Introduction au JavaScript

Développement web front-end : React

Chris Chevalier - chevalier@chris-freelance.com

Qu'est ce que JavaScript ?

Introduction au JavaScript

Chris Chevalier - chevalier@chris-freelance.com

JavaScript en quelques mots…

- Initialement nommé **LiveScript**, **JavaScript** est un langage de script, orienté web, développé en **1995** par la société **Netscape**
- Suite à une association avec le constructeur **Sun**, **Netscape** renomme le langage **JavaScript**, en référence au langage de programmation **Java**, développé par **Sun**
- Il permet l'apport d'améliorations au **HTML** puisqu'il offre la possibilité d'exécuter des scripts côté **client** (navigateur) et non plus côté **serveur**
- Il ne nécessite pas de compilateur mais est fortement dépendant du navigateur web

Un script en quelques mots…

- Un **script** correspond à une portion de code inséré dans une page **HTML**

- Bien entendu, ce dernier n'est pas visible dans un navigateur et est définit entre des **balises spécifiques** `<script></script>` lui signifiant qu'il s'agit d'un **script** écrit dans un **langage** donné

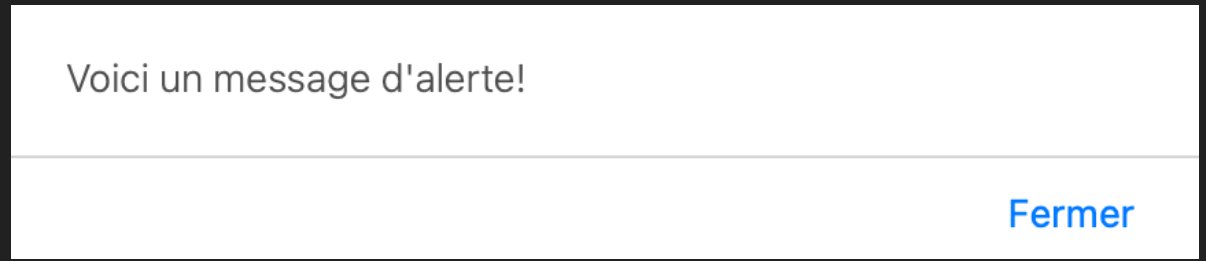
```
<script language="Javascript">

    // Placez ici le code de votre script

</script>
```

Un exemple concret

```
1   <!DOCTYPE html>
2   <html lang="fr-FR">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7       <script language="Javascript">
8           alert("Voici un message d\'alerte!");
9       </script>
10  </head>
11  <body>
12  </body>
13  </html>
```



Explications

- Dans cet exemple, on découvre l'utilisation de la **fonction JS** `alert()` permettant l'affichage d'une **modale d'alerte** au sein du navigateur
- Ici, le **script** est interprété avec **Safari**
- La **modale** s'affichera dès que la **page HTML** aura fini de charger
- Notez que le **script** peut être inséré n'importe où au sein d'un **document HTML**
- Par convention, on écrit nos **scripts JS** dans la **section** *head* d'une page web

Les commentaires

- Comme tout langage de programmation, **JavaScript** peut inclure des **commentaires** dans son code par soucis de lisibilité et de maintenance

- Toutefois, il ne faut pas confondre les balises de **commentaires HTML** avec les commentaires **JavaScript** qui propose sa propre **syntaxe**

- Pour ses **commentaires**, **JavaScript** utilise les mêmes conventions que les langages **C**, **C++**, **PHP**, **etc**

```
<script language="Javascript">

    // Tous les caractères derrière le // sont ignorés

    /* Toutes les lignes comprises entre ces repères
    sont ignorées par l'interpréteur
    de code */

</script>
```

Les balises HTML `<script></script>`

- Il existe plusieurs méthodes pour intégrer un **script** à un **document HTML** :
  - Grâce à des **balises** `<script></script>` placées au sein du **document HTML**
  - En plaçant le code **JavaScript** dans un autre fichier
  - Grâce aux **événements JavaScript**

- Pour insérer du code **JS** écrit dans un autre **fichier**, il suffit de spécifier le **chemin** pointant vers ce **fichier** en **attribut** `src` des **balises** `<script></script>`

```
<script language="Javascript" src="scripts.js"></script>
```

