Formation

JavaScript, HTML dynamique

Semaine 1

Par Christopher Blassiaux

Christopher Blassiaux

Développeur d'Application Web Front-end & Formateur

chrisblassiaux@gmail.com

https://chrisb.fr/

Je travaille avec (liste non exhaustive) :

Je travaille chez :

14H41

Introduction :
Les technologies du Web

Qu’est-ce que Javascript ?

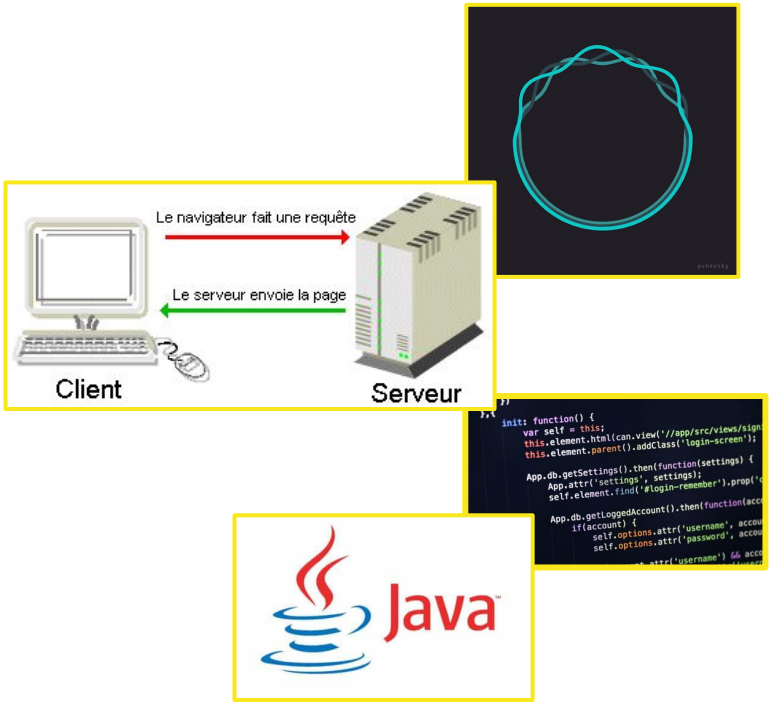
↪ Un langage de **programmation** développé en **1996** par Brendan Eich

↪ Permet d’**animer** et **dynamiser** les pages web

↪ Langage **côté client**

↪ Javascript peut-être aussi utilisé pour autre chose que le **web**

↪ À ne pas confondre avec le **Java**

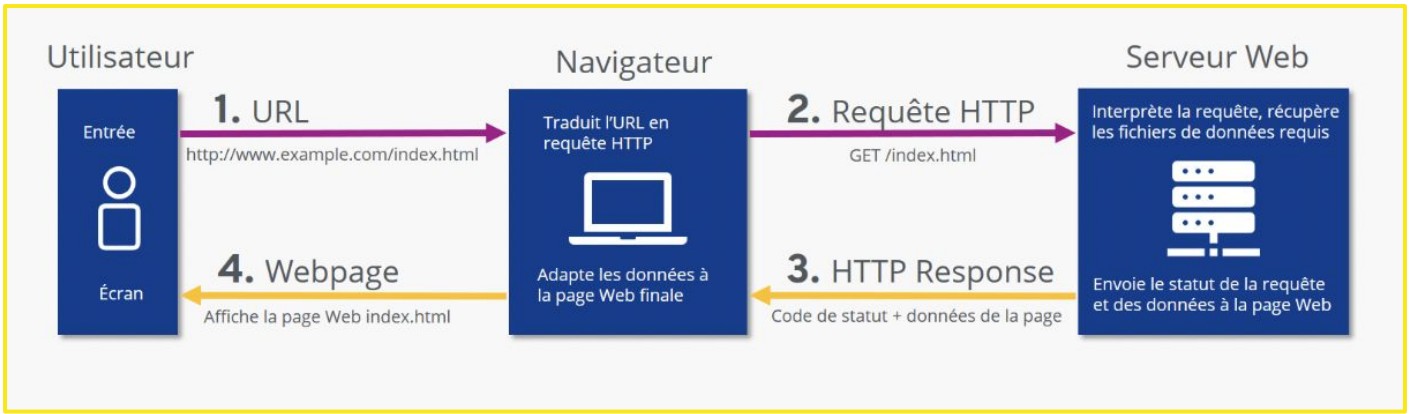


Protocoles HTTP & HTTPS

↪ **Fonctionnement de JS dans l'environnement du web** →

1 **-** Nom de domaine - **Protocole HTTP** -

2 **- Bonne requête** (2XX) **ou Mauvaise requête** (error 4XX client - 5XX server)



