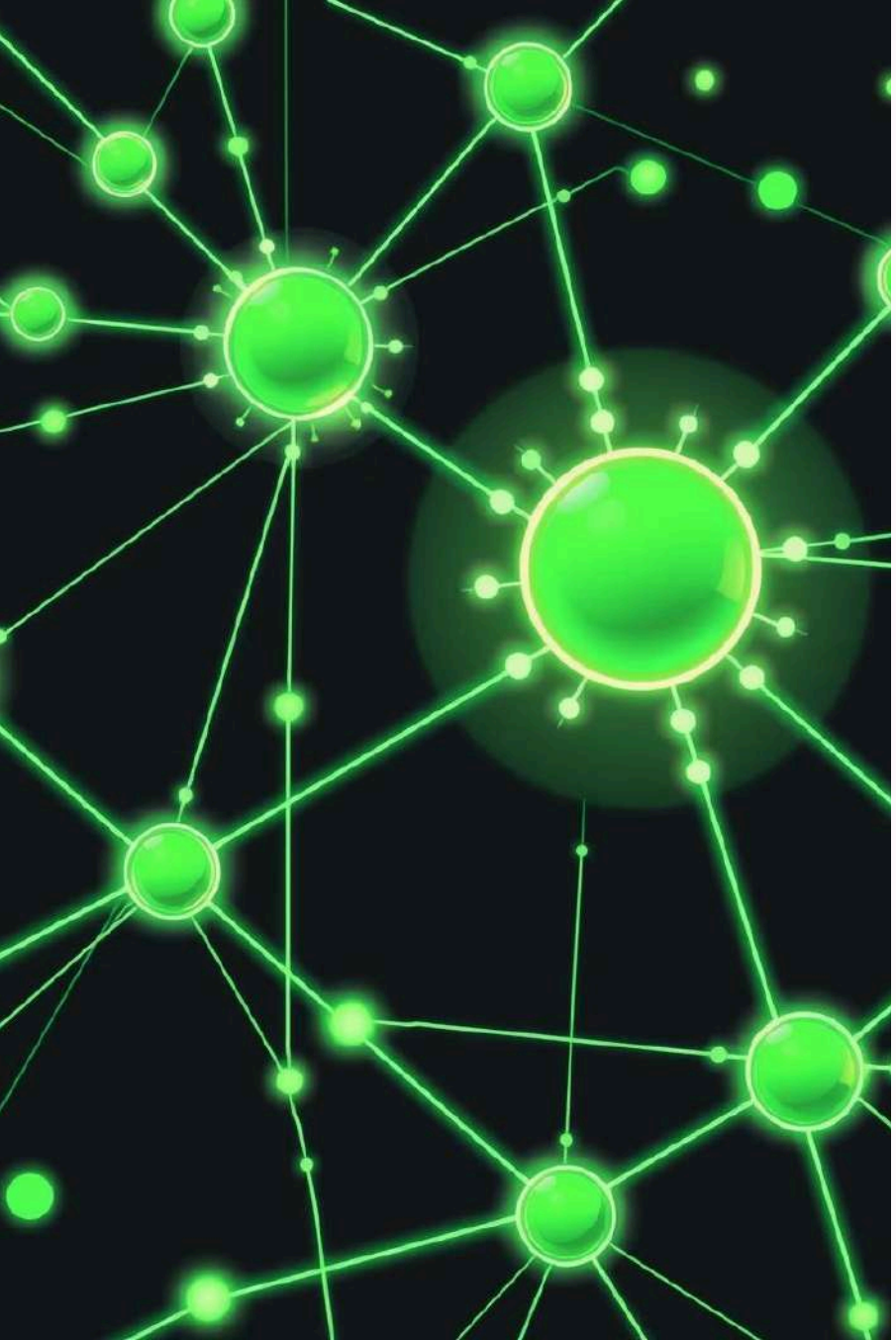


Introduction aux Fondamentaux de JavaScript

Présenté par Chaimae El Khayat et Latifa

Chapitre 1 : Les bases

Qu'est-ce que JavaScript ?

JavaScript est un **langage de programmation léger et interprété**, principalement connu comme le langage de script pour les **pages web**. Il permet de créer des **expériences interactives et dynamiques** pour l'utilisateur, allant au-delà des simples pages statiques.

