JavaScript

Master **IMSD**
IFRISSE

PIERRE CLAVER OUEDRAOGO

Tables de matières

Aperçu du cours

Objectifs du cours

Ce cours aborde l'ensemble des fonctionnalités de base et utiles du JavaScript. L'objectif principal du cours n'est pas de faire de vous des experts du JavaScript mais de compléter vos connaissances en développement web.

Ce cours va ainsi vous permettre d'écrire , de comprendre du code JavaScript, et de savoir quand et comment utiliser du JavaScript.

En somme , l'idée est de vous permettre de pouvoir résoudre par vous-même des problématiques liées au JavaScript.

Prérequis

Pour suivre ce cours dans de bonnes conditions, il est essentiel que vous possédiez des bases en HTML et en CSS. Ce qui est le cas puisque vous avez déjà eu le cours d'HTML et CSS.

Unité 0 : Introduction au JavaScript

Le JavaScript est un langage de programmation créé en 1995. Le JavaScript est aujourd'hui l'un des langages de programmation les plus populaires et il fait partie des langages web dits « standards » avec le HTML et le CSS.

On dit que le HTML, le CSS et le JavaScript sont des standards du web car les principaux navigateurs web (Google Chrome, Safari, Firefox, etc.) savent tous « lire » (ou « comprendre » ou « interpréter ») ces langages et les interprètent généralement de la même façon ce qui signifie qu'un même code va généralement produire le même résultat dans chaque navigateur.

Pour définir ce qu'est le JavaScript et le situer par rapport aux autres langages, et donc pour comprendre les intérêts et usages du JavaScript il faut savoir que :

- Le JavaScript est un langage dynamique ;
- Le JavaScript est un langage (principalement) côté client ;
- Le JavaScript est un langage interprété ;
- Le JavaScript est un langage orienté objet.

Unité 0 : Introduction au JavaScript

Le JavaScript est un langage dynamique

Le JavaScript est un langage dynamique, c'est-à-dire un langage qui va nous permettre de générer du contenu dynamique pour nos pages web.

Un contenu « dynamique » est un contenu qui va se mettre à jour dynamiquement, c'est-à-dire changer sans qu'on ait besoin de modifier le code manuellement mais plutôt en fonction de différents facteurs externes.

Pour rappel nous avons vu que le HTML et le CSS forment un premier couple très puissant. Cependant, nous allons être limités si nous n'utilisons que ces deux langages tout simplement car ce sont des langages qui ne permettent que de créer des pages « statiques ».

Une page statique est une page dont le contenu est le même pour tout le monde, à tout moment.

Unité 0 : Introduction au JavaScript

Le JavaScript est un langage dynamique

En effet ni le HTML ni le CSS ne nous permettent de créer des contenus qui vont se mettre à jour par eux-mêmes. Le CSS, avec les animations, nous permet de créer des styles pseudo-dynamiques mais tout de même prédéfinis.

C'est là où le JavaScript entre en jeu : ce langage va nous permettre de manipuler des contenus HTML ou des styles CSS et de les modifier en fonction de divers **évènements** ou **variables**. Un évènement peut être par exemple un clic d'un utilisateur à un certain endroit de la page tandis qu'une variable peut être l'heure de la journée.

Unité 0 : Introduction au JavaScript

Le JavaScript est un langage (principalement) côté client

Un langage « côté client » est un langage qui va être exécuté dans le navigateur des utilisateurs qui demandent la page. Tout comme le HTML, le CSS, le JavaScript est un langage côté client.

La chose importante à retenir ici est que le JavaScript est un langage principalement utilisé côté client, mais qui va également pouvoir s'utiliser côté serveur à condition qu'on mette en place un environnement favorable (en utilisant Node.js par exemple).

Le JavaScript, un langage interprété

Le JavaScript est un langage interprété. Cela signifie qu'il va pouvoir être exécuté directement sous réserve qu'on possède le logiciel interpréteur. Pas de panique ici : tous les navigateurs connus possèdent leur interpréteur JavaScript.

Unité 0 : Introduction au JavaScript

Le JavaScript, un langage orienté objet

La programmation orientée objet est une façon de concevoir un code autour du concept d'objets. Un objet est une entité qui peut être vue comme indépendante et qui va contenir un ensemble de variables (qu'on va appeler propriétés) et de fonctions (qu'on appellera méthodes). Ces objets vont pouvoir interagir entre eux.

JavaScript vs Java : attention aux confusions !

Ces deux langages, bien que syntaxiquement assez proches à la base, reposent sur des concepts fondamentaux complètement différents et servent à effectuer des tâches totalement différentes.

Pourquoi des noms aussi proches ? Java est une technologie créée originellement par Sun Microsystems tandis que JavaScript est un langage créé par la société Netscape.

Unité 0 : Introduction au JavaScript

Avant sa sortie officielle, le nom original du JavaScript était « **LiveScript** ». Quelques jours avant la sortie du **LiveScript**, le langage est renommé JavaScript.

A l'époque, Sun et Netscape étaient partenaires et le Java était de plus en plus populaire. Il est donc communément admis que le nom « JavaScript » a été choisi pour des raisons marketing et pour créer une association dans la tête des gens avec le Java afin que les deux langages se servent mutuellement.

Le créateur du JavaScript a également expliqué que l'idée de base derrière le développement du JavaScript était d'en faire un langage complémentaire au Java.

Unité 0 : Introduction au JavaScript

Avant sa sortie officielle, le nom original du JavaScript était « **LiveScript** ». Quelques jours avant la sortie du **LiveScript**, le langage est renommé JavaScript.

A l'époque, Sun et Netscape étaient partenaires et le Java était de plus en plus populaire. Il est donc communément admis que le nom « JavaScript » a été choisi pour des raisons marketing et pour créer une association dans la tête des gens avec le Java afin que les deux langages se servent mutuellement.

Le créateur du JavaScript a également expliqué que l'idée de base derrière le développement du JavaScript était d'en faire un langage complémentaire au Java.

Environnement de travail

Pour coder en JavaScript, nous n'allons avoir besoin que d'un éditeur de texte. Il existe de nombreux éditeurs de texte sur le web et la majorité d'entre eux sont gratuits.

