Javascript - Introduction

Javascript est un langage de script coté client, c'est à dire qu'il est exécuté sur le navigateur. Il a été créé en 1995 par Netscape Communication Corporation. A l'époque il s'appelait LiveScript. Fin 1995, et suite à une association entre Netscape et SUN (créateur du fameux langage Java), LiveScript s'est fait rebaptiser Javascript.

Javascript est un standard ECMA Script. En effet, ECMA Script est un langage de script coté client mais il sert de standard dont les spécifications sont respectées par les autres langages de script comme Javascript ou ActionScript (un langage utilisé pour ajouter de l'interactivité aux animations Flash).

Javascript est un langage de programmation orienté prototype (un concept qui ressemble à l'orienté objet mais qui n'utilise pas les classes).
Si à la base il est coté client, on peut cependant l'exécuter au serveur (notamment avec Node.js ou autres platetformes).

Quelques applications possibles

- *Faire bouger, apparaitre ou disparaitre des éléments de la page (un titre, un menu, un paragraphe, une image…).*
- *Mettre à jour des éléments de la page sans recharger la page (changer le texte, recalculer un nombre, etc).*
- *Demander au serveur un nouveau bout de page et l'insérer dans la page en cours, sans la recharger.*
- *Manipuler des sons, des vidéos, des images.*
- *Attendre que l'utilisateur face quelque chose (cliquer, taper au clavier, bouger la souris…) et réagir (faire une des opérations ci-dessus suite à cette action).*
- *Sur les téléphones, pour le moment sur FirefoxOS et PhoneGap, il permet d'écrire des applications.*

Avantage

- *Chargé en même temps que le HTML et le CSS*
- *Exécuté par le client (navigateur)*
- *Pas besoin de compilation (contrairement au JAVA, C# ou ActionScript)*

Inconvénient

- *Lisible par tous les utilisateurs*
- *Différences d'interprétation selon le navigateur*
- *Peut être désactivé par le client*

Javascript - Bases d'un script

*Il y a fondamentalement 2 manières d'intégrer du code javascript, les 2 nécessitent l'ajout d'une balise **SCRIPT** dans le code HTML en spécifiant le type en «**text/javascript**». Elle peut être placée dans le HEAD ou dans le BODY.*

*Dans le premier cas, le code est intégré **directement dans la balise** SCRIPT.*

```
<html>
    <head>
        ...
    </head>
    <body>
        ...
        <script type="text/javascript">
            ...
        </script>
    </body>
</html>
```

*Dans le second cas, le code est intégré depuis **un fichier externe en .JS**, ce fichier ne nécessite plus de balise HTML à l'intérieur.*

```
<html>
    <head>
        ...
    </head>
    <body>
        ...
        <script src="js/script.js" type="text/javascript"></script>
    </body>
</html>
```

Le code s'exécute selon l'ordre de lecture de l'HTML donc si on souhaite atteindre du contenu de la page et y appliquer une modification ou exécuter un événement, il faut soit dire au script d'attendre le chargement de la page, soit le placer après.

```
<html>
    <head>
        ...
        <script type="text/javascript">
            ...
        </script>
    </head>
    <body>
        ...
    </body>
</html>
```

Généralement, le code intégré dans la balise head est utilisé pour préparer des trackers ou pour initialiser des fonctions. Attention qu'un script lourd ralentira l'affichage du reste de la page et donc réduira son efficacité pour le référencement.

Javascript - Variables

Initialisation d'une variable

- ***VAR***
  Portée très large (hormis pour l'initialisation depuis des fonctions), le couteau suisse des variables, mais pouvant occasionner des erreurs s'il est mal utilisé. Il est aussi très lourd puisque tout est gardé en mémoire.
- ***LET***
  *Même effet que **var**, mais avec une portée plus réduite, s'il est intégré dans une condition, la clôture la supprime automatiquement de la mémoire.*
- ***CONST***
  *Elle n'est pas réinscriptible, c'est très léger pour la mémoire, sa portée est identique à **let**.*

Types de variables

- ***String***
  Les chaines de caractères comprises entre guillemets ('...' ou "..."). Les chaines sont mises les unes à coté des autres lorsqu'elles sont additionnées.

```
var message = "Hello World"
```

- ***Number***
  Toutes la valeurs numériques, un nombre décimal est défini après un point. Il ne faut pas de guillemets bien qu'il soit possible d'automatiquement convertir un String contenant un nombre avec certaines opérations tels que la multiplication et la division.

```
var prix = 17.32
```

- ***Boolean***
  La valeur ne peut être que «vrai» ou «faux» (true/false), on peut aussi utiliser 0 pour une faleur fausse et 1 pour une valeur vraie.

```
var online = false
```

- ***Null / Undefined***
  La valeur est initialisée mais n'a pas de valeur.

```
var projet = null
```