C'est l'un des trois piliers du développement web, avec HTML pour la structure et CSS pour le style.

Pourquoi apprendre JavaScript ?

Omniprésence Web

JavaScript est le **seul langage de programmation qui fonctionne nativement dans tous les navigateurs web**. Il est indispensable pour toute interaction côté client.

Développement Full-Stack

Grâce à Node.js, JavaScript peut être utilisé pour le **développement côté serveur**, permettant de construire des applications web complètes avec un seul langage.

Diversité d'Applications

Au-delà du web, JavaScript est utilisé pour les **applications mobiles** (React Native), les **applications de bureau** (Electron) et même les jeux.

Opportunités de Carrière

La maîtrise de JavaScript est une **compétence très recherchée** dans l'industrie technologique, ouvrant de nombreuses portes professionnelles.

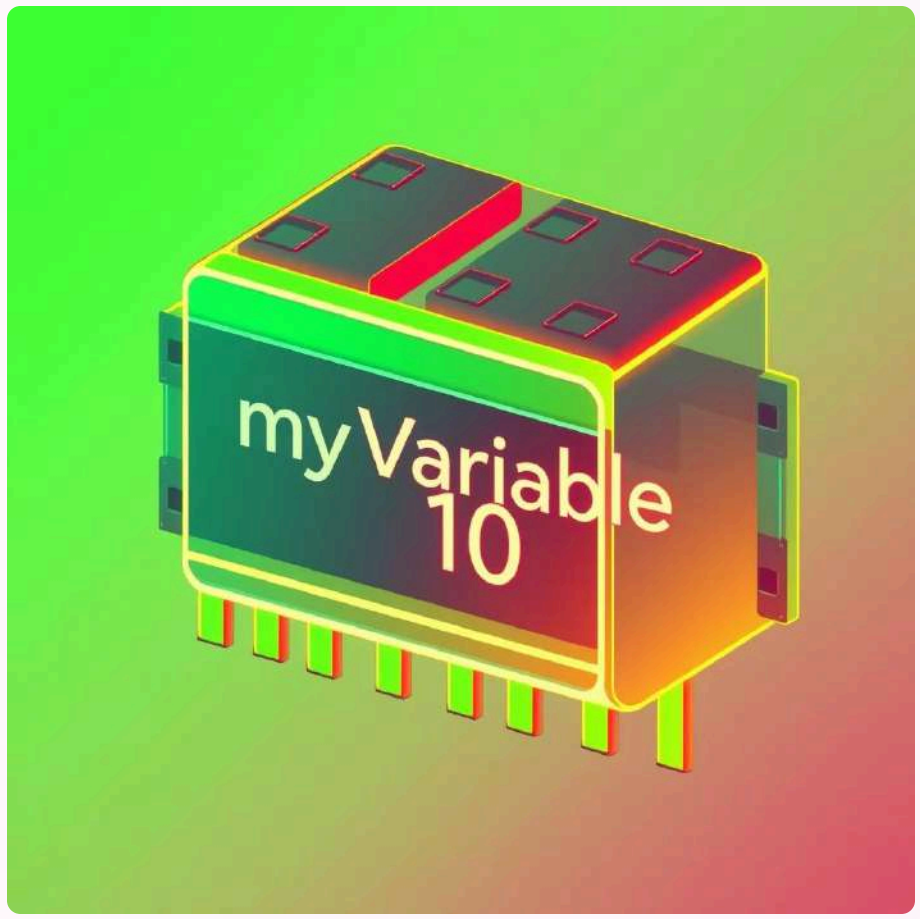
Chapitre 2 : Les outils fondamentaux

Les Variables : Stockez vos données !

Une **variable** est un conteneur nommé pour stocker des informations que vous souhaitez utiliser ou modifier ultérieurement dans votre programme.

Pourquoi les utiliser ?

- Pour **stocker des valeurs** (nombres, textes, objets, etc.).
- Pour rendre votre code **plus lisible et maintenable**.
- Pour **réutiliser des données** sans les réécrire.



Exemples de déclaration de variables :

```
// Déclaration avec 'var' (ancien)
var nom = "Alice";

// Déclaration avec 'let' (moderne, peut être réassigné)
let age = 30;
age = 31; // Valide

// Déclaration avec 'const' (moderne, ne peut pas être réassigné)
const PI = 3.14;
// PI = 3.14159; // Erreur !
```

Les conditions

Les **conditions** permettent à votre programme de prendre des decisions basées sur la **vérité ou la fausseté d'une expression**. C'est le "si ceci, alors cela" de la programmation.