Unité 0 : Introduction au JavaScript

Avant sa sortie officielle, le nom original du JavaScript était « **LiveScript** ». Quelques jours avant la sortie du **LiveScript**, le langage est renommé JavaScript.

A l'époque, Sun et Netscape étaient partenaires et le Java était de plus en plus populaire. Il est donc communément admis que le nom « JavaScript » a été choisi pour des raisons marketing et pour créer une association dans la tête des gens avec le Java afin que les deux langages se servent mutuellement.

Le créateur du JavaScript a également expliqué que l'idée de base derrière le développement du JavaScript était d'en faire un langage complémentaire au Java.

Environnement de travail

Pour coder en JavaScript, nous n'allons avoir besoin pour ce cours que d'un éditeur de code comme en HTLM et CSS.

Unité 0 : Introduction au JavaScript

Où écrire le code JavaScript ?

On va pouvoir placer du code JavaScript à trois endroits différents :

- Directement dans la balise ouvrante d'un élément HTML ;
- Dans un élément script, au sein d'une page HTML ;
- Dans un fichier séparé contenant exclusivement du JavaScript et portant l'extension .js

Placer le code JavaScript dans la balise ouvrante d'un élément HTML

Il est possible que vous rencontriez encore aujourd'hui du code JavaScript placé directement dans la balise ouvrante d'éléments HTML.

Ce type de construction était fréquent à l'époque notamment pour prendre en charge des évènements comme par exemple un clic.

Unité 0 : Introduction au JavaScript

Où écrire le code JavaScript ?

On va pouvoir placer du code JavaScript à trois endroits différents :

- Directement dans la balise ouvrante d'un élément HTML ;
- Dans un élément script, au sein d'une page HTML ;
- Dans un fichier séparé contenant exclusivement du JavaScript et portant l'extension .js

Placer le code JavaScript dans la balise ouvrante d'un élément HTML

Il est possible que vous rencontriez encore aujourd'hui du code JavaScript placé directement dans la balise ouvrante d'éléments HTML.
Ce type de construction était fréquent à l'époque notamment pour prendre en charge des évènements comme par exemple un clic.

Unité 0 : Introduction au JavaScript

il est généralement déconseillé et considéré comme une mauvaise pratique d'écrire du code JavaScript dans des balises ouvrantes d'éléments HTML.

Placer le code JavaScript dans un élément script, au sein d'une page HTML

On va également pouvoir placer notre code JavaScript dans un élément script qui est l'élément utilisé pour indiquer qu'on code en JavaScript. On va pouvoir placer notre élément script n'importe où dans notre page HTML, aussi bien dans l'élément head qu'au sein de l'élément body.

De plus, on va pouvoir indiquer plusieurs éléments script dans une page HTML pour placer plusieurs bouts de code JavaScript à différents endroits de la page.

Cette méthode est meilleure que la précédente mais pas recommandé.

Unité 0 : Introduction au JavaScript

il est généralement déconseillé et considéré comme une mauvaise pratique d'écrire du code JavaScript dans des balises ouvrantes d'éléments HTML.

Placer le code JavaScript dans un élément script, au sein d'une page HTML

On va également pouvoir placer notre code JavaScript dans un élément **script** qui est l'élément utilisé pour indiquer qu'on code en JavaScript. On va pouvoir placer notre élément **script** n'importe où dans notre page HTML, aussi bien dans l'élément head qu'au sein de l'élément body.

De plus, on va pouvoir indiquer plusieurs éléments script dans une page HTML pour placer plusieurs bouts de code JavaScript à différents endroits de la page.

Cette méthode est meilleure que la précédente mais pas recommandé.

Unité 0 : Introduction au JavaScript

Placer le code JavaScript dans un fichier séparé

Placer le code JavaScript dans un fichier séparé ne contenant que du code JavaScript est la méthode recommandée et que nous préférerons tant que possible. Pour faire cela, nous allons devoir créer un **nouveau fichier** et l'enregistrer avec une **extension .js**. Ensuite, nous allons faire appel à notre fichier JavaScript depuis notre fichier HTML.

Pour cela, on va à nouveau utiliser un élément **script** mais nous n'allons cette fois-ci rien écrire à l'intérieur. A la place, on va plutôt ajouter **un attribut src** à notre élément script et lui passer en valeur l'adresse du fichier. Si votre **fichier .js** se situe dans le même dossier que votre fichier .html, il suffira d'indiquer le nom du fichier en valeur de **l'attribut src**.

La place du code et l'ordre d'exécution

Un navigateur va lire et exécuter le code dans l'ordre de son écriture.

Unité 0 : Introduction au JavaScript

Plus précisément, lorsque le navigateur arrive à un élément script, il va stopper le traitement du reste du HTML jusqu'à ce que le code JavaScript soit chargé dans la page et exécuté.

Cette façon de faire semble en effet résoudre le problème à priori mais n'est pas toujours optimale en termes de performances. En effet résumons ce qu'il se passe dans ce cas :

- Le navigateur commence à analyser (ou à traiter) le code HTML ;
- L'analyseur du navigateur rencontre un élément script ;
- Le contenu JavaScript est demandé et téléchargé (dans le cas où il se situe dans un fichier externe) puis exécuté. Durant tout ce temps, l'analyseur bloque l'affichage du HTML, ce qui peut dans le cas où le script est long ralentir significativement le temps d'affichage de la page ;
- Dès que le JavaScript a été exécuté, le contenu HTML finit d'être analysé et est affiché.

Unité 0 : Introduction au JavaScript

Ce problème précis de temps d'attente de chargement des fichiers JavaScript va pouvoir être résolu en grande partie grâce au téléchargement asynchrone des données qui va pouvoir être ordonné en précisant un **attribut async** ou **defer** dans nos éléments script.

async : charger/exécuter les scripts de façon asynchrone.

defer : différer l'exécution à la fin du chargement du document.