- ***Array***
  Les tableaux de données. Les données sont mises les unes à la suite des autres comme dans Excel. Pour gérer des listes, il est possible de trier en réorganisant l'ordre des valeurs. Elles sont initialisées avec des «crochets» ([et]) et les données sont séparées par une virgule.

```
var liste = ["Olives","Fromage","Saucisson"]          // Il est possible d'y accéder avec liste[0] puis liste[1]
var nimporte = [ 5 , 12 , "Vert" , 12.8 ]             // Contient : 5, 12, «Vert» et 12.8
```

- ***Object***
  L'objet peut contenir des cases définies par des clés, on peut y intégrer une variable de n'importe quel type. Elles sont initialisées avec des «accolades» ({ et }), les données sont séparées par une virgule et les clés précèdent leur valeur avec « : »

```
var produits = {
        oli : "Olives",
        fro : "Fromage"
}                                                     // Il est possible d'y accéder avec produits.oli ou produits["oli"]
```

- ***Function***
  Pour associer une procédure à exécuter lors qu'elle est appelée avec des parenthères dans lesquelles il est possible d'ajouter des paramètres.

```
var addition = function(a,b){
        return a+b
}                                                     // Apès, appeler var resultat = addition(5,2) pour obtenir «7»
```

Javascript - Affichage d'une variable

Dans la console

La console du navigateur permet de visualiser différentes informations retournées lors du chargement de la page ainsi que des données souhaitées dans un script javascript avec les fonctions suivantes.

Chrome : CTRL + MAJ + J
Firefox : CTRL + MAJ + K
Internet Explorer : F12 (et puis sélectionnez l'onglet «Console»)
Safari : Command + Option + C

- ***Console Log***
  Principalement pour vérifier une donnée. Il faut l'utiliser régulièrement lors d'une développement pour s'assurer du bon fonctionnement étape par étape de notre code.

  console.log(prix **)**

  ```
  2.28
  ```

- ***Console Warning***
  Comme pour le LOG, il a l'avantage d'indiquer en un coup d'oeil l'un ou l'autre alerte importante.

  console.warn(e.msg)

  ```
  ⚠ Le mot de passe est trop court
  ```

- ***Console Error***
  Comme pour le Warning, ici l'indication est plus marquante et indique un bug, qu'il est sera nécessaire d'intervenir.

  console.error("URL non valide")

  ```
  ❗ URL non valide
  ```

Dans la page

- ***Document Write***
  Ajout d'une variable directement à la suite dans la page html, précisément à la fin de la balise BODY.

  document.write("Hello World"**)**

Dans le DOM (ciblage d'un élément de la page)

- ***getElementById***
  Cible un objet de la page avec son ID, attention qu'un ID doit être unique dans une page. «document» désigne la racine de la page.

  document.getElementById("bloc").text = "Nouveau contenu"

- ***getElementsByClassName***
  Cible un ou plusieurs objets de la page avec sa Classe. Même s'il n'y a qu'un objet retourné, il est impératif, il est tout de même intégré dans un tableau. Il faut donc ne pas oublier de choisir le premier objet avec [0] ou d'y accéder avec une boucle.

  document.getElementsByClassName(".bloc")[0].text = "Nouveau contenu"

- ***getElementsByTagName***
  Cible un ou plusieurs objets de la page avec le nom de la balise. Même s'il n'y a qu'un objet retourné, il est impératif, il est tout de même intégré dans un tableau. Il faut donc ne pas oublier de choisir le premier objet avec [0] ou d'y accéder avec une boucle.

  document.getElementsByTagName("p")[0].text = "Nouveau contenu"

- ***querySelectorAll***
  Cible un ou plusieurs objets de la page avec un sélecteur comme en CSS (ID, Classe, Balise, ...). Même s'il n'y a qu'un objet retourné, il est impératif, il est tout de même intégré dans un tableau. Il faut donc ne pas oublier de choisir le premier objet avec [0] ou d'y accéder avec une boucle.

  document.querySelectorAll(".bloc p")[0].text = "Nouveau contenu"

Javascript - Manipulation d'une variable

Nombres

Les variables de type Number permettent un nombre illimité de manipulations pour automatiser des calculs. On peut calculer un résultat, une position, une animation, une distance.

