Notes relatives au sous-projet 1

D.Moreaux

8 septembre 2024

1 Insérer du Javascript

En HTML 5, pour insérer du Javascript dans une page, on place dans la section `<head>` une ou des balises `<SCRIPT>` avec l'attribut `DEFER`.

Cet attribut permet de s'assurer que le Javascript en question ne sera exécuté qu'une fois le chargement de la page terminé. Si plusieurs fichiers Javascript sont fournis, ils seront exécutés dans l'ordre.

Les pratiques telles que le script inclus dans la page Web ou placé en fin de page sont à éviter. La première afin de séparer le code du HTML, la seconde parce que placer la balise script dans le `<HEAD>` permet de charger ce dernier en même temps que le reste de la page web et des autres ressources.

On précisera le nom du fichier à l'aide de l'attribut `SRC`.

Même en cas de fichier javascript externe, la balise `</SCRIPT>` est obligatoire. Sans cette dernière, le reste de la page sera considéré comme du Javascript.

```
<SCRIPT SRC="monscript.js" DEFER></SCRIPT>
```

2 Le langage Javascript

Le langage Javascript ressemble fortement au C et au PHP. Les instructions `if`, `for`, `while` ont la même syntaxe.

2.1 Les variables

Les variables ne demandent pas de caractère `$` au début (contrairement au PHP) et sont à la base non typées.

Une variable non déclarée sera considérée comme globale et sera contenue dans l'objet [1] `window`. Une variable `a` globale pourra donc toujours être accédée à l'aide du nom `window.a` même si elle a été masquée par une variable locale.

Les variables peuvent être déclarées à l'aide des mots clés `var`, `let` ou `const`.

Une variable déclarée `const` ne sera pas réellement constante. En effet, si l'élément associé à la variable ne peut pas être changé, sa valeur peut par contre souvent être modifiée. Dès lors, la dénomination de *constante* est trompeuse. En effet, à l'exception des entiers et float, tous les autres types de données sont des objets et ces derniers peuvent être modifiés même si la variable les référençant ne peut l'être.

Une variable déclarée `var` sera considérée comme déclarée au début de la fonction. Si une assignation est faite dans la déclaration, l'assignation aura par contre lieu à l'endroit où se trouve la déclaration.

La pression venant de développeurs issus du C# fait que l'on voit de plus en plus de déclarations faites à l'aide de `let`. Il faut cependant garder en tête que comme ces déclarations sont locales au block (et non à la fonction), elles provoquent la création de tables de symboles supplémentaires (ce qui n'est pas spécialement efficace) et peut amener à certaines surprises en cas d'utilisation du mécanisme de closures.

Au niveau du cours, on mettra donc la préférence aux variables de type `var` et on évitera d'utiliser les deux autres formes de déclaration [2]

2.2 Les fonctions

Une fonction en Javascript est un objet comme un autre. En tant que tel, elle peut être sauvée dans une variable, passée en argument ou retournée par une autre fonction.

Une fonction se déclare comme suit :

```
function (paramètres) {
   ...
   return valeur_de_retour;
}
```

1. Le Javascript utilise une programmation objet par prototype, très différente des classes du C++, Java et autres

2. Parfois, `let` peut sembler plus simple dans des situations de closure mais dans les faits, cela masque la présence de la closure et peut amener à des problèmes avec d'autres variables

Cette fonction ne possède pas de nom. La ligne return peut être omise si on ne désire pas retourner de valeur.

On pourra sauver la fonction dans une variable, la passer en argument à une autre fonction ou la retourner à partir d'une autre fonction. Pour sauver la fonction dans une variable, on peut également placer le nom de la variable entre le mot clé `function` et les arguments.

```
f1 = function (a,b) { return a+b; };
function f2 (a,b) { return a+b; }
f3(12,function (a,b) { return a+b; });
function f4(a) {
  return function(x) { return a+x; };
}
```

Dans le cas de f4, on remarquera que la fonction retournée utilise le paramètre passé à la fonction f4. Ce mécanisme appellé closure sera examiné plus tard et peut être oublié pour le moment.