https://ionos.fr/

Les moteurs Javascript

↪ **Les moteurs Javascript :** des programmes logiciel qui **interprète** et **exécute** du code JS.

→ Apple : JavaScriptCore

→ Google : V8

→ Mozilla : SpiderMonkey

↪ Les moteurs actuellement **interprète** et **compile** le code afin d'améliorer les performances.

SpiderMonkey
Premier moteur JS

https://wikipedia.org/

Le web 2.0

↪ D'un **web pour les privilégiés** à un **web social** permettant :

- → la **création** de contenu
- → le **partage** du contenu
- → de **discuter** du contenu

↪ **Javascript est très important**

https://www.unow.fr/

Versions de Javascript

↪ De la même manière que pour le **HTML et le CSS** avec le **W3C**, le **Javascript devait être normé.**

↪ **ECMAScript** est le nom donné au **standard** dirigé par **Ecma international**.

↪ Javascript rejoins **ECMAScript** entre 1996 et 1997.

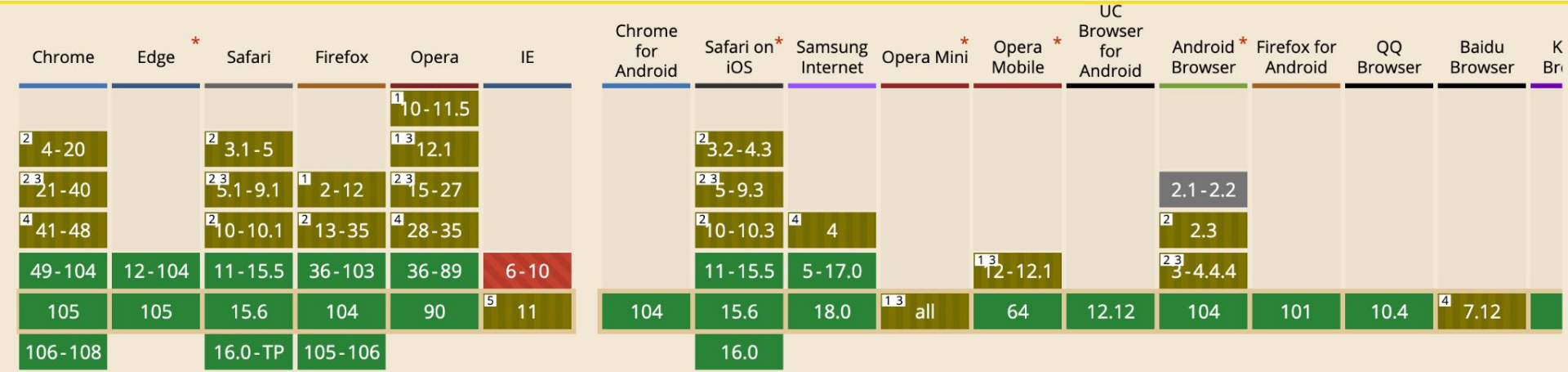
https://www.w3schools.com/

Ver	Official Name	Description
ES1	ECMAScript 1 (1997)	First edition
ES2	ECMAScript 2 (1998)	Editorial changes
ES3	ECMAScript 3 (1999)	Added regular expressions Added try/catch Added switch Added do-while
ES4	ECMAScript 4	Never released
ES5	ECMAScript 5 (2009) Read More	Added "strict mode" Added JSON support Added String.trim() Added Array.isArray() Added Array iteration methods Allows trailing commas for object literals
ES6	ECMAScript 2015 Read More	Added let and const Added default parameter values Added Array.find() Added Array.findIndex()
	ECMAScript 2016 Read More	Added exponential operator (**) Added Array.includes()
	ECMAScript 2017 Read More	Added string padding Added Object.entries() Added Object.values() Added async functions Added shared memory
	ECMAScript 2018 Read More	Added rest / spread properties Added asynchronous iteration Added Promise.finally() Additions to RegExp

Compatibilité des navigateurs

↪ Avec l'arrivée des différentes versions Javascript, les **moteurs JS** des **navigateurs** ont des **difficultés à suivre le rythme**.