Les commentaires en JavaScript

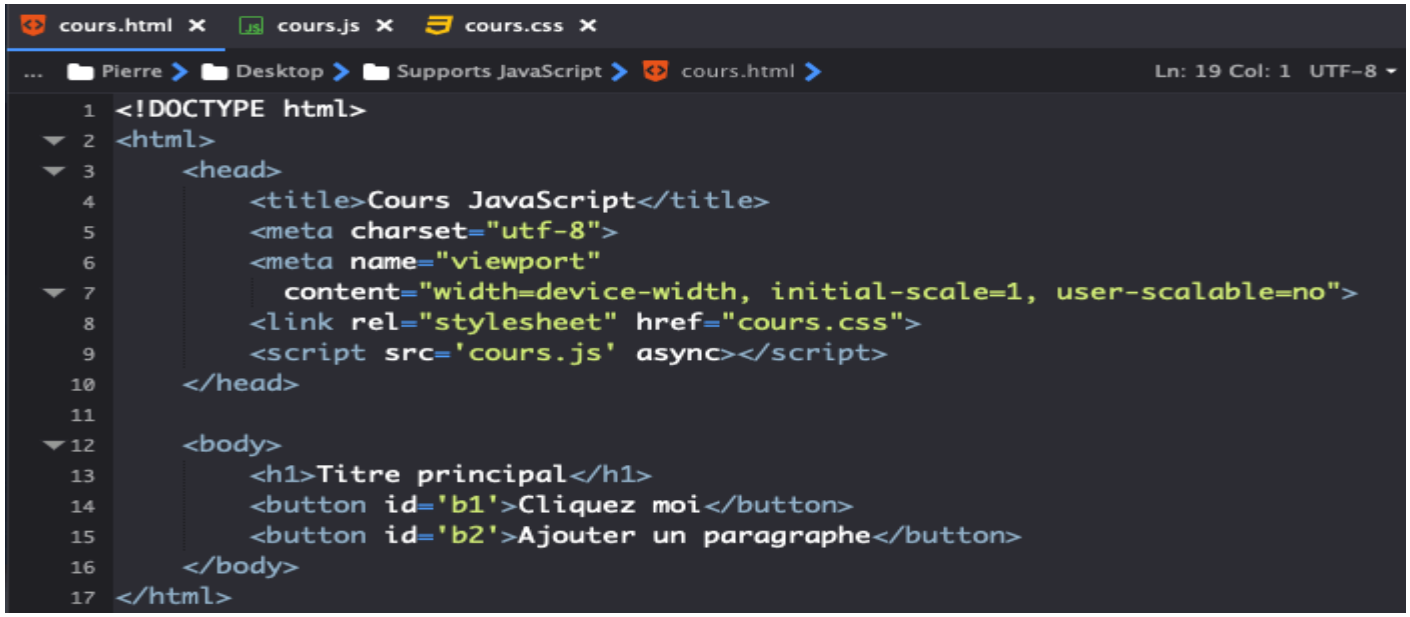
Les commentaires sont des lignes de texte (des indications) placées au milieu d'un script et servant à documenter le code, c'est-à-dire à expliquer ce que fait tel ou tel bout de script et éventuellement comment le manipuler.

En JavaScript, il existe deux types de commentaires qui vont s'écrire différemment : les commentaires mono-ligne et les commentaires multi-lignes.

Unité 0 : Introduction au JavaScript

Pour écrire un **commentaire multilignes**, il faudra entourer le texte de notre commentaire avec la syntaxe suivante **/* */**.
Pour écrire un **commentaire monoligne**, on utilisera **un double slash //** qui sera suivi du texte de notre commentaire (ou éventuellement la syntaxe multilignes).

Exemple de code JavaScript



Unité 0 : Introduction au JavaScript

Exemple de code JavaScript

cours.html × cours.js × cours.css ×

... Cours > JavaScript > Supports JavaScript > cours.js > Ln: 23 Col: 1 UTF-8

```
1  let bonjour = document.getElementById('b1');
2  let ajouter = document.getElementById('b2');
3
4  bonjour.addEventListener('click', alerte);
5  ajouter.addEventListener('click', ajout);
6
7  function alerte(){
8      alert('Bonjour');
9  }
10 function ajout(){
11     let para = document.createElement('p');
12     para.textContent = 'Paragraphe ajouté';
13     document.body.appendChild(para);
14 }
```

Unité 0 : Introduction au JavaScript

DOM ou Document Object Model

Le DOM est une interface de programmation pour des documents HTML ou XML qui représente le document (la page web actuelle) sous une forme qui permet aux langages de script comme le JavaScript d'y accéder et d'en manipuler le contenu et les styles. Le DOM est ainsi une représentation structurée du document sous forme « d'arbre » crée automatiquement par le navigateur. Il contient l'interface Document qui représente une page et sert de point d'entrée dans l'arborescence du DOM.

Pour utiliser les méthodes de l'interface Document, nous allons tout simplement utiliser la propriété document qui a plusieurs méthodes dont getElementById () que nous allons beaucoup utiliser.

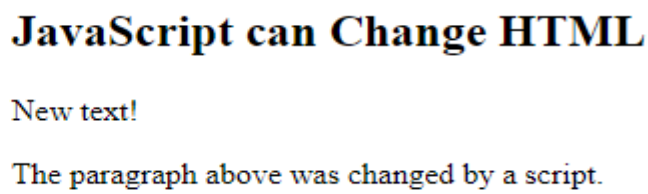
La méthode getElementById () renvoie l'élément qui a l'attribut ID avec la valeur spécifiée. Cette méthode est l'une des méthodes les plus courantes dans le DOM HTML et est utilisée presque chaque fois que vous souhaitez manipuler ou obtenir des informations sur un élément de votre document.