Les variables

Introduction au JavaScript

Chris Chevalier - chevalier@chris-freelance.com

Rappels sur les variables

- Une **variable** permet de stocker des données pouvant être modifiées au cours de l'exécution du programme - Une **variable** est reconnue par son **nom**

- Les développeurs peuvent nommer leurs **variables** librement mais doivent respecter certaines **règles** pour que le nommage soit valider :
  - Un **nom de variable** doit commencer par une **lettre** (majuscule ou minuscule) ou un **underscore** `_`
  - Un **nom de variable** peut comporter des **lettres**, des **chiffres** ainsi que les caractères **underscore** et **esperluette** `&` - Les espaces ne sont pas autorisés
  - Certains noms faisant référence à des instructions ou fonctions **JS** ne peuvent pas être utilisés pour vos **variables** (`boolean`, `break`, `abstract`, `false`, etc.)

- Le **JS** est **sensible à la casse** - Ainsi, `maVariable` ne correspond pas à `MaVariable` ou encore `mavariable`

Déclaration de variables

- On découvrira que **JS** est un langage de programmation particulièrement **souple** dans sa **syntaxe**

- Ainsi, il nous est possible de déclarer une **variable** de manière **explicite** en la faisant précéder par le mot-clé `var` ou `let`

```
var maVariable= "Hello";    let maVariable = "Hello";
```

- **JS** permet également de déclarer une **variable** de manière **implicite** en laissant la liberté au navigateur de déterminer qu'il s'agit bien d'une **déclaration de variable** - Cette fois, **aucun mot-clé** n'est nécessaire

```
maVariable = "Hello";
```

- Même si une **déclaration implicite** sera reconnue par le navigateur, il est recommandé de toujours déclarer nos **variables** de manière **explicite** avec le mot-clé `var` ou `let`

Les types de données des variables

- Contrairement à des langages typés comme le **C**, le **Java** ou le **C#**, **JS** ne nécessite pas de **déclaration de type** pour ses **variables** - Ainsi, **JS** utilise le mécanisme d'**inférence de types** afin que le navigateur détermine **automatiquement** le **type** associé à une **variable**

- En réalité, **JS** autorise la manipulation d'uniquement **4 types de données** :
  - Les **nombres** (**entiers** ou **à virgule**)
  - Les **chaines de caractères** (entre **guillemets simples** `'` ou **doubles** `"`)
  - Les **booléens** (`true` ou `false`)
  - Les **valeurs nulles** (`null`)

Conversion de type

- Même si **JS** gère le changement de **type de variables**, il peut parfois être nécessaire de forcer cette **conversion**

- En ce sens, **JS** fournit **2 fonctions natives** :
  - `parseInt()` => convertit une **variable** en nombre **entier**
  - `parseFloat()` => convertit une **variable** en nombre **décimal**

```
7   <script language="Javascript">
8       var a = "123";
9       var b = "456";
10      document.write(a + b); // Affiche 123456
11      document.write(parseInt(a) + parseInt(b)); // Affiche 579
12  </script>
```

Débogguer du code JavaScript

- Pour afficher ce que contient une variable, il est possible d'utiliser la méthode `console.log()`
- Comme son nom l'indique, cette dernière affiche ce qui lui est passé entre parenthèses (variable, texte, nombre, etc.) dans la console disponible dans le navigateur

```
var prenom = "Chris";
console.log("Bonjour", prenom)
```

- Ainsi, ce code affiche le texte `"Bonjour"` et la valeur de la variable `prenom` dans la console du navigateur

```
Bonjour – "Chris"
```

- Sur la plupart des navigateurs, la **console** est accessible en faisant un clic droit sur la page web, puis « `Inspecter …` »

Les tableaux

Introduction au JavaScript

Chris Chevalier - chevalier@chris-freelance.com

Rappels sur les tableaux

- Une **variable JS** permet de stocker uniquement **une valeur** à la fois - Il peut, néanmoins, être utile de manipuler de nombreuses données

- Pour répondre à cette nécessité, **JS** propose une **structure de données** stockant un **ensemble de valeurs** dans une « **variable commune** » - Il s'agit d'un **tableau**

- Ainsi, un **tableau JS** représente une **variable** pouvant contenir **plusieurs données indépendantes**, indexées par un numéro (**indice**) afin d'accéder simplement aux valeurs stockées