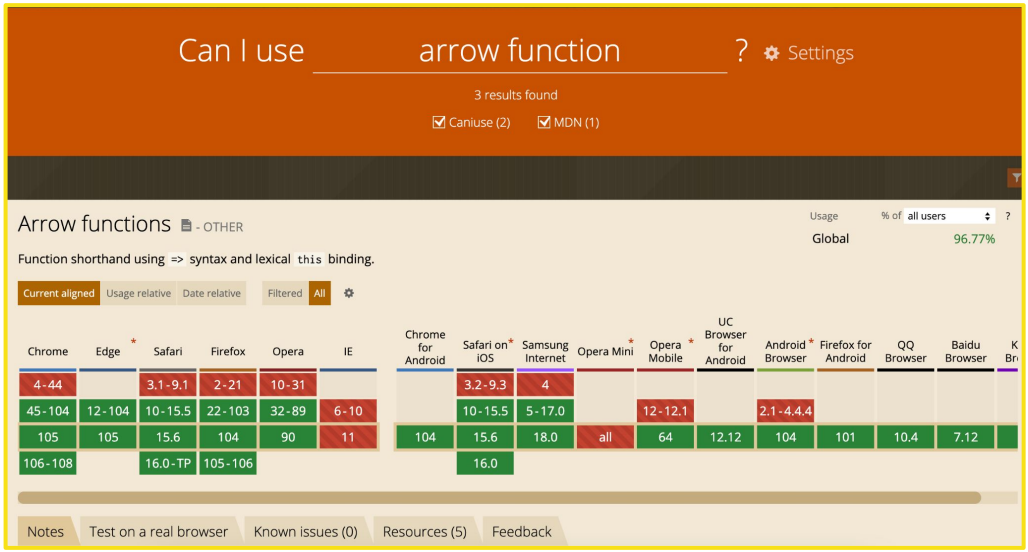
↪ Ce qui mène parfois à des **outils non compatibles** avec certains navigateurs.



Compatibilité des navigateurs

↪ **Comment découvrir ces incompatibilités ?**

https://caniuse.com/



Compatibilité des navigateurs

↪ Alors, comment éviter les problèmes de compatibilité ?

Faut-il utiliser uniquement les outils 100% compatibles ?

En attendant que ces outils soient compatibles, il existe des méthodes pour pouvoir tout de même les utiliser.

Babel est une librairie qui permet de **transpiler** (c'est-à-dire de convertir vers un langage ou une version d'un langage de même niveau) **votre code JavaScript récent en code compatible tous navigateurs**.

Accessibilité & référencement

↪ Javascript avec parcimonie, pour quelles raisons ?

→ L'**accessibilité**

→ Le **référencement (SEO)**

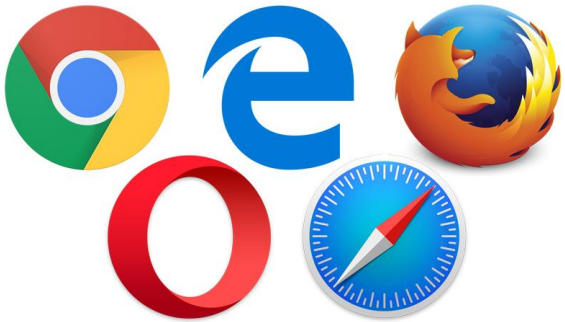
↪ Ils sont impactés lors d'une **utilisation excessive de JS.**

Ce que l'utilisateur voit n'est pas forcément ce que les robots voient. Le code est important pour ces outils.

