04-05-06 – Javascript

I – Javascript : généralités

1.1 Intégration du Javascript

Le langage Javascript est un langage _____ dont un interpréteur est incorporé dans tous les navigateurs. On peut faire exécuter du Javacript de trois façons :

Version 1 - en plaçant directement le script entre les balises **<script></script>** dans le code HTML. Simple mais peu portable.

Version 2 - en fournissant une adresse dans via l'attribut **src** de la balise **<script>** placée juste avant **</script>**. Le script va bien se télécharger et s'exécuter en dernier. Simple mais pas très propre.

Version 3 - en fournissant une adresse dans via l'attribut **src** de la balise **<script>** placée dans la balise **<head>**. Propre mais nécessite d'attendre le chargement de la page avant de lancer réellement le script.

Version 1 : dans le fichier HTML

Fichier_HTML :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
  </head>
  <body>
    <main>
      <h1>Premiers tests</h1>
    </main>
    <script>
alert("Hello World !");
    </script>
  </body>
</html>
```

Fichier_Javascript : aucun

Version 2 : Via l'attribut src de link en fin de HTML

Fichier HTLM :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
  </head>
  <body>
    <main>
      <h1>Premiers tests 2</h1>
    </main>
    <script src="mes_scripts.js"></script>
  </body>
</html>
```

Fichier Javascript nommé **mes_scripts.js**:

```
alert("Hello World !");
```

Version 3 : Via l'attribut src de link Balise link dans head

Fichier HTLM :

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <script src="mes_scripts.js"></script>
  </head>
  <body>
    <main>
      <h1>Premiers tests 2</h1>
    </main>
  </body>
</html>
```

Fichier_Javascript nommé **mes_scripts.js**:

```
function demarrage() {
  alert("Hello World !");
}
window.addEventListener("load",demarrage);
```

1,2 Quelques explications :

→ Chaque instruction se finit pas un point-virgule (rien en fin de ligne en Python).
→ C'est le bloc d'accolades qui fixe l'appartenance (c'est la tabulation après le : en Python).
→ On déclare une fonction avec le mot-clé **function** (on utilise le mot-clé def en Python).
→ La fonction **`alert()`** permet de faire apparaître un message pop-up affichant le string reçu en paramètre.

Cas du gestionnaire d'événements :
→ **`window.addEventListener("load",demarrage)`** veut dire « Sur la fenêtre (window), surveille l'événément chargement complet (addEventListener("load",...) et lorsqu'elle est chargée, lance la fonction demarrage.

II – Variables et calculs

2.1 Les types possibles en Javascript : number, boolean, string et undefined.

2.2 Portée des variables : différente de Python également :

1 – Ne rien placer devant la première déclaration d'une variable en fait une variable _____ . C'est ennuyeux.

2 – Placer le mot-clé **var** devant la première affectation d'une variable en fait une variable locale à la zone de déclaration : une variable locale à la fonction si on la déclare dans une fonction par exemple. Le comportement est identique à Python.

3 – Placer le mot-clé **let** devant la première affectation d'une variable en fait une variable locale à la zone du bloc d'accolades {} où elle est déclarée. En dehors du bloc, la variable n'existe plus.

2.3 Opérateurs arithmétiques habituels :

- Addition et soustraction avec + et -
- Multiplication et division avec * et /
- Modulo ou reste avec de la division euclidienne avec %
- Pas d'opérateur de quotient de division euclidienne (// en Python). // sert à signaler un commentaire fin de ligne.
- Pas d'opérateur de puissance.
- Remarque : le commentaire multiligne se fait entre */* en ouverture et */ en fermeture.*

2.4 Fonctions permettant de convertir des types de données en un autre :

- `parseInt(2.8)` va renvoyer 2, parseInt("2") va renvoyer 2.
- `parseFloat("2.7")` va renvoyer 2.7

III – Debug

En cas de problèmes, on peut afficher des valeurs hors de vue de l'utilisateur moyen. On affiche le contenu dans la console disponible dans les outils de développements et en utilisant la fonction log() sur l'objet console :

`console.log(a);` → On demande d'afficher le contenu de la variable a dans la console.

IV – Prompt

La fonction **prompt()** permet d'ouvrir un pop-up dans lequel l'utilisateur pourra taper sa réponse.
Attention : la réponse est toujours comprise comme un string, il convient donc de la convertir si vous voulez faire un calcul.
La fonction `prompt()` crée un pop-up équivalent au `input()` de la console Python.
La fonction `alert()` crée un pop-up équivalent au `print()` de la console Python.

V – Interaction avec l'interface graphique NAVIGATEUR WEB

Il existe trois niveaux d'interaction avec une page Web :

→ **cas 1** : page interactive localement graçe à Javascript. L'utilisateur peut parvenir à modifier l'apparence de la page en cliquant ou en tapant sur certaines touches. Aucune communication avec le serveur Web distant, toutes les modifcations sont purement locales à cet ordinateur.

→ **cas 2** : page dynamique : lorsque l'utilisateur envoie une requête HTTP, la réponse envoyée peut changer en fonction des moments. Le contenu de la page HTML envoyée va dépendre des données dans la base de données du serveur Exemple typique : voir l'ensemble des chaussures d'une marque donnée.

→ **cas 3** : communication directe. Le contenu visible à l'écran peut évoluer sans que l'utilisateur ne demande la moindre mise à jour. La page communique constamment sous forme de petits messages avec le serveur distant.

Nous allons traiter cette année uniquement les cas 1 et 2.

VI – Evenements

La plupart des langages de programmation incorporent un **gestionnaire d'événements**. Il s'agit d'un programme qui surveille ce qui se passe sur les capteurs de la machine et qui déclenche alors automatiquement une **fonction dite événementielle**.

Il existe deux façons de signaler vos demandes au gestionnaire d'événements.

→ en utilisant la méthode **`addEventListener()`**. Recommandé mais lourd pour cette partie de cours en NSI.
→ en utilisant des attributs directement dans les balises HTML à surveiller. Pratique mais peu académique.

6.1 Gérer l'événement en 100 % JAVASCRIPT

JS : `window.addEventListener("load",demarrage);`

- Avantage : plus souple que la méthode 2 car on peut activer ou désactiver l'événement
- Désavantage : plus lourd à taper que la méthode 2.

