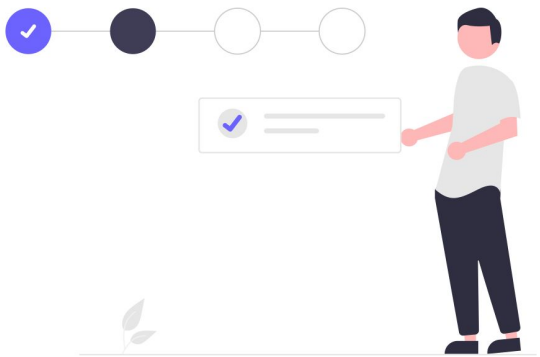
Javascript

Objectifs

1. **Découvrir JS**
2. **Ajouter du Javascript** à un site web
3. **Afficher des messages** dans la console du navigateur
4. Faire des **calculs** et des **comparaisons simples**

Découvrir JS



On a précédemment vu :

On a précédemment vu :

- Que l'**HTML** est principalement utilisé pour créer **la structure** d'une page web

On a précédemment vu :

- Que l'**HTML** est principalement utilisé pour créer **la structure** d'une page web
- Le **CSS** est le langage qu'on utilise pour **styliser** nos pages

On a précédemment vu :

- Que l'**HTML** est principalement utilisé pour créer **la structure** d'une page web
- Le **CSS** est le langage qu'on utilise pour **styliser** nos pages

Mais du coup, quel est le rôle du JS ici ?

On a précédemment vu :

- Que l'**HTML** est principalement utilisé pour créer **la structure** d'une page web
- Le **CSS** est le langage qu'on utilise pour **styliser** nos pages

Mais du coup, quel est le rôle du JS ici ?

C'est un langage de programmation qui est utile pour **ajouter de l'interactivité dans nos pages web.**

On a précédemment vu :

- Que l'**HTML** est principalement utilisé pour créer **la structure** d'une page web
- Le **CSS** est le langage qu'on utilise pour **styliser** nos pages

Mais du coup, quel est le rôle du JS ici ?

C'est un langage de programmation qui est utile pour **ajouter de l'interactivité dans nos pages web.**

Historiquement, Javascript a été conçu pour être utilisé côté frontend (dans le navigateur donc).

Historiquement, Javascript a été conçu pour être utilisé côté frontend (dans le navigateur donc).

Mais depuis **2009**, il est possible de créer du code **backend** (côté serveur donc) avec **NodeJS** ! 😎

Historiquement, Javascript a été conçu pour être utilisé côté frontend (dans le navigateur donc).

Mais depuis **2009**, il est possible de créer du code **backend** (côté serveur donc) avec **NodeJS** ! 😎

Javascript est aujourd'hui en version 7

JS version 1 : Créée en 10 jours en 1995 avec pour objectif principal d'aider java pour créer des pages web plus dynamiques

Historiquement, Javascript a été conçu pour être utilisé côté frontend (dans le navigateur donc).

Mais depuis **2009**, il est possible de créer du code **backend** (côté serveur donc) avec **NodeJS** ! 😎

Javascript est aujourd'hui en version 7

JS version 1 : Créée en 10 jours en 1995 avec pour objectif principal d'aider java pour créer des pages web plus dynamiques

1997 : Javascript se propage rapidement et sa popularité nécessite une standardisation : **ECMAscript**. Son objectif : développer Javascript avec des normes, de la maintenabilité et de nouvelles fonctionnalités au fur et à mesure.

Historiquement, Javascript a été conçu pour être utilisé côté frontend (dans le navigateur donc).

Mais depuis **2009**, il est possible de créer du code **backend** (côté serveur donc) avec **NodeJS** ! 😎

Javascript est aujourd'hui en version 7

JS version 1 : Créée en 10 jours en 1995 avec pour objectif principal d'aider java pour créer des pages web plus dynamiques

1997 : Javascript se propage rapidement et sa popularité nécessite une standardisation : **ECMAscript**. Son objectif : développer Javascript avec des normes, de la maintenabilité et de nouvelles fonctionnalités au fur et à mesure.

