Les variables

Description Sommaire

À propos de ce tutoriel

Les variables permettent de garder en mémoire une valeur lors de l'exécution d'un script. Elles sont essentielles au bon fonctionnement de nos algorithmes.

Écriture

Une variable peut être déclarée à l'aide du mot clef `const` ou `let` et ne doivent pas contenir de caractères spéciaux à l'exception des _.

```
const a = 2
const une_variable_plus_longue = "Ma super chaîne"
```

Contrairement à d'autres langages de programmation il n'est pas nécessaire de mettre un `;` à la fin des lignes en JavaScript (mais si vous voulez, vous pouvez ^^). L'interpréteur (ce qui va faire fonctionner notre code) va automatiquement détecter les fins de lignes et comprendre quand arrêter une instruction.

Une variable déclarée avec `const` ne peut se voir réassigner une valeur, par contre cela est possible avec une variable déclarée avec `let`.

```
let a = 2
a = "Je suis une chaîne maintenant !"
```

Cette différence permet d'éviter les erreurs en s'assurant, si on utilise `const` que la variable ne sera pas écrasée par la suite.

Les types de variables

Il est possible de stocker différents types d'information dans une variable.

Les nombres

```
const a = 2
const b = 3.4123
const c = -509
const d = 1/3
```

Attention dans le cas des réels, on utilise un `.` pour marquer la partie décimale. Il est aussi possible d'utiliser un `_` pour séparer les milliers visuellement (ce caractère est en fait ignoré par l'interpréteur).

```
const a = 10_000
```

Les chaînes de caractère

Les chaînes de caractères permettent de stocker des mots / phrases. On les entoure de `'`, de `"` ou de ```.

```
const a = "Salut les gens"
const b = 'Re-Salut les gens'
const c = `Re-Salut les gens`
```

Il n'y a pas de réelle différence entre les simples et doubles guillemets sauf si notre chaîne de caractère contient des ' ou ".

```
const a = "Ceci n'est pas problématique"
const b = 'Ceci n\'est pas problématique'
```

Les \``, ou backticks permettent de faire plus facilement de la concaténation avec le symbole ${}` mais permettent aussi d'avoir une chaîne de caractère sur plusieurs lignes.

```
const name = 'John'
const phrase = `Je m'appele ${name}`
const paragraph = `Je suis
sur plusieurs
lignes !!`
```

Les booléens

Les booléens permettent de stocker une information qui sera soit vraie, soit fausse.

```
const vrai = true
const je_suis_faux = false
```

Les tableaux

Les tableaux permettent de stocker une liste d'information. Cette liste peut contenir n'importe quel autre type de variable (un tableau peut même contenir un autre tableau).

```
const eleves = ['Jean', 'Marc', 'Marion']
const demo = [true, 10, 'Marc']
```

Ensuite il est possible de récupérer un élément dans un tableau en utilisant la notation `[i]` où i est un nombre représentant l'index de l'élément à récupérer (cet index commence par 0).

```
eleves[0] // Jean
eleves[2] // Marion
demo[1] // 10
demo[18] // undefined
```

Les objets

Les objets permettent de stocker des informations plus complexes qu'une simple liste. Pour le moment, vous pouvez imaginer les objets comme une liste avec des index nommés.

```
const eleve = {
   clef: 'valeur',
   nom: 'Jean',
   age: 18,
   notes: [10, 4, 18]
}
```

Dans un objet les "clefs" sont appelées des **propriétés**. Pour récupérer une valeur associée à une propriété il y a 2 notations possibles.

```
eleve.nom // Jean
eleve.notes // [10, 4, 18]
eleve.notes[1] // 4
// On peut aussi utiliser une notation proche de celle des tableaux
eleve['notes'] // [10, 4, 18]
```

De la même manière, les objets peuvent contenir des objets en valeur.

```
const eleve = {
    notes: {
        math: 18,
        francais: 14
    }
}
// Pour récupérer la note de math de l'élève on peut alors faire
eleve.notes.math // 18
eleves.nom // undefined
```

Types spéciaux

Enfin, lors de votre découverte du javascript vous allez rencontrer certaines variables qui correspondent à certains cas spécifiques.

```
undefined // quand on essaie d'accéder à une variable ou valeur inexistante
null // représente l'absence intentionnelle de toute valeur
NaN // 'not a number'
```

Typage faible

En javascript le typage est faible, suivant les opérations, les variables peuvent changer implicitement de type.

```
const a = '1'
const b = 1
a + b = '11'
// b est converti en chaîne de caractère implicitement
a * b = 1
// a est converti en nombre de manière implicite
"Salut" * 3 // NaN, Not a number
// Attention aux opérations qui n'ont pas de sens :)
"43" > 1000 // false, 1000 est converti en chaîne implicitement et il compare l'ordr
```

De manière générale on essaiera tant que possible de ne pas se reposer sur cette conversion implicite. Elle est plus souvent source de problèmes qu'autre chose.

Les commentaires

Vous pouvez commenter votre code en commençant une ligne par `//` ou en entourant votre code avec `/* */`.

```
// Ceci est un commentaire sur une ligne
/*
 Et ici un commentaire
 sur plusieurs lignes
*/
```