```
1   <!DOCTYPE html>
2   <html lang="fr">
3
4     <head>
5       <meta charset="UTF-8" />
6       <title>Gestion des événements</title>
7     </head>
8
9     <body>
10      <main>
11        <h1>Gestion au démarrage</h1>
12      </main>
13
14      <script>
15  function demarrage() {
16    alert("Hello World !");
17    alert("Déclenché depuis JS");
18  }
19
20  window.addEventListener("load",demarrage);
21      </script>
22    </body>
23  </html>
```

6.2 Gérer l'événement via HTML et JAVASCRIPT

HTML `<body onload="demarrage()" >`

- Avantage : plus simple que la méthode 1
- Désavantage : on ne peut pas désactiver la gestion une fois activée.

Exemple avec attribut **onload** dans l'une des balises HTML

```
1   <!DOCTYPE html>
2   <html lang="fr">
3
4       <head>
5         <meta charset="UTF-8" />
6         <title>Gestion des événements</title>
7       </head>
8
9       <body onload="demarrage()" >
10        <main>
11          <h1>Gestion au démarrage</h1>
12        </main>
13
14        <script>
15   function demarrage() {
16    alert("Hello World 2 !");
17    alert("Déclenché avec l'attribut HTML");
18  }
19        </script>
20      </body>
21    </html>
```

On notera que l'attribut sur la ligne 10 est **onload** avec le préfixe **on**, et pas juste **load**.

VII – L'événement click

```
<img onclick="fonction_qu_on_veut_activer()" src="soleil.png">
<p onclick="alert('CLIC sur le paragraphe détecté !')">Clique sur moi !</p>
```

VIII – Modifier une balise à l'aide de this

Lorsqu'on utilise un script basé sur un événement, on peut utiliser une variable particulière nommée **`this`**. Sa particularité ? Elle contient automatiquement **la référence de la balise sur laquelle on vient d'agir.**