2005 : **AJAX** débarque et permet d'ajouter la gestion des données asynchrones avec Javascript. C'est le début de l'explosion de javascript et de ses multitudes de librairies, outils et frameworks.

Historiquement, Javascript a été conçu pour être utilisé côté frontend (dans le navigateur donc).

Mais depuis **2009**, il est possible de créer du code **backend** (côté serveur donc) avec **NodeJS** ! 😎

Javascript est aujourd'hui en version 7

JS version 1 : Créée en 10 jours en 1995 avec pour objectif principal d'aider java pour créer des pages web plus dynamiques

1997 : Javascript se propage rapidement et sa popularité nécessite une standardisation : **ECMAscript**. Son objectif : développer Javascript avec des normes, de la maintenabilité et de nouvelles fonctionnalités au fur et à mesure.

2005 : **AJAX** débarque et permet d'ajouter la gestion des données asynchrones avec Javascript. C'est le début de l'explosion de javascript et de ses multitudes de librairies, outils et frameworks.

2006 : arrivée de **Jquery**, une grosse librairie Javascript qui permet de manipuler facilement les données asynchrones, le HTML et le DOM

2009 : arrivée de NodeJs : une plateforme open-source d'environnement de développement en Javascript pour exécuter du Javascript côté serveur. La possibilité de voir du JS en frontend et backend née alors…

2009 : arrivée de NodeJs : une plateforme open-source d'environnement de développement en Javascript pour exécuter du Javascript côté serveur. La possibilité de voir du JS en frontend et backend née alors…

2010 à maintenant : arrivée et développement des gros **frameworks** Javascript

2009 : arrivée de NodeJs : une plateforme open-source d'environnement de développement en Javascript pour exécuter du Javascript côté serveur. La possibilité de voir du JS en frontend et backend née alors…

2010 à maintenant : arrivée et développement des gros **frameworks** Javascript

Pendant tout ce temps, ECMAscript continue de se développer et d'offrir de nouvelles fonctionnalités et d'étoffer sa standardisation

Comment fonctionne Javascript ?

Javascript est ce qu'on appelle **un langage haut-niveau**. Cela signifie que JS est éloigné du langage machine.

Comment fonctionne Javascript ?

Javascript est ce qu'on appelle **un langage haut-niveau**. Cela signifie que JS est éloigné du langage machine.

Tout le code que l'on exécute sur notre machine est transformé en code binaire (**0 et 1**). Javascript est plus proche du langage humain que du langage machine.

Comment fonctionne Javascript ?

Javascript est ce qu'on appelle **un langage haut-niveau**. Cela signifie que JS est éloigné du langage machine.

Tout le code que l'on exécute sur notre machine est transformé en code binaire (**0 et 1**). Javascript est plus proche du langage humain que du langage machine.

Un langage de programmation seul est quelque peu inutile, c'est comme parler un langage que personne ne comprend.

Comment fonctionne Javascript ?

Javascript est ce qu'on appelle **un langage haut-niveau**. Cela signifie que JS est éloigné du langage machine.

Tout le code que l'on exécute sur notre machine est transformé en code binaire (**0 et 1**). Javascript est plus proche du langage humain que du langage machine.

Un langage de programmation seul est quelque peu inutile, c'est comme parler un langage que personne ne comprend.

Mais, comment, JS est transformé en **langage machine** ? Le navigateur a le pouvoir de traduire le code JS en langage machine avec ce qu'on appelle un **Moteur JavaScript (JS Engine).**

Comment fonctionne Javascript ?

Javascript est ce qu'on appelle **un langage haut-niveau**. Cela signifie que JS est éloigné du langage machine.

Tout le code que l'on exécute sur notre machine est transformé en code binaire (**0 et 1**). Javascript est plus proche du langage humain que du langage machine.

Un langage de programmation seul est quelque peu inutile, c'est comme parler un langage que personne ne comprend.