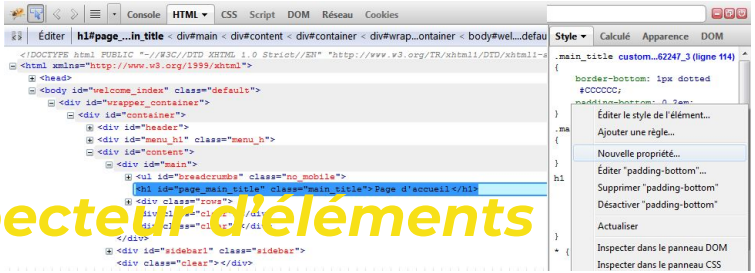
Les outils d'accessibilité ne sont pas toujours aptes à lire du code Javascript.

Les robots (scraping) ont plus de difficultés à parcourir une page codée en JS.

Les outils de développement



Les navigateurs

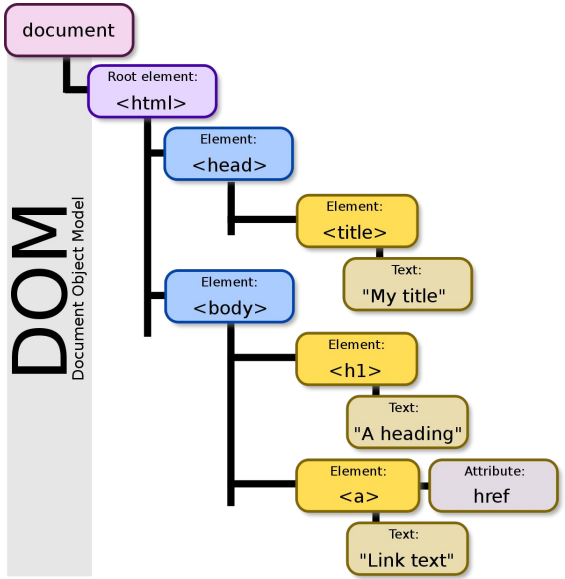


L'inspecteur d'éléments

Les éditeurs de code

Dynamic HTML (DHTML)

↪ Un **ensemble de technologies** associées afin de fournir des **pages** HTML plus **interactives**.



https://fr.wikipedia.org/

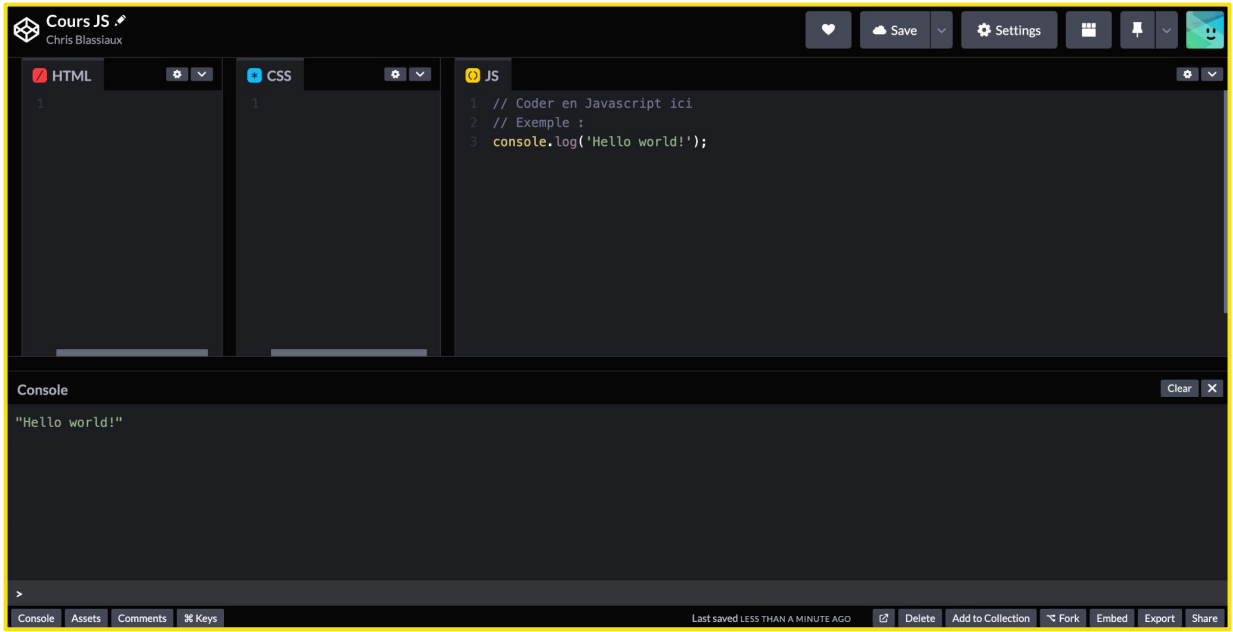
Concepts de programmation :
Le langage Javascript