Unité 0 : Introduction au JavaScript

Elle utilise la propriété innerHTML qui permet de modifier le contenu d'un élément HTML.
Pour modifier le contenu d'un élément HTML, utilisez cette syntaxe:

document.getElementById (id) .innerHTML = nouveau HTML

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript can Change HTML</h2>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML = "New
text!";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```



Unité 0 : Introduoncattion au JavaScript

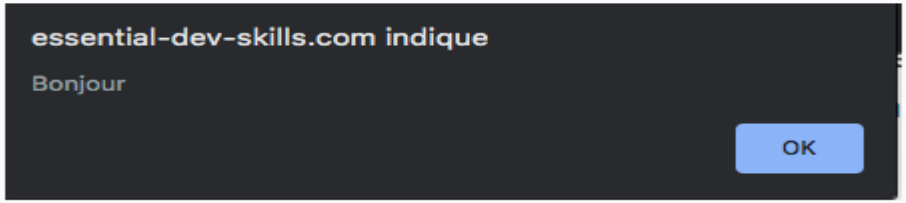
Interaction: alert, prompt, confirm

Une boîte de dialogue en JavaScript permet d'effectuer une interaction avec un internaute. Elles sont natives et peuvent avoir une apparence différente en fonction des navigateurs. Elles sont au nombre de trois :

alert() qui permet d'avertir l'utilisateur

```JS
1 alert()
```

Ceci va vous afficher une boîte de dialogue avec une indication : monsite.com vous indique :

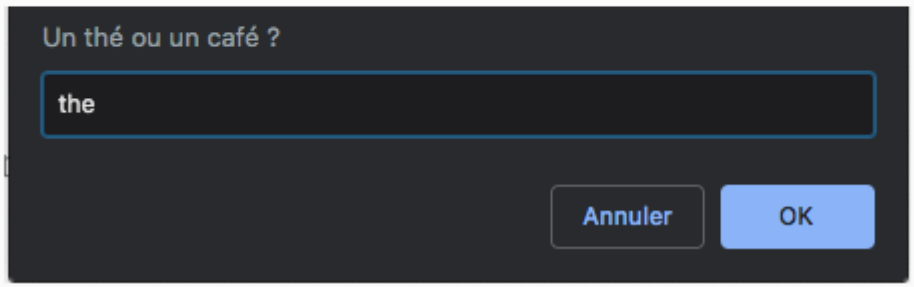


Unité 0 : Introduction au JavaScript

Interaction: alert, prompt, confirm

prompt() qui va proposer un choix à l'utilisateur.

```js
let result = prompt("Un thé ou un café ?", "the")
```



Unité 0 : Introduction au JavaScript

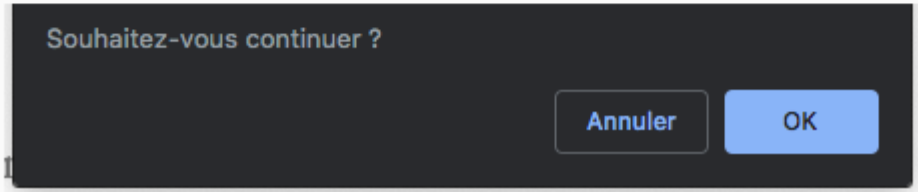
Interaction: alert, prompt, confirm

confirm() qui demander à l'internaute de compléter un champ dans lequel on va pouvoir récupérer l'information.

JS

```
1 const result = confirm("Souhaitez-vous continuer ?")
```

Un bouton « OK » qui va valider l'action et un bouton « Annuler ». En fonction du choix, vous allez recevoir une valeur de retour booléenne à **true** (Ok) ou **false** (Annuler).



Unité 0 : Introduction au JavaScript

Les évenements

Pour écouter et répondre à un évènement, nous allons définir ce qu'on appelle des gestionnaires d'évènements. Un gestionnaire d'évènements est toujours divisé en deux parties : une partie qui va servir à écouter le déclenchement de l'évènement, et une partie gestionnaire en soi qui va être le code à exécuter dès que l'évènement se produit.

Aujourd'hui, en JavaScript, il existe trois grandes façons d'implémenter un gestionnaire d'évènements :

- On peut utiliser des attributs HTML de type évènement (non recommandé) ;
- On peut utiliser des propriétés JavaScript liées aux évènements ;
- On peut utiliser la méthode addEventListener() (recommandé).

On utilise très souvent des attributs HTML pour prendre en charge un évènement.

Unité 0 : Introduction au JavaScript

Les évenements

Ces attributs HTML de « type évènement » possèdent souvent le nom de l'évènement qu'ils doivent écouter et gérer précédé par « on » comme par exemple :

- L'attribut onclick pour l'évènement « clic sur un élément » ;
- L'attribut onmouseover pour l'évènement « passage de la souris sur un élément » ;
- L'attribut onmouseout pour l'évènement « sortie de la souris d'élément » ;
- Etc.

Unité 0 : Introduction au JavaScript

Exercice :

Unité 1 : Les variables et types de valeurs JavaScript

Qu'est-ce qu'une variable ?

Une variable est un conteneur servant à stocker des informations de manière temporaire, comme une chaine de caractères (un texte) ou un nombre par exemple.

Notez bien déjà qu'une variable en soi et la valeur qu'elle va stocker sont deux éléments différents et qui ne sont pas égaux. Encore une fois, une variable n'est qu'un conteneur. Vous pouvez imaginer une variable comme une boite dans laquelle on va pouvoir placer différentes choses au cours du temps.

Les règles de déclaration des variables en JavaScript

Une variable est donc un conteneur ou un espace de stockage temporaire qui va pouvoir stocker une valeur. Lorsqu'on stocke une valeur dans une variable, on dit également qu'on assigne une valeur à une variable. Pour déclarer une variable en JavaScript, nous allons devoir utiliser le mot clef **var** ou le mot clef **let.** la coexistence des mots clefs var et let en JavaScript est due avant tout à ce souci d'héritage du langage

Unité 1 : Les variables et types de valeurs JavaScript

La syntaxe de déclaration des variables avec let correspond à la nouvelle syntaxe. La syntaxe avec var est l'ancienne syntaxe qui est vouée à disparaitre.

