Les variables

Boris Rose

Table des matières

Les types de variables 1
Exemple 1
document 2
Le moteur javascript 2
document.getElementById() 2
Un Appel 2
L'élément HTML 3
La propriété innerHTML 3
L'intérêt des fonctions et des classes 4

Les types de variables

Il faut partir de l'idée qu'une variable est une entité dans laquelle vous allez stocker une valeur.

Cette **valeur** peut être de différent type : - un objet (type composite) - un tableau (array en anglais)(type composite) - une chaîne de caractère (idem)(type primitif) - un nombre (idem) (type primitif) - un booléan (type primitif) - un symbole (type primitif) - etc.

Exemple

```
const root = document.getElementById('root')
```

Ci-dessus j'ai initialisé une variable root et l'ai assigné en constante autrement dit sa valeur ne pourra pas changer.

Ici sa valeur c'est le membre de droit de l'assignation , l'assignation étant caractérisée par la présence d'un =

La valeur de la variable root est donc : document.getElementById(‘root’)

document

document est un objet qui représente l’intégralité de ce qu’une page dans votre navigateur web contient.

Vous regardez une page et donc vous regardez le contenu de document.

Or document est un objet que l’on peut utiliser dans les fichiers .js et .ts de façon native. Tout cela se fait grâce aux bibliothèques qui vont de pair avec la possibilité pour le navigateur de comprendre les fichiers que vous écrivez en .js.

Le moteur javascript

En effet, derrière l’interface que vous présente le navigateur web que vous utilisez, cela peut être Chrome, il y a beaucoup d’outils. L’un d’entre eux est le moteur javascript qui permet à votre navigateur de comprendre les fichiers .js et de les compiler correctement. Chrome utilise le moteur V8.

Ainsi document est un objet qui représente la page Internet que vous voyez dans votre navigateur.

document.getElementById()

A partir du moment où je dis que document est un objet, vous devez vous dire que vous pouvez utiliser la notation par point (dot notation en anglais) : - pour appeler ses méthodes comme dans cet exemple getElementById() - pour utiliser ses propriétés

Un Appel

```
    const root = document.getElementById('root')
```

Ci-dessous j’appelle la méthode getElementById() de l’objet document Un appel se caractérise par des () derrière le nom de la fonction / de la méthode, une méthode étant nécessairement une fonction alors qu’une fonction n’est pas nécessairement une méthode.

Donc j’appelle la méthode getElementById() à laquelle je passe une valeur, qui est un littéral de chaîne de caractères, littéral car vous voyez littéralement une chaîne de caractères sous vos yeux: ‘root’.

```
    const root = document.getElementById('root')
```

Ce que je dis finalement ci-dessus est la chose suivante:

Je veux que la variable root (constante) contienne l'élément HTML, présent sur le DOM (= sur la page) qui a comme id (id étant un attribut) : "root" (la valeur de l'attribut)

L'élément HTML

Or normalement en quelque part vous devriez précisément avoir cet élément présent

```
    <div id="root"></div>
```

Autrement dit dans la fichier .js vous récupérer cette élément dans une variable et ce, afin de pouvoir modifier son contenu HTML autrement dit son innerHTML

La propriété innerHTML

```
    const root = document.getElementById('root')
    root.innerHTML = "<h1>Home Page</h1>"
```

En assignant la chaîne de caractères (string en anglais) :

```
"<h1>Home Page</h1>"
```

à la propriété innerHTML de l'objet root cela revient à avoir maintenant sur le DOM la chose suivante:

```
    <div id="root">
        <h1>Home Page</h1>
    </div>
```

L'intérêt des fonctions et des classes

C'est bien beau mais je ne veux pas directement assigner “

Home Page

” comme valeur à la propriété innerHTML de l'objet root.

Je veux module le code et donc:

- utiliser des fonctions qui vont retourner une chaîne de caractère représentant une page entière (composant contextualisé)
- utiliser des fonctions acceptant des paramètres pour pouvoir les réutiliser avec différentes valeurs (les composants, enfants du composant contextualisé, réutilisables et décontextualisés)

En outre l'interface utilisateur c'est bien mais j'ai envie de permettre à mes pages de fonctionner correctement et donc je vais avoir besoin d'implémenter leur logique métier tout en rendant facile le fait de les tester, je vais donc utiliser des classes.

Lire le cours sur les fonctions et celui sur les classes dans le dossier latex ci-dessus