons que l'on veuille modéliser une liste de voitures. Une méthode est de créer n variables, n étant le nombre de voitures, mais cette manière de faire n'est pas pratique.
Les vecteurs remédient a cette problématique.

```
<script>
    var vec = ['foo', 'bar', 'foobar'];
    //vec[index] retourne l'element index + 1 du vecteur
    console.log(vec[0]); // foo
    console.log(vec[1]); // bar
    console.log(vec[2]); // foobar
</script>
```

EXERCICE

Donner les scripts qui permettent :
- d'afficher le contenu du vecteur donné comme example d'une manière dynamique.
- élaborer une structure qui est composée de vecteur imbriqués.

LES MATRICES

Comme vous avez pu le voir, on peut avoir un vecteur de vecteurs. Dans ce cas il n'est plus question de vecteur, mais plutôt de matrice.

```
<script>
    var mat = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ];
    //mat[l][c] l designe la ligne - 1 , et c la column - 1
    console.log(mat[0][0]); // 1
    console.log(mat[1][1]); // 5
    console.log(mat[2][0]); // 7
</script>
```

QUELQUES ATTRIBUTS UTILES

```
<script>
    var mat = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];
    var vec = ['foo', 'bar', 'foobar'];
    console.log(mat.length); // 9
    console.log(vec.length); // 3
    // parcourire un vecteur/ une matrice
    vec.forEach(v => {
        console.log(v);
    });
</script>
```

LES FONCTIONS

Une fonction est un ensemble d'instructons enveloppés sous le même nom, qu'on peut réutiliser sans pour autant les réécrire. Il suffit juste d'appeler la fonction en lui passant les parametres necessaires.
Un paramètre est une varibale qui est "introduite" dans une fonction.

```
<script>
    function addition(a, b) {
        return a + b; // permet de retourner les resultats.
    }
    function afficher(x) {
        console.log(x);
    }
</script>
```

EXERCICE

Coder les fonctions qui :

- concatènent le contenue d'un tableau et l'affiche sur la console
- retourne la surface d'un rond.

05

ORIENTÉ OBJET À PROTOTYPE

L'orienté objet en JavaScript.

ORIENTÉ OBJET

Nous l'avons dit le premier jour, JavaScript est un langage orientée objet à prototypes.
Qu'est-ce que l'orienté objet ? Qu'est-ce que l'orienté objet à prototypes ? Quelle est la différence ?

LES OBJETS

Les objets sont une sorte de contenères. Ils permettent de grouper des caractéristiques et des comportements. Ils sont utilisés pour décrire des éléments.
Par exemple,
Un compte utilisateur est un élément qui aura un nom, prénom, identifiants, etc... et des comportements : se connecter, se déconnecter, ajouter une photo, changer une photo...
Les caractéristiques sont des attributs (des variables).
Les comportements sont des méthodes (des fonctions).

ORIENTÉ OBJET À PROTOTYPE

L'orienté objet à prototype définit des objets directement. L'objet sert de prototype pour la création d'autres objets (par duplication) et pour l'héritage (ajout dynamique de propriétés).

L'orienté objet classique passe par la création d'une classe. Il offre plus d'abstraction. Les objets sont des instances de classe.
L'héritage passe par la création d'une classe fille.
Les relations se basent sur un modèle parent-child.

DÉCLARATION

```
<script>
    // déclaration en utilisant new objet
    var objet = new Object; // création de l'objet
    objet.cara = "je suis une caractéristique"; // définition d'un attribut
    objet.metho = methode; // définition d'une méthode

    function methode() // déclaration de la méthode
    {
        console.log("bonjour je suis une méthode");
    }

    objet.metho(); // appel de la méthode
</script>
```

EXERCICE

Définissez un objet Personne.
Donnez à cet objet les attributs nom, prénom, âge, métier et passions.
Remplissez avec vos informations.

DÉLCARATION

```
<script>
    // déclaration d'un objet en utilisant un inialisateur
    var objet = {
        cara: "je suis une caractéristique", // définition d'un attribut
        metho : methode, // définiton de la méthode
    }
    function methode() // déclaration de la méthode
    {
        console.log("bonjour je suis une méthode");
    }
    objet.metho(); // appel de méthode
</script>
```

EXERCICE

Redéfinissez l'objet Personne en utilisant
un initialisateur.
Nous souhaitons définir plusieurs objets
Personne aux valeurs différentes.
Comment procéder ?

DÉCLARATION

```
<script>
    // déclaration d'un objet en utilisant un constructeur
    var objet = new Object("hello")
    function Object(cara)
    {
        this.cara=cara;
        this.metho=methode;
    }
    function methode() // déclaration de la méthode
    {
        console.log("bonjour je suis une méthode");
    }

    objet.metho(); // appel de méthode
</script>
```

EXERCICE

Un animal est défini par son nom, sa race, son âge, sa taille, son poids, son régime alimentaire et son habitat naturel et son emplacement.
Un emplacement est défini par deux points géographiques x et y.
Un animal peut se déplacer d'un emplacement à un autre.
Un animal peut changer de poids. Il peut vieillir.

EXERCICE

1) Définissez un animal.
2) Créez un constructeur pour animal.
3) Si l'âge de l'animal a moins de 15 ans, il se déplace vers l'emplacement x=10 et y=-3.
4) Si l'âge de l'animal est supérieur à 15 ans, il se déplace vers l'emplacement x=-1 et y=-2.
5) Affichez l'emplacement de l'animal.
6) Comment savoir quelle distance l'animal a parcouru ?

EXERCICE

En utilisant deux syntaxes différentes, créez un tableau d'animaux. (Indice : utilisez Array)
Comment parcourir la liste ? Ecrire le script.

DES QUESTIONS ? COMMENTAIRES ?

N'hésitez pas à partager vos commentaires.

ADRESSE E-MAIL

yasmine.trainings@gmail.com