Monodimensionnel vs multidimensionnel

- Lorsqu'un **tableau** est uniquement composé de **variables**, on parle de **tableau monodimensionnel** (ou **unidimensionnel**) - En voici une représentation

Indice	0	1	2	3
Donnée	donnée 1	donnée 2	donnée 3	donnée 4

- Le premier élément d'un **tableau** porte toujours l'**indice** `0` - Ainsi pour un **tableau à** `n` **éléments**, le dernier élément est associé à l'**indice** `n-1`

- Lorsqu'un **tableau** contient plusieurs **tableaux**, on parle de **tableau multidimensionnel** - Voici une représentation

<table>
<tr><td>0</td><td colspan="3">1</td><td>2</td><td colspan="2">3</td></tr>
<tr><td rowspan="2">donnée 1
(variable)</td><td colspan="3">donnée 2 (tableau)</td><td rowspan="2">donnée 3
(variable)</td><td colspan="2">donnée 4 (tableau)</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>0</td><td>1</td></tr>
<tr><td></td><td>donnée 1</td><td>donnée 2</td><td>donnée 3</td><td></td><td>donnée 1</td><td>donnée 2</td></tr>
</table>

Les tableaux associatifs

- Plutôt que d'utiliser des **nombres entiers**, il est possible d'indexer les éléments d'un **tableau** avec des **indices personnalisés** - On parle alors de **tableau associatif**

Indice	"Paul"	"André"	"Pierre"	"Jean-François"
Donnée	16	22	12	25

- **JavaScript** autorise, en effet, l'utilisation de **chaine de caractères** ou de **nombre spécifique** pour indexer les éléments d'un **tableau**

La déclaration de tableau

- Comme **PHP**, **JavaScript** offre 2 manières pour déclarer un tableau

```
7       <script language="Javascript">
8           var MonTableau = ["donnée 1", "donnée 2", "donnée 3", "donnée 4"];
9           var MonTableau = new Array("donnée 1", "donnée 2", "donnée 3", "donnée 4");
10      </script>
```

- Notez qu'il est également possible de déclarer un **tableau vide** à son initialisation

```
7       <script language="Javascript">
8           var MonTableau = [];
9           var MonTableau = new Array();
10      </script>
```

Accès aux données

- Pour accéder aux éléments d'un **tableau**, il suffit d'écrire son nom suivi par des **crochets** contenant l'**indice** de l'éléments à récupérer

```
7       <script language="Javascript">
8           var monTableau = ["donnée 1", "donnée 2", "donnée 3", "donnée 4"];
9           document.write("Le 4ème élément du tableau est " + monTableau[3]);
10          // Afficher "Le 4ème élément du tableau est donnée 4"
11      </script>
```

- À l'inverse, il est possible d'accéder à l'indice associé à une valeur connue d'un tableau à l'aide la méthode `indexOf()`

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.indexOf("Apple")   // Renvoie 2
```

Affectation de valeurs

- Pour créer un **tableau associatif**, il suffit de **déclarer un tableau** avant d'écrire son nom suivi de l'**indice** souhaité entre crochets `[]` et lui affecter une valeur grâce à l'**opérateur d'affectation** `=`

```
7   <script language="Javascript">
8       var monTableau = new Array();
9       monTableau[0] = "Bonjour";
10      monTableau["Pierre"] = 12;
11      monTableau["Jean-François"] = 25;
12  </script>
```

Trier un tableau

- Pour trier les éléments d'un **tableau** par ordre croissant (**alphabétique** ou **numérique**) JavaScript propose la **méthode** `sort()`

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort(); // Trie le tableau par ordre alphabétique
```

- Dans la même idée, vous pouvez utilisez la **méthode** `reverse()` pour **inverser** le contenu du **tableau** ciblé