Préparer un environnement de travail

↪ Optionnel :

↪ **Ce rendre sur :**

https://codepen.io/



Pratiquer en parallèle

Où écrire JS ?

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <script src="script.js"></script>
</head>
<body>
    <h1>Titre principal</h1>
</body>
</html>
```

Solution 1

Bonne pratique :
Cette méthode pose problème pour le DOM (Interaction avec le HTML)

Solution 2

Mauvaise pratique :
Cette méthode ne permet pas de lier le JS sur plusieurs pages à la fois et mélange les langages

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
    <script>

        // Écrire mon code JS ici

    </script>
</head>
<body>
    <h1>Titre principal</h1>
</body>
</html>
```

Où écrire JS ?

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1>Titre principal</h1>
    <button onclick="alert('Bonjour !')">Cliquez moi</button>

    <button onclick="(function(){
        let para = document.createElement('p');
        para.textContent = 'Paragraphe ajouté';
        document.body.appendChild(para);
    })();">
        Ajouter un paragraphe
    </button>
</body>
</html>
```

Solution 3

Mauvaise pratique :
Cette méthode ne permet pas de lier le JS sur plusieurs pages à la fois et mélange les langages

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Mon code HTML -->

    <script src="script.js"></script>
</body>
</html>
```

Solution 4

Meilleure pratique :
Cette méthode permet une meilleure organisation et de discuter avec le DOM sans difficultés.

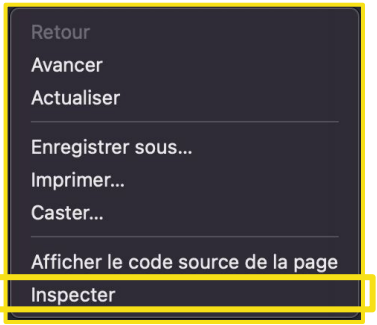
Débogueur - L'objet Console

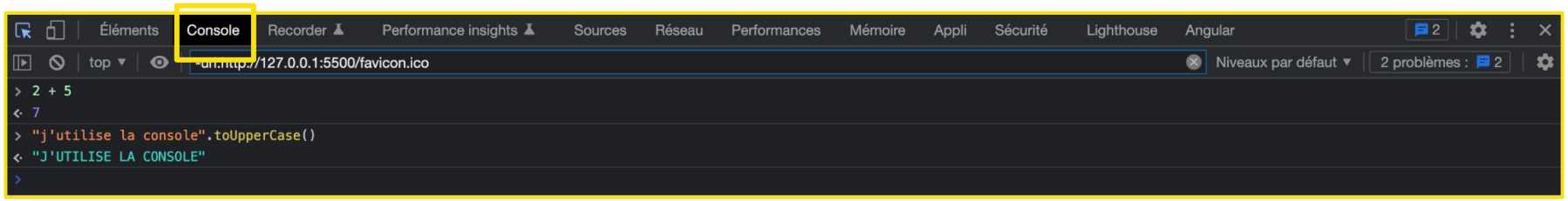
```
console.log("Failed to open the specified link")

console.table(["apples", "oranges", "bananas"]);
```

