Devweb 104 : Initiation au Javascript & à JQuery

Le Javascript permet d'ajouter de l'interactivité à une page web : on peut effectuer des actions de manière dynamique sans recharger la page, ce qui peut apporter un certain confort à l'utilisation.



**Premier (et sûrement pas le dernier) Warning : ** le JavaScript n'est pas à confondre avec le langage Java. Les deux langages n'ont rien à voir.

Sur une application comme MyCentraleAssos, le JS est utilisé à de multiples fins :

- Affichage/Désaffichage de blocs de contenu
- Préformatage des champs (date de naissance XX/XX/XXXX)
- Chargement dynamique de l'annuaire utilisateur & asso (ordering, recherche en temps réel)
- Génération de QR codes contact

Mais le JS est utilisé aussi sans qu'on le voit forcément, notamment avec Bootstrap pour les modals, les tabs...

Comment utiliser le JS

Le JS peut se mettre à divers endroits :

Dans des attributs (JS inline)

```
<div id="cache">Je suis un contenu caché</div>
<button id="un_bouton"
onclick="document.getElementById('cache').style.display='block';">Un
bouton</button>
```

Ce bout de code affichera le block #cache lors du clic sur le bouton.

Dans une balise script

```
<div id="cache">Je suis un contenu caché</div>
<button id="un_autre_bouton">Un autre bouton</button>

<script type="text/javascript">
  document.getElementById('un_autre_bouton').onclick = function(){
    document.getElementById('cache').style.display='block';
  }
```

```
</script>
```

Même action que le code précédent.

Dans un script à part

mapage.html

```
<div id="cache">Je suis un contenu caché</div>
<button id="un_autre_bouton">Un autre bouton</button>
<script type="text/javascript" src="script.js" />
```

script.js

```
document.getElementById('un_autre_bouton').onclick = function(){
  document.getElementById('cache').style.display='block';
}
```

C'est la façon recommandée : séparer les différents codes.

Tester des codes JS

Pour tester des bouts de codes html, css & js il existe un outil très pratique : JSFiddle. Il s'agit d'un outil pour éditer du code très rapidement et le tester, parfait dans le cadre de cette formation, et pour d'autres utilisations...

Un exemple : le code précédent : https://jsfiddle.net/wn5tkLb2/

Un outil pratique en plus : La console JavaScript

Pour l'activer il suffit de faire clic droit sur la page puis **Inspecter**, il faut ensuite aller sur l'onglet **Console**.

Sur Windows on peut utiliser le raccourci clavier **Ctrl + Shift + I**

Dans un code javascript, pour afficher le contenu d'une variable ou un texte dans la console, on utilise l'expression **console.log(contenu);**

```
console.log('Une phrase');
console.log(4);
console.log({'Coucou': 1, 'Bonjour':2});
console.log(['Bonjour', 'Coucou']);
```

Application : pourquoi ce script ne fonctionne pas : https://jsfiddle.net/31q4bv0j/

Utilisation recommandée de JSFiddle

Il y a 4 cases sur JSFiddle : HTML, CSS, JS & preview. Il est pratique de séparer le code sur le site, pas besoin de marquer d'inclusion avec la balise script, JSFiddle fera tout, tout seul.

Le même exemple, avec les codes dans les bonnes cases : https://jsfiddle.net/xhj7dc1t/

Les bases du JS

Les variables

Les variables ne sont pas typées et peuvent contenir tout types d'éléments. Les variables se déclarent avec le mot clé **var** (ou **let** dans les versions les plus récentes du JS).

```
// Les chaines de caractère
let chaine1 = "Bonjour, je m'appelle Jean";
let chaine2 = 'Bonjour, je m\'appelle Jean';

// Les nombres
let nombre = 440;
let flottant = 4.1;

// Plusieurs déclaration avec un seul let
let nombre1 = 3, nombre2 = 5,
    nombre3 = 8;

// Les tableaux
let tableau1 = [1, 2, 3, 4],
    tableau2 = [1, 'oui', "non"],
    tableau3 = [[111, 112], ["oui", "non", [1111, 1112]]];
// Les tableaux sont indexés à 0 comme dans la plupart des vrais langages
console.log(tableau1[0]); // Affiche 1
console.log(tableau2[2]); // Affiche non
console.log(tableau3[1][2][0]); // Affiche 1111

// Les dictionnaires
let dict = {
   'cle1': 'valeur1',
   'cle2': 'valeur2',
   'cle3': {
      'oui': 'La réponse positive',
      'non': 'La réponse négative'
   },
   'cle4': ['Oui', 'Non', 'Peut-être']
};

// On y accède comme avec les tableaux ou en faisant dict.cle :
console.log(dict.cle1); // valeur1
```

```
console.log(dict['cle2']); // valeur2
console.log(dict.cle3['oui']); // La réponse positive
console.log(dict['cle4'][1]); // Non
```

Pour voir les résultats : https://jsfiddle.net/6koLj0x1/ (Ne pas oublier d'ouvrir la console JS)

