JavaScript partie 1

Avant-propos

Les nouvelles technologies évoluent sans cesse et à un rythme effréné, qu'elles concernent les appareils connectés ou les sites web, cette course en avant semble inarrêtable et difficilement prévisible, en raison de la multiplicité des facteurs de corrélation ou de causalité.

Ainsi, l'apprentissage se doit également d'être moderne.
Être un rat de bibliothèque adepte de l'apprentissage par coeur est désuet, "old school" depuis plusieurs années.

L'objectif de ce cours est de transmettre les connaissances nécessaires, mais surtout de développer l'autonomie, car comme cité précédemment les technologies évoluent bien trop vite pour permettre de capitaliser uniquement sur les acquis d'une formation.

Introduction

JavaScript est un langage de programmation de scripts principalement utilisé pour créer des pages web interactives. Il s'exécute dans le navigateur (côté client) et peut aussi être utilisé côté serveur (avec Node.js). Il a été créé en 1995 par Brendan Eich soit plusieurs années après HTML.

L'usage de Node.js permet ainsi de réaliser des sites internet Full Stack, en utilisant un seul langage, JavaScript, aussi bien côté serveur que côté client. Cela simplifie le développement et favorise la réutilisation du code.

JavaScript est également utilisé par des frameworks pour des applications destinées aux téléphones mobiles comme React Native, qui permet de créer des applications multiplateformes (iOS et Android) à partir d'une base de code unique.

Introduction

Certains frameworks tels que **Electron** permettent de développer des applications de bureau multiplateformes, tandis que des bibliothèques comme **TensorFlow.js** ouvrent la voie à l'intégration de l'intelligence artificielle directement dans le navigateur.

JavaScript s'impose aujourd'hui comme un langage universel, présent dans le web, le mobile, le desktop et même l'IoT (Internet of Things).

De cette manière JavaScript offre aux développeurs un écosystème riche et polyvalent.

Avantages et inconvénients du JavaScript 1/3

Avantages	Inconvénients
Langage universel du web : supporté par tous les navigateurs sans installation supplémentaire.	**Sécurité** : vulnérable aux attaques XSS (Cross-Site Scripting) si mal utilisé.
Polyvalence : utilisable côté client (front-end) et côté serveur (avec Node.js).	**Performances limitées** : moins rapide que des langages compilés comme C++ ou Rust pour des calculs lourds.
Écosystème riche : énorme quantité de bibliothèques, frameworks (React, Angular, Vue, etc.) et outils.	**Fragmentation** : trop de frameworks et de versions, ce qui peut compliquer les choix technologiques.

Avantages et inconvénients du JavaScript 2/3

Avantages	Inconvénients
Communauté massive : documentation abondante, forums, tutoriels, entraide.	**Asynchronisme complexe** : la gestion des callbacks/promises peut être difficile pour les débutants.
Interopérabilité : facile à intégrer avec HTML, CSS, APIs REST, WebSockets, etc.	**Compatibilité historique** : certains vieux navigateurs ne supportent pas toutes les fonctionnalités modernes (même si ES6+ a amélioré la situation).
Évolution rapide : le langage et ses standards (ECMAScript) s'améliorent chaque année.	**Évolution rapide** (oui, c'est aussi un inconvénient) : nécessité de se former en continu pour rester à jour.

Avantages et inconvénients du JavaScript 3/3

Avantages	Inconvénients
Multiplateforme : permet de créer des applis web, mobiles (React Native), desktop (Electron), et même IoT.	**Gestion mémoire** : le garbage collector peut causer des ralentissements imprévisibles.
Exclusivité détection événements : seul JavaScript (ou du code compilé en JavaScript) peut directement détecter les mouvements de la souris dans un navigateur.	

Où exécuter du JavaScript 1/2

Catégories	Exemples
Fichier	Dans une balise <script> en HTML
Fichier externe	Dans un fichier externe .js
Navigateur	Console, extensions
Outils en ligne	JSFiddle, CodePen, JSBin, StackBlitz, Replit, etc…
Environnement côté serveur	Node.js, Deno, Bun (runtime ultra-rapide basé sur JavaScriptCore le moteur de Safari). Autres runtimes (plus spécialisés) : Winter.js, Txiki.js, Napa.js, etc.