(index)	Values
0	"apples"
1	"oranges"
2	"bananas"

https://developer.mozilla.org/



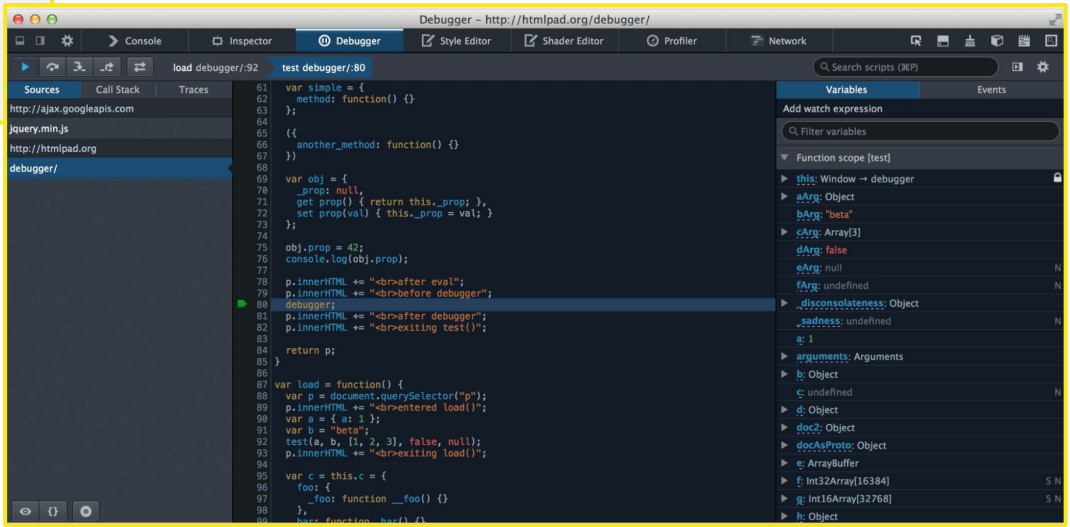


Débogueur - L'instruction

```
function codeProbablementBogue() {
    debugger;
    // exécuter des instructions qu'on veut
    // examiner, exécuter pas à pas etc.
}
```

https://developer.mozilla.org/

↪ En positionnant l'instruction **debugger**, vous pouvez avoir **un oeil affuté sur l'exécution du programme**



https://developer.mozilla.org/

Les variables : Qu'est-ce que c'est ?

↪ Un **espace de stockage** dans lequel on peut stocker
tous les types données en Javascript

Les variables : Déclaration et assignation

↪ **Déclarer une variable** - *Une variable peut être déclarée et assignée de manière séparée*

```
var maVariable; // Déclaration

maVariable = "J'assigne une chaîne de caractère"; // Assignation
```

↪ **Assigner à une variable** - *Une variable peut être déclarée et assignée en même temps*

```
var maSecondeVariable = "J'assigne une chaîne de caractère" // Déclaration & Assignation
```

Pratiquer en parallèle

Les variables : portée des variables

La portée d'une variable est l'accessibilité de celle-ci en fonction de sa zone de déclaration

↪ **Portée des variables :**

→ ***Portée globale :*** *Utilisable dans* ***tout le fichier .js*** *puisqu'elle n'a pas été déclarée dans un bloc*

→ ***Portée locale :*** *Utilisable uniquement* ***dans le bloc dans lequel elle est déclarée***

↪ **Let, Var, Const :** *Il existe 3 mots clés pour déclarer un espace de stockage*

```
var maVariable = 1; // Je n'utilise plus var - Précaunisé par ES6
let maSecondeVariable = 2; // J'utilise let dès que je déclare une variable
const maSecondeVariable = 3; // J'utilise const dès que je déclare une constante
```

Syntaxe ES6

Les types de données

↪ Il existe différents **types de données** :

```
let typeString = 'Un chaîne'        // String : chaîne de caractères
let typeNumber = 9                  // Number : Nombre
let typeBoolean = true              // Boolean : Booléen (true ou false)
let typeNull = null                 // Null
let typeUndefined = undefined       // Undefined : Non défini
let typeObject = { cle: 'valeur' } // Object : objet
let typeArray = [2, 4, 'string']    // Object : Les tableaux sont de type "object" en JS
```

En savoir plus sur tous les types de données :

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

Pratiquer en parallèle

Les types de données

↪ Nous pouvons **détecter** **le type** d'une donnée *via la fonction* ***typeof***

```
let typeArray = [2, 4, 'string'];
console.log(typeof typeArray); // Affiché dans le console : object
```