Le JSON

Le JSON (JavaScript Object Notation) est la manière dont on note les objets en JS. Ce format permet l'échange de données entre différents services, notamment c'est le format utilisé dans les API pour que les applications communiquent entre elles.

```
{
    "id": 1,
    "username": "rgrondin",
    "email": "XXXX",
    "nom": "GRONDIN",
    "prenom": "Romain",
    "surnom": null,
    "promoEntrante": 2016,
    "showTel": true,
    "tel": "+336XXXXXXXX",
    "associations": [
        {
            "id": 1,
            "nom": "GInfo"
        },
        {
            "id": 19,
            "nom": "Fablab Marseille"
        },
        {
            "id": 19,
            "nom": "Fablab Marseille"
        }
    ],
    "showCotisation": true
}
```

Ici par exemple les informations renvoyées par MyCentraleAssos lorsqu'on se connecte à une autre application via My. Généralement on représente des dictionnaires dans différents langages avec le JSON.

Structures de base

les conditions

```
let value = 4, value2 = 6;

if(value == 4){
   if(value2 == 6){
   }else if(value2 < 6){
   }
}else if(value == 3 && value2 == 6){

}
```

Les boucles

```
let i, nb = 0;

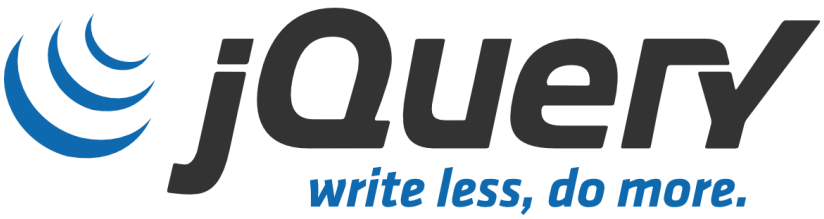
// Ces 2 boucles font la même chose
while(nb < 10){
   console.log(nb);
   nb++;
}

for(i = 0; i < 10; i++){
   console.log(i);
}

// Itérer sur un tableau
let tab = [1, 12, 13, 14];
for(i in tab){
   console.log(i); // 0 1 2 3
   console.log(tab[i]); // 1 12 13 14
}
```

JQuery

Il est rare aujourd'hui de voir des développeurs utiliser JS tout seul, ils utilisent quasiment toujours un Framework leur permettant de gagner du temps. Un des plus populaires (qui n'est pas vraiment un framework mais plutôt une boite à outils) est JQuery.



**Ainsi le code précédent passe de ça : **

script.js

```
document.getElementById('un_autre_bouton').onclick = function(){
  document.getElementById('cache').style.display='block';
}
```

**à ça : **

script.js

```
$('#un_autre_bouton').click(function(){
   $('#cache').show();
});
```

Et le petit fiddle : https://jsfiddle.net/rdog9m17/



Petit bonus avec JQuery, mettez en argument de la fonction show les nombres 200, 400, ou 800 et voyez ce qu'il se passe.

Les sélecteurs

JQuery marche sur le principe des sélecteurs DOM. On sélectionne des éléments dans la page et on interagit avec. Lorsqu'on sélectionne un ou des éléments avec JQuery on obtient un objet jquery avec lequel on peut faire des actions.

Quelques exemples :

```
$('body'); // Récupère toutes les instances de balise body... Normalement
y'en a qu'une
$('div'); // Récupère tous les div de la page

$('.card'); // Récupère tous les éléments ayant la class card sur la page
$('#card'); // Récupère UN élément ayant l'id card

$('div.card'); // Div ayant class card
$('div#card'); // Div ayant class card

$('[href]'); // Récupère tous les éléments de la page ayant un attribut
href... Du coup tous les liens
$('input[type=checkbox]'); // Récupère les checkbox de la page
$('[type=checkbox]:checked'); // Toutes les checkbox cochées
```

Quelques exemples : https://jsfiddle.net/hb41nqpu/8/

Altérer le contenu

Pour modifier le contenu d'une page, on peut utiliser les méthodes .text(valeur) ou .html(valeur) de JQuery. La différence est que **.text** va échapper le html, alors que **.html** non.

Comme vu précédemment, les méthodes **.show** et **.hide** permettent d'afficher ou de cacher un élément.

Exemple : https://jsfiddle.net/9dy12pau/

Utilisation avec les formulaires

on change

Le comptage des feuilles

On veut un input qui contient un nombre de feuilles et qui affiche un widget avec le poids associé. (Indice : une feuille fait à peu près 5g)

```
<input type="number" name="nb" id="nb" />
<button id="calculer">Calculer</button>
<div id="poids" style="display: none;">
  <span></span>g
</div>
```

Corrigé : https://jsfiddle.net/27ctgs9e/

La validation d'un nom d'utilisateur

Cette fois ci on ne veut pas une actualisation à chaque caractère mais une fois la saisie finie. On veut s'assurer que le nom d'utilisateur doit comprendre entre 3 et 15 caractères, et qu'il ne doit pas être déjà pris. (Indice : faire un tableau avec une liste de noms déjà utilisés)

```
<div id="erreur"></div>
<input type="text" name="username" id="username" placeholder="Nom
d'utilisateur" />
<input type="password" name="pass" id="pass" placeholder="Mot de passe" />
```