```
<img width="100px" onclick="alert(this.src);" src="HTML5_150.png">
<img width="100px" onclick="alert(this.src);" src="HTML5_300.png">
```

Lorsqu'on cliquera sur la première image, cela va donc provoquer l'apparition d'un pop-up qui affichera le contenu de l'attribut src de l'image, soit **`"HTML5_150.png".`**

8.1 Lire les attributs de la balise sur laquelle vient de se déclencher un événement

Pour obtenir la largeur : **`this.width`**
Pour obtenir la source : **`this.src`**
Pour obtenir le style CSS imposée depuis le HTML : **`this.style`**

8.2 Modifier les attributs de la balise active lors de l'événement

Il suffit de réaliser une affectation.

Pour augmenter par deux la taille d'une image :
```
<img width="100px" onclick="this.width = this.width * 2;" src="HTML5.png">
```

Pour changer l'URL de la source d'une image :
```
<img width="100px" onclick="this.src = 'python.png';" src="HTML5.png">
```

Pour changer la couleur de fond d'un paragraphe :
```
<p onclick="this.style.backgroundColor = 'red';">Contenu interne qui s'affiche</p>
```

Remarque : attention, JS interdit l'utilisation de – dans les noms d'attributs ou de variables.
La propriété CSS se nomme background-color mais l'attribut JS se nomme backgroundColor.
Pour une autre raison, la propriété CSS des classes se nomme class mais l'attribut JS se nomme className.

Attention : n'oubliez pas style lorsque vous voulez modifier les propriétes CSS.

IX – Autres événements

Attention : pensez à rajouter on devant le nom de l'événement si vous l'activez directement depuis HTML.

Nom Javascript	Description
click	On clique et on relâche sur la balise.
dblclick	Pareil mais en clic double.
mousedown	On clique gauche sans relâcher sur la balise.
mouseup	On relâche gauche sur la balise.
mouseover	On survole la balise
mousemove	On déplace la souris sur l'élément.
mouseout	On fait sortir la souris de la balise.

Il existe aussi des événements liés au clavier.

X – innerHTML pour lire ou modifier le contenu interne des balises

Contenu interne : il s'agit du texte contenu entre la balise d'ouverture et la balise de fermeture.

`<p>`Bonjour à tous !`</p>` → contenu interne : _____

`<p` style="color:'blue';"`>`Bonjour à tous !`</p>` → contenu interne : _____

Seules les balises orphelines n'ont pas de contenu interne.

On peut lire ou modifer ce contenu interne en utilisant l'attribut **`innerHTML`**.

Exemple : on rajoute un a au texte affiché à chaque fois qu'on clique sur le paragraphe :
```
<p onclick="this.innerHTML = this.innerHTML + 'a';">Bonjour à tous !</p>
```

XI – Formulaire

Les formulaires permettent deux choses :
* transmettre des informations choisies par l'utilisateur (pas pour aujourd'hui)
* localiser facilement certaines balises (ça, c'est aujourd'hui).

On obtient facilement la référence mémoire d'une balise button, input ou output située dans une formule à l'aide de cette codification : **name_du_formulaire.name_de_la_balise**.

```
<form name="asticot" method='' action=''>
   <button type="button" name="bA" onclick="toto.bB.style.backgroundColor='red';">
      Bouton n°1
   </button>
   <button type="button" name="bB" onclick="toto.bA.style.backgroundColor='blue';">
      Bouton n°2
   </button>
</form>

<form name="toto">
   <button type="button" name="bA" onclick="asticot.bB.style.backgroundColor='red';">
      Bouton n°3
   </button>
   <button type="button" name="bB" onclick="asticot.bA.style.backgroundColor='blue';">
      Bouton n°4</button>
</form>
```

onclick="toto.bB.style.backgroundColor='red';" veut donc dire :
* dans le formule toto, trouve la balise dont le name est bB.
* change son fond coloré en rouge.

XII – Balises d'interaction

On trouve deux types particuliers de **balises orphelines** : les balises d'interaction liées aux formulaires.
Les balises **input** permettent de facilement récupérer les réponses de l'utilisateur.
Les balises **output** permettent de facilement afficher des messages à l'utilisateur.

Les deux principales différences avec les balises normales ?
* **on peut leur attribuer un attribut name** qui fonctionne (contrairement à p ou autre), cela permet de la localiser facilement (voir la partie XI).
* elles sont orphelines et ne possèdent pas de innerHTML. Par contre, elles posssèdent un **attribut value** qui va jouer le même rôle : contenir la réponse de l'utilisateur.

```
<form name="saisie" method='' action=''>
  <input type="number" name="choix">
  <button type="button" name="affiche" onclick="alert(2*saisie.choix.value);">
    Afficher le double de la valeur
  </button>
</form>
```

Il existe une multitude de balises input, chacune adaptée à un type particulier de réponse.
Le seul intérêt de la balise output par rapport à une balise p est d'avoir un name fonctionnel.

XIII – Attribut onchange

Il s'agit d'un attribut bien pratique puisqu'il permet de lancer automatiquement une action lorsque la réponse de l'utilisateur dans le formulaire vient de changer.