JavaScript avancé

Développement web front-end : React

Chris Chevalier - chevalier@chris-freelance.com

Les fonctions

Introduction au JavaScript

Chris Chevalier - chevalier@chris-freelance.com

Rappels sur les fonctions

- Une **fonction** correspond à un **sous-programme** permettant l'exécution d'un ensemble d'**instructions** d'un simple **appel** de celle-ci depuis le **programme principal**
- Elles permettent de **simplifier** le code et ainsi **diminuer la taille d'un programme**
- Notez que l'on parlera de **fonction récursive** lorsqu'une **fonction** s'appelle elle-même

Déclarer une fonction

- Pour être utilisée, une **fonction** doit d'abord être **déclarée**
- Le navigateur doit connaître son **nom**, ses **arguments** éventuels et ses **instructions** pour être capable de l'exécuter
- En **JS**, la **déclaration d'une fonction** se fait à l'aide du mot-clé `function` suivi d'un **nom**

```
13  function somme(a, b) {
14      return a + b;
15  }
```

- Le **nom d'une fonction** suit les mêmes **règles de nommage** que les **variables**
- Les **arguments** (ici, `a` et `b`) sont **facultatifs**

Appeler une fonction

- Pour **exécuter une fonction**, il nous faut **l'appeler**
- Pour ce faire, il suffit d'**écrire son nom** (en respectant la casse) suivi de **parenthèses** contenant ses **arguments** éventuels

```
somme(1, 2);
```

- Notez qu'en **JS**, le **point-virgule** `;` n'est pas obligatoire mais permet de préciser au navigateur qu'il s'agit de la **fin d'une instruction**
- **L'appel à une fonction** doit contenir le même nombre d'**arguments** initialisés à sa **déclaration**
- Une **fonction** doit toujours être **déclarée** avant d'être **appelée**

Les objets

JavaScript avancé

Les objets en quelques mots... (1/2)

- Un **objet** JS est une variable dans laquelle on peut stocker plusieurs couples clé-valeurs

```
var user = {
    name: "Chevalier",
    genre: "male",
    age: 29,
    isStudent: false,
    skills: ["html", "css", "javascript", "react"]
};
```

- 💡Comme un tableau, un objet peut contenir toute sorte de **type de donnée** : **chaînes de caractères**, **nombres**, **booléens**, **fonctions**, **tableaux** ou encore d'autres **objets**

Les objets en quelques mots… (2/2)

- Il est préférable d'utiliser un **objet** quand les valeurs stockées à l'intérieur sont de catégories différentes et descriptibles par une clé unique

- On préfèrera stocker dans un **tableau** des valeurs de la même catégorie, qui sera représentée par le nom du **tableau** (`fruits`, `legumes`, `users`, etc.)

Accéder aux valeurs d'un objet

- Il est possible d'accéder à une valeur spécifique d'un objet de 2 manières :
  - `console.log(user.name) // Affiche "Chevalier"`
  - `console.log(user["name"]) // Affiche également "Chevalier"`

- 💡Avec la seconde syntaxe, on peut utiliser une variable à la place du nom de la **clé**, ce qui peut être utile dans une **boucle** !

Les événements

Introduction au JavaScript

Chris Chevalier - chevalier@chris-freelance.com

Un événement en quelques mots…

- Un **événement** correspond à une **action** réalisée par l'**utilisateur** pour donner lieu à une **interactivité**
- Le **clic de souris** est probablement l'**événement** le plus répandu puisque c'est le seul que le **HTML** peut gérer
- Grâce au **JS**, il est possible d'associer des **fonctions** à des **événements** comme le **passage de la souris au dessus d'une zone**, le **changement de valeurs**, etc.
- Ce sont les **gestionnaires d'événements** qui permettent d'associer une **action** à un **évènement** - Voyons sa syntaxe dans un **lien hypertexte**

