Les types de données en JavaScript

> Les types de données définissent la nature des valeurs que les variables peuvent stocker.
>
> Ils permettent à JavaScript de manipuler différents types d'informations comme des nombres, des chaînes de caractères ou des objets.

1. Les types primitifs

> Les types primitifs sont des types de données simples et immuables.

1.1. String (chaîne de caractères)

> Utilisé pour représenter du texte. Les chaînes doivent être entourées de guillemets simples ou doubles.

Exemple :

```javascript
const prenom = "Alice";
const message = 'Bienvenue!';
```

1.2. Number (nombre)

> Utilisé pour représenter des nombres, qu'ils soient entiers ou décimaux.

Exemple :

```javascript
const age = 30;
const prix = 19.99;
```

1.3. BigInt (grands entiers)

> Utilisé pour représenter des nombres entiers très grands qui dépassent la capacité de `Number`.
>
> Il est créé en ajoutant `n` à la fin du nombre.

Exemple :

```javascript
const grandNombre = 123456789012345678901234567890123456789012345678901234567890n;
```

1.4. Boolean (booléen)

> Représente une valeur vraie ou fausse : `true` ou `false`.

Exemple :

```javascript
const estActif = true;
const estConnecte = false;
```

1.5. Undefined

> Une variable déclarée mais à laquelle aucune valeur n'a été assignée porte la valeur par défaut `undefined`.

Exemple :

```javascript
let resultat;
console.log(resultat); // undefined
```

1.6. Null

> Représente une absence volontaire de valeur. Contrairement à `undefined`, il est assigné manuellement par le programmeur.

Exemple :

```javascript
const personne = null;
```

1.7. Symbol

> Type **unique** introduit avec ES6, utilisé pour créer des **identifiants uniques**.

Exemple :

```javascript
const idUnique = Symbol('id');
```

2. Les types complexes

> Les **types complexes** permettent de stocker des données plus **structurées** ou **variées**.

2.1. Object (objet)

> Un **objet** est une **collection** de paires **clé-valeur**. Il est utilisé pour représenter des **entités plus complexes**.

Exemple :

```javascript
const utilisateur = {
  nom: "Jean",
  age: 25,
  estActif: true
};
```

2.2. Array (tableau)

> Un **tableau** est une **collection ordonnée** de valeurs, accessibles par leur **position** (index).

Exemple :

```javascript
const fruits = ["pomme", "orange", "banane"];
```

2.3. Function

> En JavaScript, les fonctions sont des objets de type spécial capables d'exécuter une tâche.

Exemple :

```javascript
function saluer() {
  return "Bonjour!";
}
```

3. Conversion de types

> JavaScript permet de convertir des données d'un type à un autre si nécessaire.

3.1. Conversion implicite

> JavaScript convertit automatiquement les données en fonction du contexte.

Exemple :

```javascript
const resultat = "10" + 5; // "105" : le nombre est converti en chaîne
```

3.2. Conversion explicite

> Utilisez des fonctions pour convertir manuellement les données.

Exemple :

```javascript
const nombre = Number("45"); // Convertit une chaîne en nombre
const texte = String(123);   // Convertit un nombre en chaîne
```

4. Détection des types

> Pour connaître le type d'une variable, on utilise l'opérateur `typeof`.

Exemple :

```javascript
const test = "Bonjour";
console.log(typeof test); // "string"
```

4.1. Cas particulier : null

> Bien que `null` soit un type primitif, `typeof null` retourne "object".
>
> C'est une incohérence historique de JavaScript.

5. À RETENIR

- Types primitifs (simples et immuables) :
  - `String` : chaînes de caractères.
  - `Number` : nombres entiers/décimaux.
  - `BigInt` : grands entiers.
  - `Boolean` : vrai/faux.
  - `Undefined` : valeur non assignée.
  - `Null` : absence de valeur définie.
  - `Symbol` : identifiants uniques.
- Types complexes :
  - `Object` : collection de paires clé-valeur.
  - `Array` : liste ordonnée de valeurs.
  - `Function` : exécute des tâches spécifiques.
- Conversion :
  - Implicite : automatique selon le contexte.
  - Explicite : effectuée avec `Number()`, `String()`, etc.
- Détection des types :
  - utilisez `typeof` pour vérifier le type d'une variable.