Types de données JavaScript

Dans ce chapitre, vous découvrirez les types de données disponibles en JavaScript.

Types de données en JavaScript

Les types de données spécifient essentiellement quel type de données peut être stocké et manipulé dans un programme.

Il existe six types de données de base en JavaScript qui peuvent être divisés en trois catégories principales : les types de données primitifs (ou primaires), composites (ou de référence) et spéciaux . String, Number et Boolean sont des types de données primitifs. Object, Array et Function (qui sont tous des types d'objets) sont des types de données composites. Alors que Undefined et Null sont des types de données spéciaux.

Les types de données primitifs ne peuvent contenir qu'une seule valeur à la fois, tandis que les types de données composites peuvent contenir des collections de valeurs et des entités plus complexes. Discutons de chacun d'eux en détail.

Le type de données chaîne

Le type de données chaîne (*string*) est utilisé pour représenter des données textuelles (c'est-à-dire des séquences de caractères). Les chaînes sont créées à l'aide de guillemets simples ou doubles entourant un ou plusieurs caractères, comme indiqué ci-dessous :

```
let a = 'Hello World !';  // using single quotes
let b = "Hello World !";  // using double quotes
```

Vous pouvez inclure des guillemets à l'intérieur de la chaîne tant qu'ils ne correspondent pas aux guillemets englobants.

```
let a = "Prenons une tasse de café.";
let b = 'Il a dit "Hello" et est parti.';
let c = 'Je n\'ai pas dormi la nuit.';
```

Vous en apprendrez plus sur les chaînes dans le chapitre à venir.

Le type de données numérique

Le type de données *number* est utilisé pour représenter des nombres positifs ou négatifs avec ou sans décimale, ou des nombres écrits en notation exponentielle, par exemple 1,5e-4 (équivalent à $1{,}5 \times 10^{-4}$).

```
let a = 25;
let b = 80.5;
let c = 4.25e+6;
let d = 4.25e-6;
```

Vous en apprendrez plus sur les nombres dans le chapitre à venir

Le type de données booléen

Le type de données booléen ne peut contenir que deux valeurs : true ou false. Il est généralement utilisé pour stocker des valeurs telles que oui (true) ou non (false), activé (true) ou désactivé (false), etc., comme illustré ci-dessous

```
let isReading = true;
let isSleeping = false;
```

Les valeurs booléennes sont également le résultat de comparaisons dans un programme. L'exemple suivant compare deux variables et affiche le résultat dans une boîte de dialogue d'alerte :

```
let a = 2, b = 5, c = 10;
document.write(b > a) // Output: true
document.write(b > c) // Output: false
```

Vous en apprendrez plus sur les booleens dans le chapitre à venir

Le type de données indéfini

Le type de données indéfini ne peut avoir qu'une seule valeur, la valeur spéciale *undefined*. Si une variable a été déclarée, mais n'a pas reçu de valeur, elle aura la valeur *undefined*.

```
let a;
let b = "Hello World!“

alert(a) // Output: undefined
alert(b) // Output: Hello World!
```

Le type de données nul

Il s'agit d'un autre type de données spécial qui ne peut avoir qu'une seule valeur, la valeur *null*. Une valeur *null* signifie qu'il n'y a pas de valeur. Ce n'est pas équivalent à une chaîne vide ("") ou 0, c'est tout simplement rien.

Une variable peut être explicitement vidée de son contenu actuel en lui attribuant la valeur *null*.

```
<script>
    let a = null;
    document.write(a + "<br>"); // Print: null

    let b = "Hello World!"
    document.write(b + "<br>"); // Print: Hello World!

    b = null;
    document.write(b) // Print: null
</script>
```

Le type de données d'objet

object est un type de données complexe qui vous permet de stocker des collections de données.

Un objet contient des propriétés, définies comme une paire clé-valeur. Une clé de propriété (nom) est toujours une chaîne, mais la valeur peut être n'importe quel type de données, comme des chaînes, des nombres, des booléens ou des types de données complexes comme des tableaux, des fonctions et d'autres objets. Vous en apprendrez plus sur les objets dans les prochains chapitres.