- ***Calculer un prix TVAC***

```
var prix = 17.99
const tva = 0.21
var total = prix*(1+tva)                    // total = 21.767899999999997
```

- ***Afficher le prix en affichant uniquement 2 chiffres après la virgule***
  On peut appliquer une méthode qui s'appelle «toFixed» sur un nombre en arrondissant et en signalant le nombre d'élément après la virgule souhaité. Il est conseillé de ne l'utiliser que pour l'affichage, sinon nous perdrions une précision dans la valeur et des erreurs pourraient survenir lors d'une manipulation future de la variable.

```
console.log( total.toFixed(2) )             // log = 21.77
```

- ***Conversion depuis une chaine de caractères***

```
var prix = "17.99€"
console.log(parseFloat(prix))               // log = 17.99
console.log(parseInt(prix))                 // log = 17
```

- ***Math**.fonction()*
  Il existe beaucoup de méthodes pour manipuler des nombres directement depuis la variable, mais l'objet Math permet d'appliquer des calculs scientifiques ou des racourcis de cacul.

```
var distance = 24.1662591
var km = Math.floor(distance)               // km = 24

var largeur = 4
var carre = Math.pow(largeur,2)             // carre = 16
var cube = Math.pow(largeur,3)              // cube = 64

var opacity = 0.2
opacity = Math.max(opacity,0.5)             // opacity = 0.5
// Le but est d'avoir le nombre le plus élevé entre notre variable et le nombre «0.5». En bref, je veux que l'opacité soit au minimum de 0.5
```

Les Chaines de caractères

Outre la simple concaténation, il est possible d'effectuer des recherches, de découper, de remplacer des éléments et bien d'autres méthodes. Un nombre peut être considéré comme une chaine de caractère pour y appliquer des transformations.

- ***La concaténation (addition)***

```
var prix = 17.99
console.log( prix+"€" )                                 // log = 17.99€

var nom = "Phoenix"
var prenom = "Joaquin"
console.log( prenom + " " + nom )                       // log = Joaquin Phoenix
```

- ***Conversion et modifications***
  Des modifications peuvent être appliquées les unes après les autres pour obtenir le résultat souhaité.

```
var prix = 17.9932
console.log( prix+"€" )                                 // log = 17.9932€
console.log( prix.toString().replace( "." , "," )+"€" )    // log = 17,9932€
console.log( prix.toFixed(2).toString().replace( "." , "," )+"€" )    // log = 17,99€
```

Javascript - Les tableaux, les objets et les boucles

Les Tableaux

Un tableau, ou plutôt un array en anglais, est une variable qui contient plusieurs valeurs, appelées items. Chaque item est accessible au moyen d'un indice (index en anglais) et dont la numérotation commence à partir de 0.

Le contenu du tableau se définit entre crochets, et chaque valeur est séparée par une virgule. Les valeurs sont introduites comme pour des variables simples, c'est-à-dire qu'il faut des guillemets ou des apostrophes pour définir les chaînes de caractères :

```
var tableau1 = [42, 12, 6, 3];
var tableau2 = [42, "Sébastien", 12, "Laurence"];
```

Comment faire pour récupérer la valeur de l'index 1 de mon tableau ? Rien de plus simple, il suffit de spécifier l'index voulu, entre crochets, comme ceci :

```
var tableau = [42, "Sébastien", 12, "Laurence"];
console.log(tableau[1]);                    // Log : « Sébastien »
```

Différentes méthodes permettent de manipuler les tableaux :

```
var tableau = [42, "Sébastien", 12, "Laurence"];
tableau.shift();                            // Retire le premier élément : 42
tableau.pop();                              // Retire le dernier élément ; "Laurence"
tableau.push( "Ludovic" );                  // Ajoute un élément à la fin : "Ludovic"
```

Les Objets

S'il est possible d'accéder aux items d'un tableau via leur indice, il peut être pratique d'y accéder au moyen d'un identifiant.

```
var myObject = {
   item1: "Texte 1",
   item2: "Texte 2"
}
```

Comment faire pour récupérer un élément de mon objet ? 2 possibilités, mais il est nécessaire de connaitre la clé :

```
var myObject = { seb: "Sébastien", lau: "Laurence" };
console.log( myObject["seb"] )              // Log : « Sébastien »
console.log( myObject.seb )                 // Log : « Sébastien »
```

Pour assigner une nouvelle valeur à l'objet :

```
var datas = {};                             // Objet vide
datas.title = "Hello World";
datas["description"] = "Description de ma page";
```

Les Boucles sur les Tableaux et les Objets

Deux fonctions permettent de boucler les tableaux, l'une renvoit les clés (id ou référence) et l'autre renvoit les valeurs.

```
var liste = { seb: "Sébastien", lau: "Laurence" };
for(cle in liste){
   console.log( cle )                       // Log : « seb », « lau »
}
for(valeur of liste){
   console.log( valeur )                    // Log : « Sébastien », « Laurence »
}
```

Il y a aussi une méthode liée directement aux objets et aux tableaux trè pratique qui renvoit les clés et les valeurs.

```
var liste = { seb: "Sébastien", lau: "Laurence" };
liste.forEach(function(valeur,cle){
   console.log( cle + " : " + valeur)       // Log : « seb : Sébastien », « lau : Laurence »
}
```