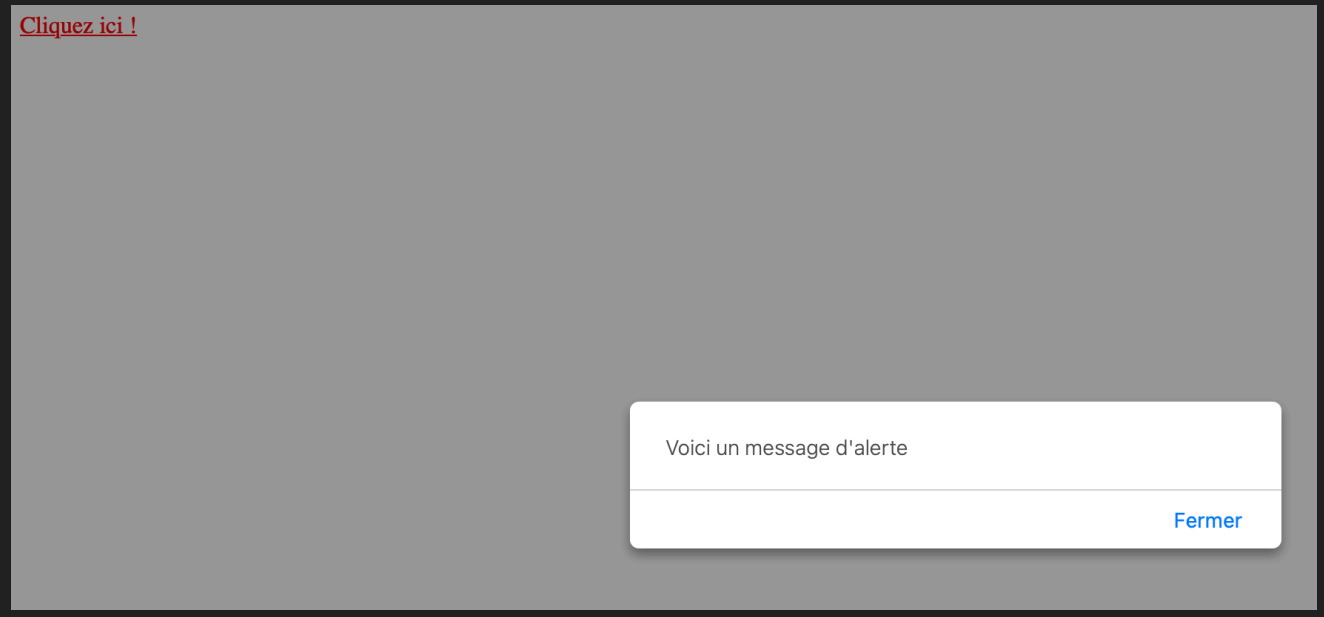
```
11  <body>
12      <a href="URL" onEvenement='Action_Javascript_ou_Fonction();'>Mon Lien</a>
13  </body>
```

Quelques événements JavaScript

- Voici une liste non exhaustive d'**événements** :
  - `onclick` : Lorsque l'utilisateur clique sur un élément associé à l'événement
  - `Dblclick` : Lorsque l'utilisateur double-clique sur un élément associé
  - `keydown` : Lorsque l'utilisateur appuie sur une touche de son clavier
  - `keypress` : Lorsque l'utilisateur maintient une touche de son clavier
  - `keyup` : Lorsque l'utilisateur relâche une touche de son clavier
  - `MouseOver` : Lorsque l'utilisateur positionne la souris sur un élément
  - `MouseOut` : Lorsque la souris quitte un élément
  - `Select` : Lorsque l'utilisateur sélectionne un texte
  - `Submit` : Lorsque l'utilisateur soumet un **formulaire HTML**

Un événement en image…

```
 9    <body>
10        <a href="javascript:;" onclick="window.alert('Voici un message d\'alerte');">
11            Cliquez ici !
12        </a>
13    </body>
```



Explications

- Ici, le **gestionnaire d'événement** `onclick` est inséré dans les **balises** `<a></a>`
- Pour signaler au navigateur qu'on souhaite exécuter un script **JS** plutôt que rediriger l'utilisateur sur une autre page, il suffit de placer `"javascript:;"` en valeur de **l'attribut** `href`
- Vous remarquerez que la **méthode** `alert()` de l'**objet JS** `window` se charge d'afficher la valeur passée en argument sous la forme d'une **modale** au sein de notre page **HTML**

Le Document Object Model (DOM)