Mais, comment, JS est transformé en langage machine ? Le navigateur a le pouvoir de traduire le code JS en langage machine avec ce qu'on appelle un Moteur JavaScript (JS Engine).

Ce moteur (par exemple le moteur **V8** dans Google Chrome) va "lire" notre JS et le traduire, ce qui fait que notre ordinateur va le comprendre.

Et Java dans tout ça ?

Attention : Bien que leurs noms soient similaires, **Javascript est totalement différent de Java.**

Et Java dans tout ça ?

Attention : Bien que leurs noms soient similaires, **Javascript est totalement différent de Java.**

JS est appelé **Java**Script car lorsqu'il a été créé, **Java était populaire**. Les créateurs ont par conséquent pensé qu'ajouter Java dans son nom le rendrait plus accessible.

Et Java dans tout ça ?

Attention : Bien que leurs noms soient similaires, **Javascript est totalement différent de Java.**

JS est appelé **Java**Script car lorsqu'il a été créé, **Java était populaire**. Les créateurs ont par conséquent pensé qu'ajouter Java dans son nom le rendrait plus accessible.

C'est tout ☺️

Et Java dans tout ça ?

Attention : Bien que leurs noms soient similaires, **Javascript est totalement différent de Java.**

JS est appelé **Java**Script car lorsqu'il a été créé, **Java était populaire**. Les créateurs ont par conséquent pensé qu'ajouter Java dans son nom le rendrait plus accessible.

C'est tout ☺️

Java # Javascript

Erreur à ne jamais commettre !

Et Java dans tout ça ?

Attention : Bien que leurs noms soient similaires, **Javascript est totalement différent de Java.**

JS est appelé **Java**Script car lorsqu'il a été créé, **Java était populaire**. Les créateurs ont par conséquent pensé qu'ajouter Java dans son nom le rendrait plus accessible.

C'est tout ☺️

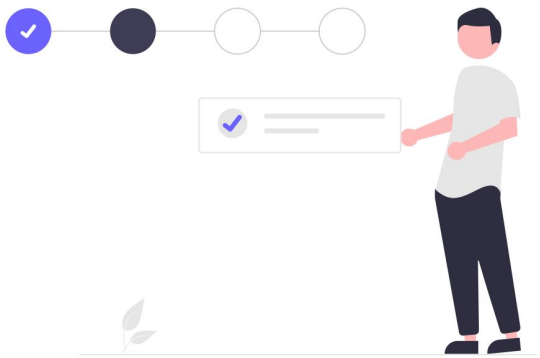
Java # Javascript

Erreur à ne jamais commettre !

SVP ☺️ c'est important ☺️ merci ☺️

JS

Ajouter du JS à une page web



Ajouter un script JS à une page HTML

Afin de commencer à écrire du JS, la première chose dont nous avons besoin est une page HTML.

Ajouter un script JS à une page HTML

Afin de commencer à écrire du JS, la première chose dont nous avons besoin est une page HTML.

On ajoute la balise **<script>** juste avant de fermer la balise **</body>.**

Ajouter un script JS à une page HTML

Afin de commencer à écrire du JS, la première chose dont nous avons besoin est une page HTML.

On ajoute la balise **<script>** juste avant de fermer la balise **</body>.**

Dans cette balise, on précise où se trouve notre fichier **script.js** à l'instar de la balise link avec le **style.css**

JS

Ajouter un script JS à une page HTML

Afin de commencer à écrire du JS, la première chose dont nous avons besoin est une page HTML.