Où exécuter du JavaScript 2/2

Catégories	Exemples
Environnements mobiles et hybrides	React Native, Ionic, NativeScript : frameworks qui permettent d'exécuter du JS pour créer des applis mobiles natives.
Moteurs de jeux	Unity (via un plugin), Godot (avec bindings), ou encore Roblox (qui utilise Lua mais a des passerelles JS).
Applications desktop	via Electron (VS Code, Slack, Discord utilisent du JS), ou Tauri.
Objets connectés	Objets connectés / IoT : frameworks Espruino, Johnny-Five (exécution JS sur microcontrôleurs et cartes : Arduino, Raspberry Pi).

Premières instructions - document.write

Comme énoncé précédemment, JavaScript est intégré nativement dans le navigateur, il permet notamment de modifier la page internet.

1. Créer un fichier html classique
2. Ajouter la balise <script> juste avant </body>
3. Ajouter l'instruction document.write("Hello World !")

```
<body>
    <h1>Première instruction</h1>
    <script>
        document.write("Hello World !");
    </script>
</body>
```

Premières instructions - document.write

Cette instruction est montrée à titre d'exemple pour illustrer le fait que JS peut réécrire la page, mais est déconseillée. Nous allons privilégier append, appendChild…
D'autres instructions permettent d'ouvrir et de fermer le flux d'écriture.
L'exemple ci-dessous est purement illustratif, il n'est pas nécessaire de le comprendre :

```
<button onclick="ouvrirFenetre()">Ouvrir une nouvelle fenêtre</button>
<script>
  function ouvrirFenetre() {
    let nouvelleFenetre = window.open("", "Titre onglet", "width=400,height=300");
    nouvelleFenetre.document.open();
    nouvelleFenetre.document.write("<h1>Hello World depuis document.open() !</h1>");
    nouvelleFenetre.document.write("<p>Ce contenu est généré dynamiquement.</p>");
    nouvelleFenetre.document.close();
  }
</script>
```

Premières instructions - console.log

La console est un élément intégré au navigateur, elle est très utile pour obtenir des informations et pour débugger. L'instruction la plus connue est console.log, mais il en existe d'autres que nous verrons plus tard.

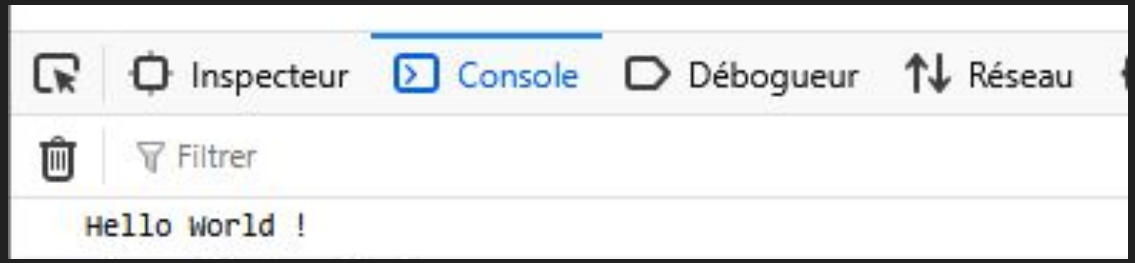
Instruction	Rôle principal
`console.log("Hello !")`	Pour visualiser des données ou déboguer.
`console.warn` et `error`	pour signaler des problèmes.
console.table	pour visualiser des données (tableaux et objets)
`console.time` / `timeEnd`	pour mesurer des performances.
`console.count`	pour suivre des appels.
console.group	pour organiser les logs.
`console.trace` et `assert`	pour déboguer plus finement.

Premières instructions - console.log

Saisir l'instruction :

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

Visualiser dans le navigateur (F12) :



Variables primitives - déclaration

7 types de variables primitives en JS :

String, Number, Boolean, Undefined, Null, Symbol (ES6), bigint (ES2020).

⚠️pas les objets (object), ni les fonctions ou tableaux (type spécialisé d'objet en JS).

Nous verrons dans un premier temps String, Number, Boolean.

Historiquement la déclaration de variables en JS se faisait avec le mot clé var, mais depuis ES6, il faut privilégier let ou const.

```
// Déclaration
let nom;
let age;
let estUnNinja;
```

Variables primitives - affectation

Affectation :

Comme en pseudo-code l'affectation de valeur se fait par `=`

```
// Affectation
nom = "Uzumaki";     // String
age = 33;            // Number
estUnNinja = true;   // Boolean

console.log(name);
console.log(age);
console.log(estUnNinja);
```

Concaténation

```
let nom = "Alexander"

// concaténation 1
let phrase1 = "Je suis " + nom;
console.log(phrase1);

// Concaténation 2
  // Template literals (littéraux de gabarits, ES6)
let phrase2 = `Je suis ${nom}`;
console.log(phrase2);

// Template literals operations
let age = 18;
console.log(`${age + 1}`);
```