3 L'arbre DOM

La page HTML est convertie en une structure en arbre appelée *arbre DOM*. Différentes méthodes permettent de récupérer des éléments de cette page ou de la modifier.

Si un élément HTML possède un ID, on peut le récupérer à partir du Javascript à l'aide de `el=document.getElementById("monID");`. A noter qu'en HTML, l'ID doit être unique, ce qui prend tout sons sens ici. Le Navigateur garde généralement une table des ID et des éléments associées ce qui rend cette méthode d'accès très rapide.

On peut également récupérer un élément sur base de sa balise ou d'un sélecteur CSS à l'aide de

```
el1=document.getElementsByTagName("DIV");
el2=document.queryAll(".spoil SPAN");
```

Les valeurs obtenues seront des tableaux d'éléments. `el1.length` permettra de connaître le nombre d'élements et on y accèdera à l'aide de `el1[0]`, `el1[1]`,...

On peut également remplacer le mot clé `document`[3] par un élément (obtenu par une des méthodes précédentes par exemple) pour restreindre la recherche au contenu de l'élément en question.

3. En fait, une variable globale `window.document` qui pointe vers le document complet

Une fois que l'on dispose d'un élément, on peut changer ses propriétés CSS à l'aide de `el.style.xxx=...` où xxx est le nom de la propriété CSS. Si la propriété est composée de plusieurs mots séparés par des tirets, on retirera les tirets et mettra des majuscule au mot qui le suit. `border-top-color:` deviendra `borderTopColor` par exemple.

Pour les champs de formulaire, on pourra généralement utiliser la propriété `value` ou (pour les checkbox et radio) `checked` pour récupérer la valeur du champ.

4 Evenements

On peut associer une fonction à un événement de deux manières différentes. La première, appelée DOM1, donne moins de contrôle au processus mais reste utile dans certains cas [4]. Elle consiste à utiliser une propriété de l'élément dont le nom correspond à l'événement. Mettre la propriété à NULL permet de retirer le gestionnaire.

```
document.getElementById("MonBouton").onclick = function (ev) {
        ...
};
document.getElementById("MonBouton").onclick = null;
```

La seconde méthode, appelée DOM2, permet d'associer plusieurs gestionnaires à un même événement et de contrôler la phase dans laquelle la fonction sera exécutée. Par contre, si on désire pouvoir retirer le gestionnaire, il est nécessaire qu'il s'agisse d'une fonction nommée (ce qui peut poser quelques problèmes quand on utilise les closures par exemple).

```
function btnclick(ev) { ... }
document.getElementById("MonBouton").addEventListener("click",btnclick);
document.getElementById("MonBouton").removeEventListener("click",btnclick);
```

Le paramètre du gestionnaire d'événement recevra un objet *event* qui permettra d'avoir plus d'informations sur l'événement.

On peut bloquer la fonction *normale* de l'événement en utilisant la fonction `preventDefault()` de l'objet event transmis au gestionnaire d'événement.

4. Par exemple quand on souhaite associer une fonction anonyme et désire pouvoir la retirer plus tard

5 En Vrac

De nombreux Frameworks[5] définissent la fonction `$()` comme un raccourci à `document.getElementById()`. En effet, cette fonction est probablement une des plus utilisées dans un programme javascript.

Une variation est, si le paramètre n'est pas une chaîne, de retourner le paramètre ce qui permet de faire des fonctions acceptant indifféremment l'ID ou directement l'élément.

On peut trouver des exemples dans le fichier common.js fourni avec les fonctions `show()` et `hide()`.

Les commentaires marqués par `/**` sont des commentaires de type DOxygen et permettent de générer de la documentation automatiquement. A noter que PHPStorm est capable de reconnaître de tels commentaires et de les afficher en pop-up. Des exemples peuvent être trouvés dans le fichier common.js.

5. à part jQuery