HTML et JavaScript

Le **JavaScript** est un langage de programmation de type script (sans compilation), exécuté par le navigateur.

Grace au JavaScript, on peut dynamiquement (et sans rechargement) modifier l'apparence de la page web, en fonction **d'événements** (déplacement souris, clic, redimensionnement fenêtre, …).

Un JavaScript peut effectuer des calculs, des analyses de chaines de caractères, des opérations répétitives (boucles), des tests et des affichages.

Le JavaScript sait également utiliser le DOM (*Document Object Model*) qui permet de « voir » la page html comme un ensemble d'objet, avec leurs méthodes et leurs propriétés.
AJAX et WebSocket sont des technologies ajoutées à JavaScript qui permettent d'obtenir des informations venant de serveurs sans rechargement de la page.
Le JavaScript peut être complété par des bibliothèques d'outils comme JQuery. Cependant les versions récentes de Javascript (ECS6) combiné au CCS3-webKit et au HTML5 rendent le JQuery moins nécessaire.

Où placer le JavaScript ?
Un JavaScript peut être intégré à la page html (balise `<script> </script>`)
ou placé dans un fichier à part (.js) en ajoutant dans le <head> (s'il s'agit de bibliothèques) ou à la fin du <body> si le script touche à la structure du HTML :

```
<script src="myScript.js"></script>
```

1. **Premier exemple** : un script exécuté au chargement de la page.

```
<p id="demo" > Bonjour </p>

<script>
        document.getElementById("demo").innerHTML="Au revoir";
</script>
```

Résultat : Bonjour est remplacé par Au revoir …
Explication : L'attribut « **id » permet de repérer de façon unique** une balise.
Le script recherche cette balise (« getElementById ») et modifie sa propriété « innerHTML » qui correspond au texte « Bonjour ».

C'est de la programmation OBJET : objet « document », méthode « getElementById », propriété « innerHTML »

A SAVOIR : En HTML5/JS ECS6, on peut remplacer :

~~`document.getElementById("demo").innerHTML="Au revoir";`~~

par :

```
document.querySelector("#demo").innerHTML="Au revoir";
```

ou même :

```
demo.innerHTML="Au revoir";
```

Remarque : une balise possédant un attribut « id » est directement un objet accessible dans le DOM.

2. **Déclencher un script sur un évènement**

Différentes façons de déclencher sur évènement (exemple de l'évènement *onclick*) :

Méthode HTML : `<element onclick="myScript()">`

Méthode « DOM » (DOM 0): `objet.onclick = function(){myScript};`
(DOM 2): `Objet.addEventListener("click", myScript);`
Observation : en DOM2, les évènements n'ont pas le préfixe « on »

Pour la suite nous utiliserons la technique la plus récente : DOM2

Exemple : un script exécuté sur un évènement de la souris.

```
<p id="demo" > Bonjour </p>
<button id="maDiv">ALLEZ!</button>

<script>
        maDiv.addEventListener('click',ma_fonction);

        function ma_fonction()
        {
            document.querySelector("#demo").innerHTML="Au revoir";
        }
</script>
```

Lorsque l'on clique sur le bouton identifié par « maDiv », le texte « Bonjour » est remplacé par « Au revoir ».

Explication : Ici, on utilise DOM2 et la méthode qui permet d'associer à l'objet « maDiv » une réaction à l'évènement « click » : en cas de « click », on exécute la fonction « ma_fonction ».

Une fonction se déclare avec le mot **function** des parenthèses et des accolades. Le passage de paramètres est possible entre les parenthèses.

3. Exploiter l'évènement :

Lorsqu'un évènement est déclenché, il est possible d'en extraire des informations utiles pour le script, par exemple l'objet qui est à l'origine de l'évènement : `e.target`
Ce peut être aussi une information concernant l'évènement lui-même.
Dans l'exemple qui suit, sur l'évènement « Touche clavier » on récupère le code associé à la touche (*keycode*) :

```
<input type="text" id="saisie"/>
<div id="info"> Tapez une touche </div>

<script>
    saisie.addEventListener("keyup", laTouche);

    function laTouche(e)
    {
          var msg = "Le code de la touche est : " ;
          msg = msg + e.keyCode;
          info.innerHTML = msg;
    }
</script>
```

Lorsqu'une touche remonte (KeyUp), la fonction « laTouche » est appelée. Entre parenthèses, on indique le nom d'une variable qui servira à stocker l'**objet évènement système.** Ici on l'a nommée « **e** ». Cette variable spéciale est générée automatiquement par le navigateur et contient les informations de l'évènement (la touche relâchée).
On note au passage comment la variable « msg » est déclarée (var) et remplie par petit bout grâce au « + ».

4. Evènement « chargement »

```
window.onload = function(){
      document.body.style.color = "blue";
}
```

A chaque rechargement de page le script met en texte bleu tout ce que contient le <body>
La *méthode* onload existe pour d'autres objets, les images par exemple.

5. Des POPUPs :

Les fonctions *alert(), confirm(), prompt()* servent à ouvrit des fenêtres popup pour un message ou une saisie.

6. Debug :

Les navigateurs ont une option « debug » : <F12>, ou clic-droit, « inspecter » ou « examiner l'élément »
De plus, la fonction console.log() permet de visualiser le contenu d'une variable.
Ex : console.log("info", toto); pour afficher la variable « toto » dans la fenêtre « console » du débug.