Capturer saisie utilisateur : fonction prompt

La fonction prompt() est une fonction native permettant de capturer la saisie de l'utilisateur

```
<script>
    let prenom = prompt("Quel est votre prénom ?");
    let age = prompt("Quel est votre age ?");

    console.log(prenom);
    console.log(age);
</script>
```

Capturer réponse utilisateur : fonction confirm

La fonction confirm() est une fonction native permettant de capturer la réponse de l'utilisateur sous forme de booléen.

```
<script>
    let reponse = confirm("Voulez-vous continuer ?");
    console.log(reponse);
</script>
```

Capturer saisie user : fonction prompt

La fonction confirm() est une fonction native permettant de capturer la réponse de l'utilisateur sous forme de booléen.

```
<script>
    let reponse = confirm("Voulez-vous continuer ?");
    console.log(reponse);
</script>
```

Opérations de base

```
let x = 5;

x = x+1; // SOIT x++ OU x += 1;  => incrémentation -> increase
x = x-1; // SOIT x-- OU x -= 1; => décrémentation -> decrease
x = x*2; // SOIT x *= 2; Multiplication
x = x/2; // SOIT x /= 2; Division
```

Opérations de base - ordre des opérations

```
let x = 5;
let y = x++; // != let y = ++x;

console.log(x); // 6
console.log(y); // 5
```

Dans le cas présent la valeur de *x* est d'abord affectée à *y* puis *x* est incrémenté.

Opérations de base - modulo

```
<script>
    let x = 10%5; // 10 modulo 2 = 0. RESTE DE LA DIVISION
    let y = 7%2;  // 7 modulo 2 = 1

    console.log(x); // 0
    console.log(y); // 1
</script>
```

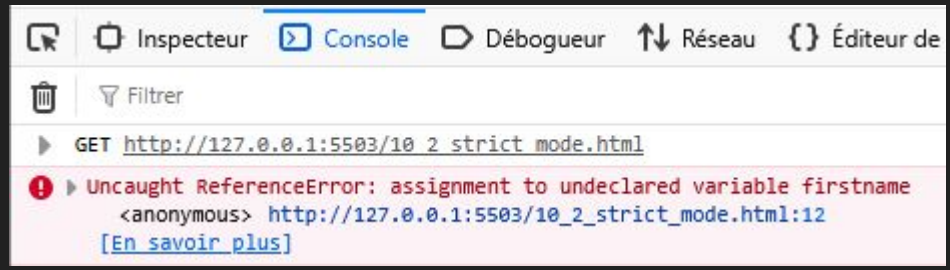
Strict mode

Utiliser le strict mode est vivement recommandé pour éviter les erreurs silencieuses :

```
<script>
    //Erreur silencieuse 1
    firstname = "Marcus";// pas de mot clé => var utilisé automatiquement
    console.log(firstname);
```


```
<script>
    "use strict";
```

=>



Opérateurs de comparaison

Ils comparent 2 valeurs et retournent un booléen

`==` : teste l'**égalité des valeurs** (conversion de type possible).
`===` : teste l'**égalité stricte** → la valeur **et** le type doivent être identiques.
`!=` : teste la **différence de valeurs** (conversion de type possible).
`!==` : teste la **différence stricte,** soit la valeur, soit le type (ou les deux) sont différents.
`>` : signifie **strictement supérieur à**.
`>=` : signifie **supérieur ou égal à**.
`<` : signifie **strictement inférieur à**.
`<=` : signifie **inférieur ou égal à**.

Opérateurs de logique

But : combiner ou inverser des expressions booléennes.

Retour : retourne un booléen.

&&	(ET logique)	: vrai si les deux conditions sont vraies
||	(OU logique)	: vrai si au moins une condition est vraie
!	(NON logique)	: inverse la valeur booléenne

Structure conditionnelle if

```
SI(CONDITION)
    1 = CONDITION REMPLIE => Instructions 1
SINON
    Instructions 2
```

```
let age = 25;
if(age >= 18)
    console.log("personne majeure");
else
    console.log("personne mineure");
```

Opérateur ternaire

```
let typeRepas = "vegan";
if(typeRepas === "vegan")
    console.log(`pas viande`);
else
    console.log(`viande`);
```

```
let typeRepas = "vegan";
console.log(typeRepas === "vegan" ? "pas de viande" : "viande");
```