L'exemple suivant vous montrera la manière la plus simple de créer un objet en JavaScript.

```
let emptyObject = {};
let person = {"name": "Clark", "surname": "Kent", "age": "36"};

// Pour une meilleure lecture
let car = {
    "modal": "BMW X3",
    "color": "white",
    "doors": 5
}
```

Vous pouvez omettre les guillemets autour du nom de la propriété si le nom est un nom JavaScript valide. Cela signifie que les guillemets sont obligatoires "first-name" mais facultatifs firstname. Ainsi, l'objet voiture dans l'exemple ci-dessus peut également être écrit comme suit :

```
let car = {
    modal: "BMW X3",
    color: "white",
    doors: 5
}
```

Vous en apprendrez plus sur les objets dans les chapitres à venir

Le type de données tableau

Un tableau est un type d'objet utilisé pour stocker plusieurs valeurs dans une seule variable. Chaque valeur (également appelée élément) dans un tableau a une position numérique, connue sous le nom d'index, et elle peut contenir des données de n'importe quel type de données, des nombres, des chaînes, des booléens, des fonctions, des objets et même d'autres tableaux. L'index du tableau commence à 0.

Le moyen le plus simple de créer un tableau consiste à spécifier les éléments du tableau sous forme de liste séparée par des virgules entre crochets, comme illustré dans l'exemple ci-dessous

```
<script>
    // Creating arrays
    let colors = ["Red", "Yellow", "Green", "Orange"];
    let cities = ["London", "Paris", "New York"];

    // Printing array values
    document.write(colors[0] + "<br>");   // Output: Red
    document.write(cities[2]);   // Output: New York
</script>
```

Vous en apprendrez plus sur les tableaux dans les chapitres à venir

Le type de données function

La fonction est un objet appelable qui exécute un bloc de code. Les fonctions étant des objets, il est donc possible de les affecter à des variables, comme le montre l'exemple ci-dessous :

```
<script>
    let greeting = function(){
        return "Hello World!";
    }
    document.write(typeof greeting) // Output: function
    document.write("<br>");
    document.write(greeting());     // Output: Hello World!
</script>
```

En fait, les fonctions peuvent être utilisées à n'importe quel endroit où n'importe quelle autre valeur peut être utilisée. Les fonctions peuvent être stockées dans des variables, des objets et des tableaux. Les fonctions peuvent être transmises en tant qu'arguments à d'autres fonctions et les fonctions peuvent être renvoyées à partir de fonctions. Considérez la fonction suivante :

```
<script>
function createGreeting(name){
    return "Hello, " + name;
}
function displayGreeting(greetingFunction, userName){
    return greetingFunction(userName);
}

var result = displayGreeting(createGreeting, "Peter");
document.write(result); // Output: Hello, Peter
</script>
```

Vous en apprendrez plus sur les fonctions dans les chapitres à venir

Le type d'opérateur

L' opérateur *typeof* peut être utilisé pour savoir quel type de données contient une variable ou un opérande. Il peut être utilisé avec ou sans parenthèses (typeof(x) ou typeof x).

L' opérateur *typeof* est particulièrement utile dans les situations où vous devez traiter différemment les valeurs de différents types, mais vous devez être très prudent, car il peut produire un résultat inattendu dans certains cas, comme le montre l'exemple suivant :

```
// Null
document.write(typeof Null + "<br>");  // Prints: "object"

// Objects
document.write(typeof {name: "John", age: 18} + "<br>");  // Prints: "object"
```

Comme vous pouvez le voir clairement dans l'exemple ci-dessus lorsque nous testons la valeur *null* à l'aide de l'opérateur *typeof* , il a renvoyé "*object*" au lieu de "*null*".

Il s'agit d'un bogue de longue date dans JavaScript, mais comme de nombreux codes sur le Web sont écrits autour de ce comportement, et donc le corriger créerait beaucoup plus de problèmes, donc l'idée de résoudre ce problème a été rejetée par le comité qui conçoit et maintient JavaScript .