Concernant le nom de nos variables, nous avons une grande liberté dans le nommage de celles-ci mais il y a quand même quelques règles à respecter :

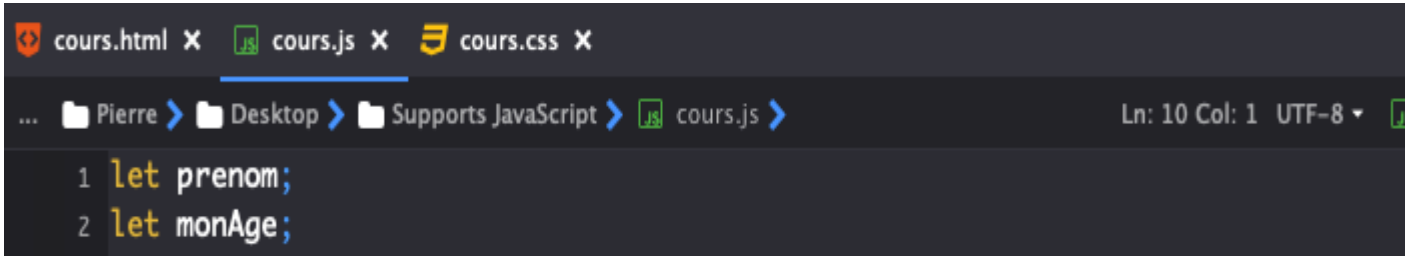
- Le nom d'une variable doit obligatoirement commencer par une lettre ou un underscore (_) et ne doit pas commencer par un chiffre ;
- Le nom d'une variable ne doit contenir que des lettres, des chiffres et des underscores mais pas de caractères spéciaux ;
- Le nom d'une variable ne doit pas contenir d'espace.

De plus, notez que le nom des variables est sensible à la casse en JavaScript. Vous pouvez également noter qu'on utilise généralement la convention lower camel case pour définir les noms de variable en JavaScript.

Unité 1 : Les variables et types de valeurs JavaScript

Cette convention stipule simplement que lorsqu'un nom de variable est composé de plusieurs mots, on colle les mots ensemble en utilisant une majuscule pour chaque mot sauf le premier. Par exemple, si je décide de nommer une variable « monage » j'écrirai en JavaScript let **monAge** ou var **monAge.**

Exemple de création de variables



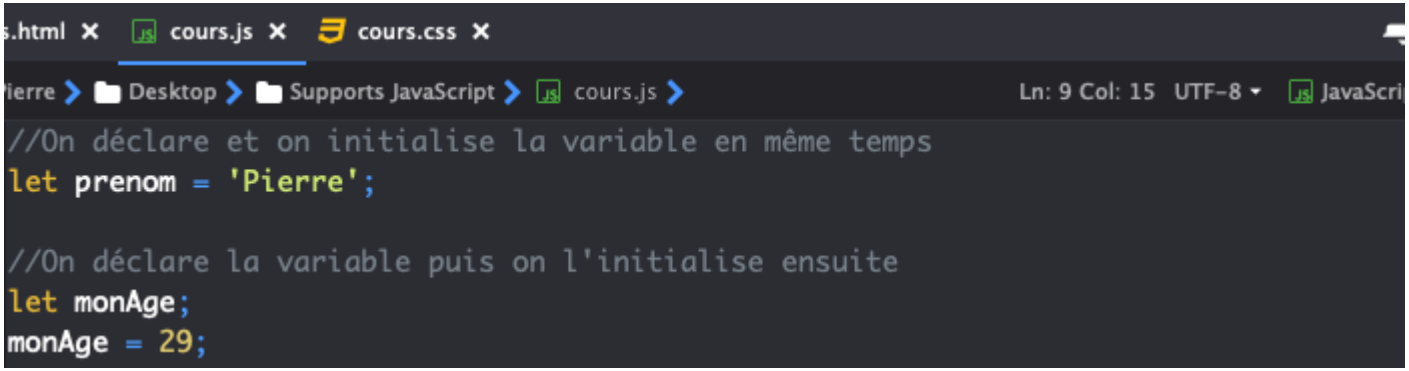
Nos deux premières variables sont désormais créées. Cependant, elles ne stockent aucune valeur pour le moment.

Unité 1 : Les variables et types de valeurs JavaScript

Initialiser une variable

Lorsqu'on assigne une valeur pour la première fois à une variable, c'est-à-dire lorsqu'on stocke une valeur dans une variable qui n'en stockait pas encore, on dit également qu'on initialise une variable.

On va pouvoir initialiser une variable après l'avoir déclarée ou au moment de sa déclaration.
Pour initialiser une variable, on utilise **l'opérateur =** qui est dans ce cas non pas un opérateur d'égalité mais un opérateur d'assignation ou d'affectation comme ceci :



Unité 1 : Les variables et types de valeurs JavaScript

Initialiser une variable

L'opérateur = est ici un opérateur d'affectation. Il sert à indiquer qu'on affecte (ou « assigne » ou encore « stocke ») la valeur à droite du signe dans le conteneur à gauche de celui-ci. Encore une fois, la variable n'est pas « égale » à sa valeur. le signe = ne possède pas du tout la même signification que le « égal » mathématique que vous utilisez dans la vie de tous les jours.

Modifier la valeur stockée dans une variable

Le propre d'une variable et l'intérêt principal de celles-ci est de pouvoir stocker différentes valeurs. Pour affecter une nouvelle valeur dans une variable déjà initialisée, on va se contenter d'utiliser à nouveau l'opérateur d'affectation =.En faisant cela, la nouvelle valeur va venir écraser l'ancienne valeur stockée qui sera alors supprimée.

Unité 1 : Les variables et types de valeurs JavaScript

Modifier la valeur stockée dans une variable

```
//On déclare et on initialise la variable en même temps
let prenom = 'Pierre';

//On déclare la variable puis on l'initialise ensuite
let monAge;
monAge = 29;

/*On modifie la valeur stockée dans preonm.
 *Notre variable stocke désormais la valeur "Mathilde"*/
prenom = 'Mathilde';
```

Les types de données en JavaScript

Les variables JavaScript vont pouvoir stocker différents types de valeurs, comme du texte ou un nombre par exemple. Par abus de langage, nous parlerons souvent de « types de variables » JavaScript.