On ajoute la balise **<script>** juste avant de fermer la balise **</body>.**

Dans cette balise, on précise où se trouve notre fichier **script.js** à l'instar de la balise link avec le **style.css**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=dev
    <title>Document</title>
</head>
<body>
    <script src="script.js"></script>
</body>
</html>
```

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

C'est un peu le lorem ipsum des langages de programmation

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

C'est un peu le lorem ipsum des langages de programmation

Le navigateur possède une console javascript qui nous permet de faire tout un tas de choses, dont afficher des messages.

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

C'est un peu le lorem ipsum des langages de programmation

Le navigateur possède une console javascript qui nous permet de faire tout un tas de choses, dont afficher des messages.

Pour commencer, nous allons donc afficher “hello world” dans cette console.

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

C'est un peu le lorem ipsum des langages de programmation

Le navigateur possède une console javascript qui nous permet de faire tout un tas de choses, dont afficher des messages.

Pour commencer, nous allons donc afficher "hello world" dans cette console.

Pour cela, on écrit la ligne suivante dans notre fichier script.js :

JS

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

C'est un peu le lorem ipsum des langages de programmation

Le navigateur possède une console javascript qui nous permet de faire tout un tas de choses, dont afficher des messages.

Pour commencer, nous allons donc afficher "hello world" dans cette console.

Pour cela, on écrit la ligne suivante dans notre fichier script.js :

```
console.log("Hello World");
```

Première ligne de code

De manière générale, et qu'importe le langage de programmation utilisé, la première ligne de code que l'on écrit permet d'afficher "**hello world**".

C'est un peu le lorem ipsum des langages de programmation

Le navigateur possède une console javascript qui nous permet de faire tout un tas de choses, dont afficher des messages.

Pour commencer, nous allons donc afficher "hello world" dans cette console.

Pour cela, on écrit la ligne suivante dans notre fichier script.js :

```
console.log("Hello World");
```

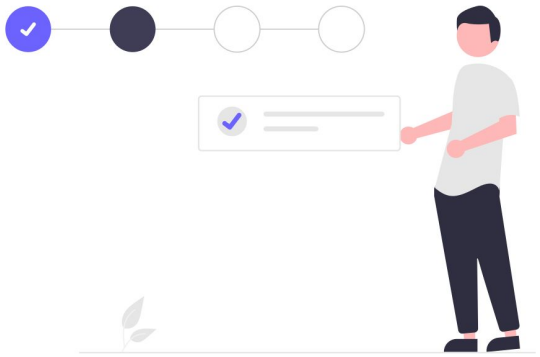
En ouvrant notre console, on obtient :

```
Hello World                                                    script.js:1
```


JS

JS

Syntaxe



Le code Javascript **se termine avec un point-virgule (;)**

Le code Javascript **se termine avec un point-virgule (;)**

Le point-virgule indique la fin de la commande.

Le code Javascript **se termine avec un point-virgule (;)**

Le point-virgule indique la fin de la commande.

Si on oublie ce point-virgule, Javascript va automatiquement l'ajouter pour chaque retour à la ligne grâce au mécanisme ASI *(Automatic Semicolon Insertion)*. Mais il faut faire attention : en se fiant à ce mécanisme, on peut rencontrer des erreurs qui seront alors difficile à repérer.

String

Les *strings* en Javascript sont des chaînes de caractères. Elles doivent être entourées de guillemets *(quotes* 🇬🇧*)* (simple ou double)

JS

String

Les *strings* en Javascript sont des chaînes de caractères. Elles doivent être entourées de guillemets *(quotes* 🇬🇧*)* (simple ou double)