Corrigé : https://jsfiddle.net/oyfarkq1/

Le focus

focus blur

Les différents éléments du formulaire

Récupérer le contenu d'un formulaire entier

```
<form action="" method="post" id="formulaire">
    <input type="text" name="username" id="username" />
    <input type="password" name="password" id="password" />
</form>
```

```
console.log($('#formulaire').serialize());
```

Les plugins JQuery

Beaucoup de fonctionnalités proviennent de plug-ins JS qui sont fait pour fonctionner avec JQuery (tous les plugins liés à bootstrap notamment : tooltip, popover, modal...).

AJAX

AJAX est la technologie qui permet d'effectuer des requêtes HTTP sans recharger la page. Cette technologie est massivement utilisée sur énormément du site pour apporter du confort à la navigation, pour des usages divers et variés.

*On retrouvera cette technologie aussi sous les noms de **XMLHttpRequest**, ou **XHR**.*

Principe

Pour un chargement de page normal, le navigateur effectue la requête au site, et le site renvoie la page en HTML et le navigateur effectue le rendu.

Pour une requête AJAX, la page est déjà chargée comme précédemment, et le code Javascript suite à une action spécifique (clic sur un lien, au chargement de la page, lors du remplissage d'un champ...) effectue une requête au serveur sans recharger la page, et le site peut renvoyer différents types de données (HTML pour remplacer une partie de la page, JSON pour envoyer des données) que le code javascript va traiter après.





(Source)

Sur ce schéma, la partie **données XML** peut représenter un bout de code HTML, du code JSON (ou en effet du XML, tout dépend des choix des développeurs : nous utiliserons du JSON dans notre cas).

Utilisation avec JQuery

Comme pour la plupart des choses en javascript, les requêtes AJAX sont lourdes en javascript pur. Mais JQuery rend cela plus facile.



Cette affirmation n'est plus forcément vrai aujourd'hui, avec les nouvelles version de javascript une nouvelle fonction (**fetch**) a été ajoutée ce qui permet des requêtes AJAX plus simples.

Un exemple de requête ajax se fait de cette façon :

```
$.ajax({
    success: function(data){
        console.log(data);
    }
});
```

Ce code est le plus simple possible. Il effectue une requête sur la page courante et logue dans la console le résultat de celle-ci. Bien évidemment il est possible de faire bien plus avancée :

```
$.ajax({
```

```
    url: '/ma/page/ajax',
    method: 'POST',
    data: {
        username: 'rom',
        password: 'coucou'
    },
    success: function(data){
        console.log(data);
    },
    error: function(){
        console.log('Erreur !');
    }
});
```

Exemple : un compteur en temps réel

Pour cet exemple on veut un compteur de visiteurs sur la page, qui s'actualise en temps réel pour tous les visiteurs. Pour cela on va faire une requête AJAX toutes les 2 secondes qui vont checker un fichier compteur.txt qui sera incrémenté à chaque chargement de la page (par un script php).

Pour l'exemple on peut avoir le fichier html de base suivant :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>Mon compteur</h1>
    <h2 id="compteur"></h2>
</body>
</html>
```

Ou ici le h2#compteur sera modifié pour afficher la valeur via le javascript.

Le code suivant montre la requête AJAX pour récupérer la valeur du fichier et l'afficher.

```
$.ajax({
    url: 'compteur.txt?'+(new Date()),
    success: function(data){
        $('#compteur').text(data);
    }
})
```



Le `new Date()` sur le fichier permet d'éviter les problèmes de cache : un fichier déjà chargé par le navigateur ne va pas être rechargé directement, or ici on veut la valeur en temps réel.

Il ne manque plus que le code php pour incrémenter la valeur :

```php
<?php
$compteur = file_get_contents('compteur.txt');
file_put_contents('compteur.txt', $compteur+1);
?>
```

Ce qui nous donne au final :

index.php

```php
<?php
$compteur = file_get_contents('compteur.txt');
file_put_contents('compteur.txt', $compteur+1);
?>
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>Mon compteur</h1>
    <h2 id="compteur"></h2>

    <script src="js/jquery.min.js"></script>
    <script>
        function getNumber(){
            $.ajax({
                url: 'compteur.txt?'+(new Date()),
                success: function(data){
                    $('#compteur').text(data);
                }
            })
        }
        getNumber();
        let interv = setInterval(getNumber, 2000);
    </script>
</body>
</html>
```



Ne pas oublier de télécharger JQuery ainsi que de créer un fichier compteur.txt initialisé à 0.

Cette formation a été réalisée par Romain GRONDIN.

From:
https://wiki.centrale-med.fr/ginfo/ - **Wiki GInfo**

Permanent link:
https://wiki.centrale-med.fr/ginfo/formations:devweb_4

Last update: **03/09/2021 13:30**