Unité 1 : Les variables et types de valeurs JavaScript

Les types de données en JavaScript

En JavaScript, contrairement à d'autres langages de programmation, nous n'avons pas besoin de préciser à priori le type de valeur qu'une variable va pouvoir stocker. Le JavaScript va en effet automatiquement détecter quel est le type de la valeur stockée dans telle ou telle variable, et nous allons ensuite pouvoir effectuer différentes opérations selon le type de la variable, ce qui va s'avérer très pratique pour nous !

Une conséquence directe de cela est qu'on va pouvoir stocker différents types de valeurs dans une variable au fil du temps sans se préoccuper d'une quelconque compatibilité. Par exemple, une variable va pouvoir stocker une valeur textuelle à un moment dans un script puis un nombre à un autre moment.

En JavaScript, il existe 7 types de valeurs différents. Chaque valeur qu'on va pouvoir créer et manipuler en JavaScript va obligatoirement appartenir à l'un de ces types. Ces types sont les suivants :

Unité 1 : Les variables et types de valeurs JavaScript

Les types de données en JavaScript

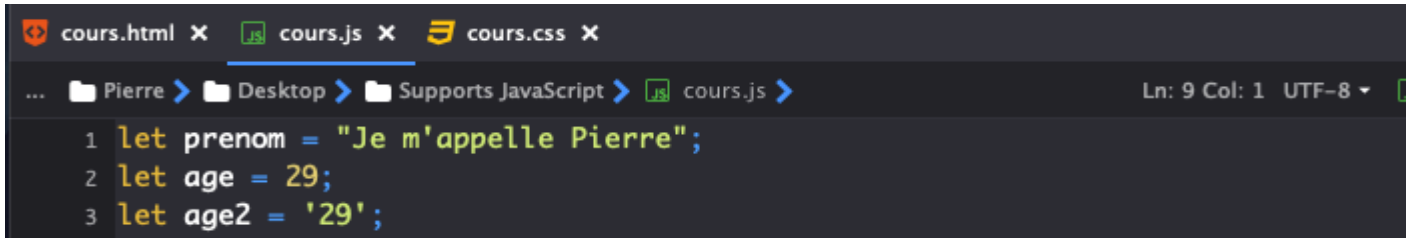
- String ou « chaine de caractères » en français ;
- Number ou « nombre » en français ;
- Boolean ou « booléen » en français ;
- Null ou « nul / vide » en français;
- Undefined ou « indéfini » en français ;
- Symbol ou « symbole » en français ;
- Object ou « objet » en français ;

Ce que vous devez bien comprendre ici est que les données vont pouvoir être manipulées différemment en fonction de leur type et qu'il est donc essentiel de les connaitre pour créer des scripts fonctionnels.

Unité 1 : Les variables et types de valeurs JavaScript

Le type chaîne de caractères ou String

Le premier type de données qu'une variable va pouvoir stocker est le **type String** ou **chaîne de caractères**. Une chaine de caractères est une séquence de caractères, ou ce qu'on appelle communément un texte.
Notez que toute valeur stockée dans une variable en utilisant des guillemets ou des apostrophes sera considérée comme une chaine de caractères, et ceci même dans le cas où nos caractères sont à priori des chiffres comme "29" par exemple.



Ici, notre première variable **let prenom** stocke la chaine de caractère « *Je m'appelle Pierre* ». Notre deuxième variable **let age**, quant à elle, stocke le nombre 29. En revanche, notre troisième variable let age2 stocke la chaine de caractères « 29 » et non pas un nombre.

Unité 1 : Les variables et types de valeurs JavaScript

Le type nombre ou Number

Les variables en JavaScript vont également pouvoir stocker des nombres. En JavaScript, et contrairement à la majorité des langages, il n'existe qu'un type prédéfini qui va regrouper tous les nombres qu'ils soient positifs, négatifs, entiers ou décimaux (à virgule) et qui est le type **Number**.
Pour affecter une valeur de type à une variable, on n'utilise ni guillemet ni apostrophe**Number**

```
let x = 10; //x stocke un entier positif
let y = -2; //y stocke un entier négatif
let z = 3.14; //z stocke un nombre décimal positif
```

Attention ici : lorsque nous codons nous utilisons toujours des notations anglo-saxonnes, ce qui signifie qu'il faut remplacer nos virgules par des points pour les nombres décimaux.

Unité 1 : Les variables et types de valeurs JavaScript

Le type de données booléen (Boolean)

Une variable en JavaScript peut encore stocker une valeur de type **Boolean**, c'est-à-dire un booléen.
Un booléen, en algèbre, est une valeur binaire (soit 0, soit 1). En informatique, le type booléen est un type qui ne contient que deux valeurs : les valeurs **true** (vrai) et **false** (faux).

Une nouvelle fois, faites bien attention : pour qu'une variable stocke bien un booléen, il faut lui faire stocker la valeur true ou false, sans guillemets ou apostrophes car dans le cas contraire le JavaScript considèrera que c'est la chaine de caractères « true » ou « false » qui est stockée et on ne pourra plus effectuer les mêmes opéra

```
let vrai = true; //Stocke le booléen true
let faux = false; //Stocke le booléen false

/*On demande au JavaScript d'évaluer la comparaison "8 > 4". Comme 8 est bien
 *strictement supérieur à ', le JavaScript renvoie true en résultat. On
 *stocke ensuite ce résultat (le booléen true) dans la variable let resultat*/
let resultat = 8 > 4;
```

Unité 1 : Les variables et types de valeurs JavaScript

Le type de données booléen (Boolean)

Le troisième exemple est un peu plus complexe à comprendre. Ici, vous devez comprendre que l'affectation se fait en dernier et que la comparaison est faite avant. Lorsqu'on écrit **« 8 > 4 »,** (qui signifie 8 strictement supérieur à 4) on demande au JavaScript d'évaluer cette comparaison.
Si la comparaison est vérifiée (si elle est vraie), alors JavaScript renvoie le booléen true. Dans le cas contraire, il renvoie le booléen false. On stocke ensuite le booléen renvoyé dans la variable let resultat.

Les types de valeurs Null et Undefined

Les types de valeurs **Null** et **Undefined** sont des types un peu particuliers car ils ne contiennent qu'une valeur chacun : les valeurs **null** et **undefined**.