```
console.log("Hello, world");
console.log('Hello, world');
```

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

Javascript, tout comme le navigateur, sont livrés avec un ensemble de fonctions intégrées. **La fonction console.log** est l'une d'entre elles.

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

Javascript, tout comme le navigateur, sont livrés avec un ensemble de fonctions intégrées. **La fonction console.log** est l'une d'entre elles.

Cette fonction console.log() nous permet d'écrire des choses dans la console :

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

Javascript, tout comme le navigateur, sont livrés avec un ensemble de fonctions intégrées. **La fonction console.log** est l'une d'entre elles.

Cette fonction console.log() nous permet d'écrire des choses dans la console :

- console (appeler la console)

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

Javascript, tout comme le navigateur, sont livrés avec un ensemble de fonctions intégrées. **La fonction console.log** est l'une d'entre elles.

Cette fonction console.log() nous permet d'écrire des choses dans la console :

- console (appeler la console)
- log (afficher/relever)

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

Javascript, tout comme le navigateur, sont livrés avec un ensemble de fonctions intégrées. **La fonction console.log** est l'une d'entre elles.

Cette fonction console.log() nous permet d'écrire des choses dans la console :

- console (appeler la console)
- log (afficher/relever)

Pour notre première approche avec **console.log,** nous allons afficher un message, c'est-à-dire une chaîne de caractères…. ! Pour cela, **on passe notre message entre les parenthèses de notre fonctions.**

console.log()

console.log() est une **fonction Javascript** (on parlera un peu plus des fonctions dans un futur proche). Pour le moment, gardons en tête qu'une fonction en Javascript est un bout de code réutilisable que l'on peut appeler pour effectuer certaines opérations.

Javascript, tout comme le navigateur, sont livrés avec un ensemble de fonctions intégrées. **La fonction console.log** est l'une d'entre elles.

Cette fonction console.log() nous permet d'écrire des choses dans la console :

- console (appeler la console)
- log (afficher/relever)

Pour notre première approche avec **console.log,** nous allons afficher un message, c'est-à-dire une chaîne de caractères…. ! Pour cela, **on passe notre message entre les parenthèses de notre fonctions.**

```
console.log("coucou hibou coucou hibou");
```

```
coucou hibou coucou hibou                                    script.js:1
```

console.log() : différents types de messages

Il y a différents types de message de console.

console.log() : différents types de messages

Il y a différents types de message de console.

console.log() est celui qu'on va utiliser le plus souvent : il affiche le message de la manière la plus simple.

console.log() : différents types de messages

Il y a différents types de message de console.

console.log() est celui qu'on va utiliser le plus souvent : il affiche le message de la manière la plus simple.

Mais il y a aussi :

console.log() : différents types de messages

Il y a différents types de message de console.

console.log() est celui qu'on va utiliser le plus souvent : il affiche le message de la manière la plus simple.

Mais il y a aussi :

- console.error() ⇒ pour afficher des erreurs
- console.warn() ⇒ pour afficher des avertissements

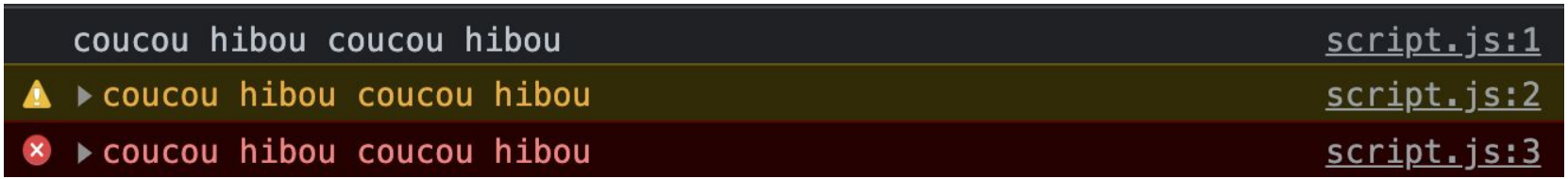
console.log() : différents types de messages

Il y a différents types de message de console.

console.log() est celui qu'on va utiliser le plus souvent : il affiche le message de la manière la plus simple.

Mais il y a aussi :

- console.error() ⇒ pour afficher des erreurs
- console.warn() ⇒ pour afficher des avertissements



console.log() : erreurs

S’il y a une erreur dans l’écriture du code, un message d'erreur apparaît dans la console qui permet de comprendre ce qu'il se passe.

console.log() : erreurs

S'il y a une erreur dans l'écriture du code, un message d'erreur apparaît dans la console qui permet de comprendre ce qu'il se passe.

Exemple : Si j'écris **console.loge** au lieu de **console.log**

console.log() : erreurs

S'il y a une erreur dans l'écriture du code, un message d'erreur apparaît dans la console qui permet de comprendre ce qu'il se passe.

Exemple : Si j'écris **console.loge** au lieu de **console.log**