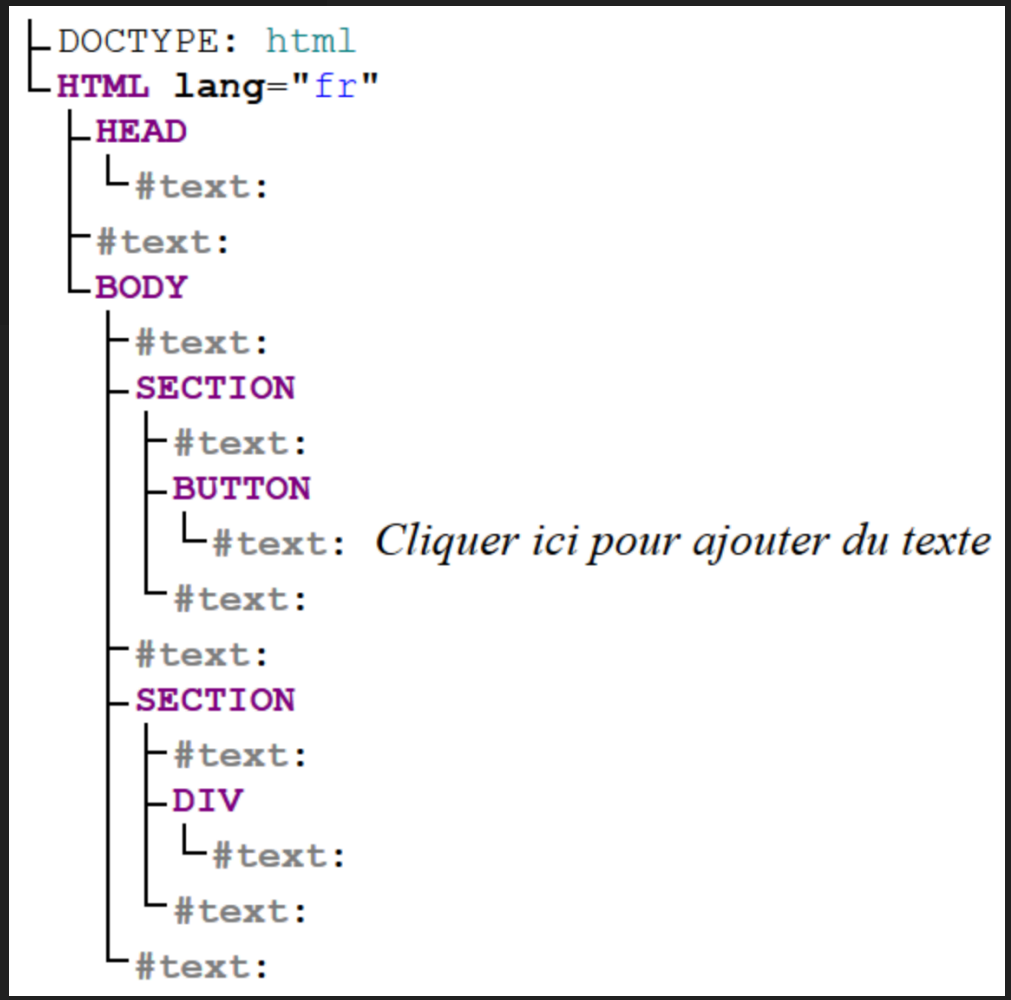
JavaScript avancé

Chris Chevalier - chevalier@chris-freelance.com

Le DOM en quelques mots…

- Dans de nombreuses situations, il peut être intéressant de modifier **dynamiquement** le contenu visuel de nos pages web sans nécessairement avoir à réaliser d'**échanges HTTP** avec le **serveur**
- Grâce à **JavaScript**, nous allons découvrir qu'il est possible d'accéder, manipuler et interagir avec des éléments d'un **document HTML**
- Ces **interactions** se basent sur un **modèle normalisé** appelé **DOM** (pour **Document Object Model**)
- Il s'agit d'une **arborescence** permettant d'accéder facilement à la **structure d'une page HTML** afin de la manipuler de manière **dynamique** et sans échange avec le **serveur**

```html
<!DOCTYPE html>
<html lang="fr">
<head></head>
<body>
    <section>
        <button>Cliquer ici pour ajouter du texte</button>
    </section>
    <section>
        <div></div>
    </section>
</body>
</html>
```



Le DOM en quelques mots…

- Grâce à JavaScript, il est possible de modifier le DOM de manière dynamique en répondant à des événements
- Tentons d'afficher et faire disparaître un texte à chaque clic sur un bouton

Le code HTML


```
8   <body>
9       <!-- Un bouton qui appelle la fonction JS 'toggleText()' -->
10      <a href="javascript:;" onclick="toggleText()">Cliquez-moi</a>
11
12      <div id="toggle-bloc">
13          <!-- Zone où l'on souhaite insérer le texte -->
14      </div>
15
16      <!-- On oublie pas de charger le script JS qui contient la fonction -->
17      <script src="./scripts/script.js"></script>
18  </body>
```

Le code JavaScript


```
1   function toggleText() {
2       // Récupération de l'élément 'div' où insérer le text
3       let toggleBloc = document.getElementById("toggle-bloc");
4       // Si le texte en enfant de la div est vide
5       if (toggleBloc.innerText === "") {
6           // Insertion de code HTML en enfant de l'élément 'div'
7           toggleBloc.innerHTML = "<i>Mon texte caché ;)</i>";
8       } else {
9           // Insertion d'un code HTML vide en enfant de l'élément 'div'
10          toggleBloc.innerHTML = "";
11      }
12  }
```

Des questions ?