Pourquoi les utiliser ?

- Pour exécuter du code **différemment selon les situations**.
- Pour gérer les **différents chemins logiques** d'une application.

les instructions des conditionnes

• Instruction if

L'instruction if permet d'exécuter un bloc de code seulement si une condition donnée est vraie.

Exemple:

Javascript Copier

```javascript
let age = 18;

if (age >= 18) {
  console.log("Vous êtes majeur.");
}
```

Instruction if...else

L'instruction if...else permet d'exécuter une alternative si la condition est fausse.

Exemple :

Javascript

Copier

```
if (heure < 12) {
    console.log("Bonjour !");
} else if (heure < 18) {
    console.log("Bon après-midi !");
} else {
    console.log("Bonsoir !");
}
```

➡️ Cela permet de **gérer plusieurs cas successifs.**

√ Opérateurs de comparaison

Les opérateurs de comparaison permettent de comparer des valeurs.

Opérateur	Description	Exemple (retourne true)
==	Égalité (valeur)	5 == "5"
===	Égalité stricte (valeur et type)	5 === 5
!=	Différent (valeur)	5 != "4"
!==	Différent strict (valeur et type)	5 !== "5"
>	Supérieur	10 > 5
<	Inférieur	3 < 5
>=	Supérieur ou égal	10 >= 10
<=	Inférieur ou égal	5 <= 5

√ Opérateurs logiques

Les opérateurs logiques permettent de combiner plusieurs conditions.

Opérateur	Description	Exemple (retourne true = vrai)
&&	ET (toutes les conditions doivent être vraies)	(5 > 3) && (10 > 5)
||	OU (au moins une condition doit être vraie)	(5 > 10) || (10 > 5)
!	NON (inverse la condition)	! (5 > 10)

Les Boucles

Les boucles sont utilisées pour répéter un bloc de code un certain nombre de fois ou tant qu'une condition spécifique est vraie. Elles sont essentielles pour automatiser les tâches répétitives.

Pourquoi les utiliser ?

✅ **Gagner du temps**

✅ **Réduire le code répété**

✅ **Automatiser des tâches**

✅ **Organiser la logique du programme**

Les principaux Types de Boucles :For, While,

Do... While

La boucle for

La boucle *for* est idéale lorsque vous connaissez à l'avance le nombre d'itérations nécessaires.

syntaxe générale :

for (initialisation; condition; incrémentation) {

// Code à exécuter

}

Javascript Copier

```
for (let i = 0; i < 5; i++) {
  console.log("Compteur : " + i); // Affiche 0, 1, 2, 3, 4
}
```

La boucle while

La boucle ***while*** est utilisée lorsqu'on **ne connaît pas à l'avance** le nombre d'itérations. Elle continue de s'exécuter tant qu'une condition donnée est vraie.>

Syntaxe :

while (condition) {

// Code à exécuter

}

Javascript Copier

```
let j = 0;
while (j < 3) {
  console.log("Tours de boucle : " + j); // Affiche 0, 1, 2
  j++;
}
```

La boucle do while

La boucle *do…while* fonctionne comme une boucle *while* , à une différence près : le code est **exécuté au moins une fois** , même si la condition est fausse dès le départ.

Syntaxe :

do {

// Code à exécuter

} while (condition);

Javascript Copier

```
let compteur = 0;

do {
  console.log("Tour numéro : " + compteur);
  compteur++;
} while (compteur < 3);
```

Conclusion

Maîtrisez JavaScript, ouvrez les portes du web !

Points clés à retenir :

- JavaScript rend le web **interactif et dynamique**.
- Les **variables, conditions et boucles** sont les briques de base.
- Les **fonctions** organisent le code, le **DOM** interagit avec la page.
- C'est un langage **polyvalent** avec un avenir prometteur.

Prochaines étapes :

- Pratiquez régulièrement avec des **mini-projets**.
- Explorez les concepts avancés : **objets, tableaux, POO**.
- Découvrez les **frameworks** et **bibliothèques** (React, Vue, Angular).
- Construisez, échouez, apprenez, **répétez** !

Merci de votre attention ! Des questions ?