La valeur null correspond à l'absence de valeur ou du moins à l'absence de valeur connue. Pour qu'une variable contienne null, il va falloir stocker cette valeur qui représente donc l'absence de valeur de manière explicite.

Unité 1 : Les variables et types de valeurs JavaScript

Les types de valeurs Null et Undefined

La valeur undefined correspond à une variable **« non définie »,** c'est-à-dire une variable à laquelle on n'a pas affecté de valeur.

Cette définition peut vous paraitre similaire à celle de null et pourtant ces deux valeurs ont une signification différente.
Si on déclare une variable sans lui attribuer de valeur, alors son type sera Undefined.
Si on déclare une variable et qu'on lui passe null, alors son type sera Object.

Unité 2 : Les opérateurs arithmétiques et d'affectation JavaScript

Qu'est-ce qu'un opérateur ?

Un opérateur est un symbole qui va être utilisé pour effectuer certaines actions notamment sur les variables et leurs valeurs. Il existe différents types d'opérateurs qui vont nous servir à réaliser des opérations de types différents. Nous allons nous concentrer sur les plus fréquemment utilisés :

- Les opérateurs arithmétiques ;
- Les opérateurs d'affectation / d'assignation ;

Les opérateurs arithmétiques

Les opérateurs arithmétiques vont nous permettre d'effectuer toutes sortes d'opérations mathématiques entre les valeurs de type Number contenues dans différentes variables. Nous allons pouvoir utiliser les opérateurs arithmétiques suivants en JavaScript :

Unité 2 : Les opérateurs arithmétiques et d'affectation JavaScript

Opérateur	Nom de l'opération associée
+	Addition
–	Soustraction
*	Multiplication
/	Division
%	Modulo (reste d'une division euclidienne)
**	Exponentielle (élévation à la puissance d'un nombre par un autre)

Unité 2 : Les opérateurs arithmétiques et d'affectation JavaScript

Les opérateurs d'affectation

Les opérateurs d'affectation vont nous permettre, comme leur nom l'indique, d'affecter une certaine valeur à une variable. Nous connaissons déjà bien l'opérateur d'affectation le plus utilisé qui est le signe **=**
Il existe d'autres opérateurs qui sont:

Opérateur	Définition
+=	Additionne puis affecte le résultat
-=	Soustrait puis affecte le résultat
*=	Multiplie puis affecte le résultat
/=	Divise puis affecte le résultat
%=	Calcule le modulo puis affecte le résultat

Unité 2 : Les opérateurs arithmétiques et d'affectation JavaScript

La concaténation

Concaténer signifie littéralement « mettre bout à bout ». La concaténation est un mot généralement utilisé pour désigner le fait de rassembler deux chaines de caractères pour en former une nouvelle.

En JavaScript, l'opérateur de concaténation est le signe +. Faites bien attention ici : lorsque le signe + est utilisé avec deux nombres, il sert à les additionner. Lorsqu'il est utilisé avec autre chose que deux nombres, il sert d'opérateur de concaténation.

Les constantes en JavaScript

Une constante est similaire à une variable au sens où c'est également un conteneur pour une valeur. Cependant, à la différence des variables, on ne va pas pouvoir modifier la valeur d'une constante. Pour créer ou déclarer une constante en JavaScript, nous allons utiliser le mot clef **const**.

Unité 2 : Les opérateurs arithmétiques et d'affectation JavaScript

Les constantes en JavaScript

On va pouvoir déclarer une constante exactement de la même façon qu'une variable à la différence qu'on va utiliser **const** à la place de **let**.

```
const prenom = 'Pierre';
const age = 29;
```

Unité 3 : Les Structures de contrôles JavaScript

Les structures de contrôle conditionnelles

Elles vont nous permettre d'exécuter une série d'instructions si une condition donnée est vérifiée ou (éventuellement) une autre série d'instructions si elle ne l'est pas.

Nous avons accès aux structures conditionnelles suivantes en JavaScript :

- La condition if (si) ;
- La condition if… else (si… sinon) ;
- La condition if… elseif… else (si… sinon si… sinon).

Les opérateurs de comparaison

En pratique, nous allons donc comparer la valeur d'une variable à une certaine autre valeur donnée et selon le résultat de la comparaison exécuter un bloc de code ou pas. Pour comparer des valeurs, nous allons devoir utiliser des opérateurs de comparaison.

Unité 3 : Les Structures de contrôles JavaScript

Les opérateurs de comparaison

Opérateur	Définition
==	Permet de tester l'égalité sur les valeurs
===	Permet de tester l'égalité en termes de valeurs et de types
!=	Permet de tester la différence en valeurs
<>	Permet également de tester la différence en valeurs
!==	Permet de tester la différence en valeurs ou en types
<	Permet de tester si une valeur est strictement inférieure à une autre
>	Permet de tester si une valeur est strictement supérieure à une autre
<=	Permet de tester si une valeur est inférieure ou égale à une autre
>=	Permet de tester si une valeur est supérieure ou égale à une autre

Unité 3 : Les Structures de contrôles JavaScript

Les opérateurs logiques

Les opérateurs logiques sont des opérateurs qui vont principalement être utilisés avec des valeurs booléennes et au sein de conditions. Ce sont :

Opérateur (nom)	Opérateur (symbole)	Description
AND (ET)	&&	Lorsqu'il est utilisé avec des valeurs booléennes, renvoie true si toutes les comparaisons sont évaluées à true ou false sinon
OR (OU)	||	Lorsqu'il est utilisé avec des valeurs booléennes, renvoie true si au moins l'une des comparaisons est évaluée à true ou false sinon
NO (NON)	!	Renvoie false si une comparaison est évaluée à true ou renvoie true dans le cas contraire

Unité 3 : Les Structures de contrôles JavaScript

L'instruction switch en JavaScript

L'instruction switch va nous permettre d'exécuter un code en fonction de la valeur d'une variable. On va pouvoir gérer autant de situations ou de « cas » que l'on souhaite. Sa syntax est la suivante :

```
switch (expression) {
   case x:
       // execute case x code block
       break;
   case y:
       // execute case y code block
       break;
   default:
       // execute default code block
}
```