↪ Nous pouvons **convertir** **le type** d'une donnée *via différentes fonctions de conversion*

```
let typeNumber = 9;
console.log(typeNumber.toString()); // Affiché dans la console : '9'
// Autre exemple : Number()
```

Pratiquer en parallèle

La concaténation

↪ La concaténation de **deux chaînes de caractères** consiste à les **mettre bout à bout**

```
console.log('Hello ' + 'world'); // Output : Hello world
console.log(2 + ' days');        // Output : 2 days
console.log('Day ' + 3);         // Output : Day 3
```

↪ L'opérateur ***** ne peut pas être utilisé sur une chaîne de caractère en JS

```
console.log('Cou' * 2);          // Output : NaN
console.log('Cou'.repeat(2));    // Output : CouCou
```

Pratiquer en parallèle

Les structures conditionnelles

↪ **Exécuter** une ou plusieurs lignes de code **sous certaines conditions**

```
// Les blocs en Javascript
{

}
```

```
// Je souhaite exécuter ce bloc
if (condition) {
    // Je m'exécute uniquement si la condition est respectée (true)
}

// Exemples de conditions :
// 1 < 2 : true
// 3 > 4 : false
```

```
if (condition) {
    // soit ce code s'execute
} else if (condition) {
    // soit ce code s'execute
} else {
        // soit ce code s'execute
}
```

else if (sinon si) : optionnel, peut être répété plusieurs fois
else (sinon) : optionnel, ne peut être répété

Pratiquer en parallèle

Opérateurs

↪ Les opérateurs sont utiles pour construire des **conditions précises**

```
// OPÉRATEURS DE COMPARAISON
// == : est égal à (en valeur)
// === : est égal à (en valeur et en type)
// != : est différent de (en valeur)
// !== : est différent de (en valeur et en type)
//  > : est supérieur (strictement)
//  < : est inférieur (strictement)
//  >= : est supérieur ou égal
//  <= : est inférieur ou égal


// OPÉRATEURS LOGIQUES
// && : ET
// || : OU
// if (age >= 21 && age > 32)
```

Pratiquer en parallèle

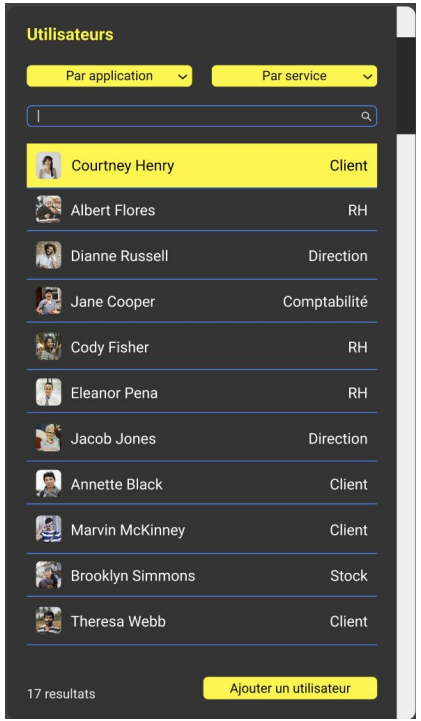
Les structures itératives : Les boucles

↪ **Exécuter** une ou plusieurs lignes de code **en boucle**
(un certain nombre de fois)

↪ **Exemple d'utilisation**

Affichage d'une liste d'utilisateurs sur la page.

x 17



Les structures itératives : Les boucles

↪ Il existe **différentes boucles** au sein de Javascript *(While, For, Do While, ForEach)*

Exemple : **Boucle WHILE**

```
// La Boucle WHILE - Tant que
// Variable d'incrémentation
let i = 0;

// Condiction
while(i <= 100) {
    // Instruction
    console.log(i);

    // Incrémentation
    i = i + 1; // Autre syntaxe : i++
}
```

Exemple : **Boucle FOR**

```
// Variable d'incrémentation | Condiction | Incrémentation
for (let i = 0; i <= 100; i++) {
    // Instruction
    console.log(i)
}
```

Pratiquer en parallèle

Fin de la formation

JavaScript, HTML dynamique

Semaine 1

Par Christopher Blassiaux