```
console.loge("zé barti pour le test");
```

```
▸Uncaught TypeError: console.loge is not a function          script.js:1
    at script.js:1
```

Opérations arithmétiques

Javascript nous permet d'écrire des **opérations arithmétiques.**

Opérations arithmétiques

Javascript nous permet d'écrire des **opérations arithmétiques.**

Faisons un peu de maths avec Javascript (pas de panique, ça devrait le faire)


```
1 + 1
// 2

2 - 2
// 0

3 * 2
// 6

9 / 3
// 3

5 % 3
// 2

6 % 2
// 0
```

Incréments / Décréments

Il est possible d'incrémenter ou de décrémenter très simplement une variable avec JS.

```
let age = 23;

// Incrémenter :
age = age + 1;
age += 1;
age++


// Décrémenter :
age = age - 1;
age -= 1;
age --
```

Comparaison

On peut aussi utiliser Javascript pour **comparer des valeurs**

Comparaison

On peut aussi utiliser Javascript pour **comparer des valeurs**

Javascript donnera une réponse : **true ou false**. Ces valeurs sont appelées **booléens.**

Comparaison

On peut aussi utiliser Javascript pour **comparer des valeurs**

Javascript donnera une réponse : **true ou false**. Ces valeurs sont appelées **booléens.**

- **Valeur égale et type égal**

Comparaison

On peut aussi utiliser Javascript pour **comparer des valeurs**

Javascript donnera une réponse : **true ou false**. Ces valeurs sont appelées **booléens.**

- **Valeur égale et type égal**
  - Dans ce cas, on va regarder si les valeurs sont strictement égales.

Comparaison

On peut aussi utiliser Javascript pour **comparer des valeurs**

Javascript donnera une réponse : **true ou false**. Ces valeurs sont appelées **booléens.**

- **Valeur égale et type égal**
  - Dans ce cas, on va regarder si les valeurs sont strictement égales.
  - Cela signifie que les valeurs et types sont les mêmes.

Comparaison

On peut aussi utiliser Javascript pour **comparer des valeurs**

Javascript donnera une réponse : **true ou false**. Ces valeurs sont appelées **booléens.**

- **Valeur égale et type égal**
  - Dans ce cas, on va regarder si les valeurs sont strictement égales.
  - Cela signifie que les valeurs et types sont les mêmes.

Comparaison

- **Valeur égale**
  - Dans ce cas, on regarde seulement si les valeurs sont égales.

```
1 == 1; // true ✅
1 == "1"; // true ✅
```

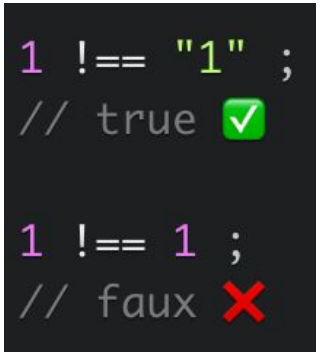
Comparaison

- **Différentes valeurs**
  - Dans ce cas, on regarde si les valeurs sont différentes.

```
1 != 2; // true ✅
1 != "1"; // false ❌
```

Comparaison

- **Différentes valeurs ou type différent**
  - Dans ce cas, nous vérifions si les valeurs ou le type sont différents.



Comparaison

- **Supérieur à, supérieur ou égal**
  - Ici, on vérifie que la valeur est supérieure à une autre
  - en ajoutant le symbole d'égalité **=** juste après le **>,** on peut vérifier si la valeur est supérieure ou égale.

```
2 > 1; // true ✅
2 >= 2; // true ✅
```

Comparaison

- **Inférieur à, inférieur ou égal**
  - Ça fonctionne aussi dans l'autre sens avec le symbole inférieur **<**

```
2 < 3; // true ✅
2 <= 2; // true ✅
```

Pratiquons !