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.reverse(); // Trie le tableau dans le sens inverse
```

Les chaines de caractères

Introduction au JavaScript

Rappels sur les chaînes de caractères

- Une **chaîne de caractères** correspond à une suite de **caractères**
- En **JS**, on la représente entre des **guillemets simples** `'` ou **doubles** `"`
- Certains **caractères spéciaux** peuvent simuler des caractères non visuels grâce à l'**opérateur antislash** `\` :
  - `\n` : retour à la ligne
  - `\t` : tabulation
  - `\'` : guillemet simple
  - `\\` : caractère antislash
  - etc.

L'architecture d'une chaîne de caractères

- En programmation, une **chaîne de caractères** est représentée par un **tableau** constitué de `n` **éléments** pour `n` **caractères**

- Ainsi, le **premier caractère** est situé à l'**indice** `0` et le dernier à l'**indice** `n-1`

Chaîne	C	o	m	m	e	n	t		ç	a		m	a	r	c	h	e	?
Position des caractères	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Les opérateurs

Introduction au JavaScript

Les opérateurs JS

- Il existe **5 principaux types d'opérateurs JS** :
  - Les **opérateurs de calcul** (`+`, `-`, `*`, `/` et `%`)
  - L'**opérateur d'affectation** (`=`, `+=`, `-=`, `*=`, `/=`, `%=`)
  - Les **opérateurs d'incrémentaion et de décrémentaion** (`++` et `--`)
  - Les **opérateurs de comparaison** (`==`, `===`, `!=`, `!==`, `<`, `<=`, `>` et `>=`)
  - Les **opérateurs logiques** (`&&`, `||` et `!`)

Les structures conditionnelles

Introduction au JavaScript

L'instruction `if` (1/2)

- Pour rappel, une **structure conditionnelle** permet de tester si une **condition** est **vraie** ou **fausse**

- Voici la syntaxe utilisée pour **l'instruction** `if`

```
11    <body>
12        <script>
13            var a = 0;
14            if (a == 0) {
15                // Liste d'instructions
16            }
17        </script>
18    </body>
```

- Notez que l'**instruction conditionnelle** doit être placée entre **parenthèses**

L'instruction `if` (2/2)

- Il est possible de définir **plusieurs conditions** dans une même **instruction** `if` à l'aide des **opérateurs** `&&` et `||`

```
12      <script>
13          var a = 0;
14          var b = 1;
15          if ((a == 0) && (b == 1)) {
16              // Liste d'instructions
17          }
18      </script>
```

- Pour **une seule instruction**, les **accolades** `{}` ne sont **pas obligatoires**

```
12      <script>
13          var a = 0;
14          if (a == 0) document.write("a vaut 0");
15      </script>
```

L'instruction `else`

- **L'expression** `else` permet d'exécuter une autre série d'**instructions** en cas de non-réalisation de la condition testée par un **bloc** `if`

```
12      <script>
13          var a = 1;
14          if (a == 0) {
15              // Liste d'instructions
16          }
17          else {
18              // Une autre série d'instructions
19          }
20      </script>
```

- Notez qu'il est important de correctement **indenter** les différents **blocs d'instructions** afin de gagner en **lisibilité**

L'opérateur ternaire

- L'**opérateur ternaire** permet de tester une **condition** avec **syntaxe** beaucoup plus **concise**

```
12      <script>
13          var a = 0;
14          (a == 0) ? a = 1 : a = 2
15      </script>
```

- La **condition** doit être entre **parenthèses**
- Si la **condition** est **vraie**, l'**instruction de gauche** est exécutée (ici, `a = 1`)
- Si la **condition** est **fausse**, l'**instruction de droite** est exécutée (ici, `a = 2`)

La boucle `for`

- Les **boucles** permettent d'exécuter plusieurs fois une série d'instructions jusqu'à ce qu'une **condition** donnée ne soit plus **vraie**

- La **boucle** la plus commune consiste à créer une **variable** représentant un **compteur** et **s'incrémentant** à chaque **itération** - La **boucle** s'arrête lorsque le **compteur** atteint une valeur prédéfinie

```
12      <script>
13          for (var i = 0; i < 6; i++) {
14              document.write(i)
15          }
16      </script>
```

- Cette simple **boucle** affiche **6 fois** la valeur de `i`, soit `012345`

La boucle `while`

- L'**instruction** `while` représente un autre moyen d'exécuter plusieurs fois la même **série d'instructions**

```
12      <script>
13          var i = 0
14          while (i < 6) {
15              document.write(i)
16              i++
17          }
18      </script>
```

- Cette **instruction** exécute la **liste d'instructions tant que** la **condition** est **vraie**

- Notez qu'il y a un risque de **boucle infinie** et qu'il est donc très important de définir une **instruction d'arrêt** atteignable

Des questions ?