Unité 3 : Les Structures de contrôles JavaScript

Les boucles

Les boucles vont nous permettre d'exécuter plusieurs fois un bloc de code, c'est-à-dire d'exécuter un code « en boucle » tant qu'une condition donnée est vérifiée et donc ainsi nous faire gagner beaucoup de temps dans l'écriture de nos scripts. En voici quelques uns :

- La boucle while (« tant que ») ;
- La boucle do… while (« faire… tant que ») ;
- La boucle for (« pour ») ;

La boucle JavaScript while

La boucle while (« tant que » en français) va nous permettre de répéter une série d'instructions tant qu'une condition donnée est vraie c'est-à-dire tant que la condition de sortie n'est pas vérifiée. Sa syntaxe est :

Unité 3 : Les Structures de contrôles JavaScript

La boucle JavaScript while

Sa syntaxe est :

```
while (condition) {
 instructions ;
}
```

La boucle JavaScript do….while

La boucle do… while (« faire… tant que ») est relativement semblable à la boucle while dans sa syntaxe. La grande différence entre les boucles while et do… while va résider dans l'ordre dans lequel vont se faire les opérations.

```
do{

Instructions;
}
while(b < 10);
```

Unité 3 : Les Structures de contrôles JavaScript

La boucle JavaScript for

La boucle for (« pour » en français) est structurellement différente des boucles while et do… while puisqu'on va cette fois-ci initialiser notre variable à l'intérieur de la boucle.
La boucle for utilise une syntaxe relativement condensée et est relativement puissante ce qui en fait la condition la plus utilisée en JavaScript. Sa syntaxe est :

```
for(let i = 0; i < 10; i++){
    instructions;
}
```

Unité 3 : Les Structures de contrôles JavaScript

Exercice

Unité 4 : Les Fonctions en JavaScript

Une fonction correspond à un bloc de code nommé et réutilisable et dont le but est d'effectuer une tâche précise. En JavaScript, comme dans la plupart des langages les supportant, nous allons très souvent utiliser des fonctions car celles-ci possèdent de nombreux atouts que l'on va énumérer par la suite.

Fonctions JavaScript prédéfinies

Le langage JavaScript dispose de nombreuses fonctions que nous pouvons utiliser pour effectuer différentes tâches. Les fonctions définies dans le langage sont appelées fonctions prédéfinies ou fonctions prêtes à l'emploi car il nous suffit de les appeler pour nous en servir. Pour être tout à fait précis, les fonctions prédéfinies en JavaScript sont des méthodes. Une méthode est tout simplement le nom donné à une fonction définie au sein d'un objet. Pour le moment, nous allons considérer que ce sont simplement des fonctions.

Par exemple, le JavaScript dispose d'une fonction nommée **random()** (qui appartient à l'objet Math) et qui va générer un nombre décimal aléatoire entre 0 et 1

ou encore d'une fonction replace() (qui appartient cette fois-ci à l'objet String) qui va nous permettre de chercher et de remplacer une expression par une autres dans une chaine de caractères.
L'intérêt principal des fonction prédéfinies est de nous permettre de réaliser des opérations complexes de manière très simple : en les appelant, tout simplement.

Les fonctions personnalisées

En plus des nombreuses fonctions JavaScript prédéfinies et immédiatement utilisables, nous allons pouvoir créer nos propres fonctions en JavaScript lorsque nous voudrons effectuer une tâche très précise.

Pour pouvoir utiliser une fonction personnalisée, en pratique, il faut déjà la définir. Pour définir une fonction, on va utiliser le mot clef **function** suivi du nom que l'on souhaite donner à notre fonction puis d'un couple de parenthèses dans lesquelles on peut éventuellement définir des paramètres et finalement d'un couple d'accolades dans lesquelles on va placer le code de notre fonction.

Unité 4 : Les Fonctions en JavaScript

Les fonctions personnalisées

Une fois notre fonction définie, on n'aura plus qu'à l'appeler pour l'utiliser. Voyons immédiatement comment faire en pratique.

```
/*On définit deux fonctions personnalisées.
 *La fonction aleatoire() se sert de la fonction (méthode) random().
 *La fonction multiplication() multiplie deux nombres entre eux.
 *On utilise une instruction return pour que nos fonctions, une fois appelées,
 *retournent le résultat de leur calcul afin qu'on puisse utiliser ce résultat*/
function aleatoire(){
    return Math.random() * 100;
}

function multiplication(nombre1, nombre2){
    //Attention : les "+" sont utilisés pour la concaténation !
    return nombre1 + ' * ' + nombre2 + ' = ' + (nombre1 * nombre2);
}
```

Notez qu'on utilise également ici pour nos deux fonctions une instruction **return**. Cette instruction va permettre à nos fonctions de retourner une valeur qu'on va ensuite pouvoir manipuler.

Unité 4 : Les Fonctions en JavaScript

Récapitulatif sur les fonctions

Voici un petit résumé des choses importantes à retenir à votre niveau sur les fonctions :

- Les fonctions sont des blocs de code nommés et réutilisables et dont le but est d'effectuer une tâche précise ;
- Il existe deux grands types de fonctions en JavaScript : les fonction natives ou prédéfinies (qui sont en fait des méthodes) qu'on n'aura qu'à appeler et les fonctions personnalisées qu'on va pouvoir créer ;
- Pour exécuter le code d'une fonction, il faut l'appeler. Pour cela, il suffit d'écrire son nom suivi d'un couple de parenthèses en passant éventuellement des arguments dans les parenthèses ;
- On crée une fonction personnalisée grâce au mot clef **function** ;
- Si une fonction a besoin qu'on lui passe des valeurs pour fonctionner, alors on définira des paramètres lors de sa définition. Lors de son appel, on lui passera des arguments qui prendront la place des arguments.

Unité 4 : Les Fonctions en JavaScript

Exercice

Prendre le fichier formulaire.html crée précédemment puis écrire un code javascript dans un second fichier js pour faire la table de multiplication d'un nombre que vous allez demander à l'utilisateur.

Conclusion