TP web10 : initiation au JavaScript

Objectifs : Découverte des améliorations majeures apportées par le Langage JavaScript

Le JavaScript est un langage essentiellement utilisé avec le HTML, vous allez donc apprendre dans ce TP comment intégrer ce langage à vos pages Web, découvrir sa syntaxe de base et afficher un message sur l'écran de l'utilisateur.

1) Afficher une boîte de dialogue : Le Hello World!

Ne dérogeons pas à la règle traditionnelle qui veut que tous les tutoriels de programmation commencent par afficher le texte « Hello World! » (« Bonjour le monde ! » en français) à l'utilisateur.
Voici un code HTML simple contenant une instruction (nous allons y revenir) Javascript, placée entre 2 balises **<script> et </script>** :

Code : HTML - Hello World!

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
  </head>

  <body>

    <script>

      alert('Hello world!');

    </script>

  </body>
</html>
```

Dans le code HTML donné précédemment, on remarque quelques nouveautés.
Tout d'abord, un élément **<script>** est présent : c'est lui qui contient le code JavaScript

Il s'agit d'une instruction que l'ordinateur va devoir réaliser. Les langages de programmation sont constitués d'une suite d'instructions qui, mises bout à bout, permettent d'obtenir un programme ou un script complet.

Dans cet exemple, il n'y a qu'une instruction : l'appel de la fonction alert().

La boîte de dialogue alert()

alert() est une instruction simple, appelée **fonction**, qui permet d'afficher une boîte de dialogue contenant un message. Ce message est placé entre apostrophes, elles-mêmes placées entre les parenthèses de la fonction alert().

2) La syntaxe du JavaScript Les instructions

- La syntaxe du JavaScript n'est pas compliquée. De manière générale, les instructions doivent être séparées par un point-virgule que l'on place à la fin de chaque instruction.
- Le JavaScript n'est pas sensible aux espaces. Cela veut dire que vous pouvez aligner des instructions comme vous le voulez, sans que cela ne gêne en rien l'exécution du script.
- L'indentation, en informatique, est une façon de structurer du code pour le rendre plus lisible. Les instructions sont hiérarchisées en plusieurs niveaux et on utilise des espaces ou des tabulations pour les décaler vers la droite et ainsi créer une hiérarchie.
- Les commentaires sont des annotations faites par le développeur pour expliquer le fonctionnement d'un script, d'une instruction ou même d'un groupe d'instructions. Les commentaires ne gênent pas l'exécution d'un script.

// Commentaire de fin de ligne /* Commentaire multi lignes */

- Les codes JavaScript sont insérés au moyen de l'élément **<script>**. Cet élément possède un attribut type qui sert à indiquer le type de langage que l'on va utiliser. Dans notre cas, il s'agit de JavaScript,

3) Fichier JavaScript :

On vient d'écrire une instruction JavaScript directement dans la page Web, mais il est possible, et même conseillé, d'écrire le code JavaScript dans un fichier externe, portant l'extension .js…. ça fait plus professionnel. Ce fichier est ensuite appelé depuis la page Web au moyen de l'élément **<script>** et de son attribut src qui contient l'URL du fichier .js.

Code : JavaScript - Contenu du fichier hello.js

```
alert('Hello world!');
```

Code : HTML - Page Web

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
  </head>

  <body>

    <script src="hello.js"></script>

  </body>
</html>
```

On suppose ici que le fichier *hello.js* se trouve dans le même répertoire que la page Web.

Où positionner l'élément <script> dans la page html ?

La plupart des cours de JavaScript, et des exemples donnés un peu partout, montrent qu'il faut placer l'élément **<script>** au sein de l'élément **<head>** quand on l'utilise pour charger un fichier JavaScript. C'est correct, oui, mais il y a mieux !

Une page Web est lue par le navigateur de façon linéaire, c'est-à-dire qu'il lit d'abord le **<head>**, puis les éléments de **<body>** les uns à la suite des autres. Si vous appelez un fichier JavaScript dès le début du chargement de la page, le navigateur va donc charger ce fichier, et si ce dernier est volumineux, le chargement de la page s'en trouvera ralenti. C'est normal puisque le navigateur va charger le fichier avant de commencer à afficher le contenu de la page.
Pour pallier ce problème, il est conseillé de placer les éléments **<script>** juste avant la fermeture de l'élément **<body>**.

4) Les variables

Une variable est un espace de stockage sur votre ordinateur permettant d'enregistrer tout type de donnée, que ce soit une chaîne de caractères, une valeur numérique ou autres.

a) Déclarer une variable

« Déclarer une variable » veut dire de réserver à la variable un espace de stockage en mémoire. Une fois la variable déclarée, vous pouvez commencer à y stocker des données.
Pour déclarer une variable, il vous faut d'abord lui trouver un nom. Il est important de préciser que le nom d'une variable ne peut contenir que des caractères alphanumériques, autrement dit les lettres de A à Z et les chiffres de 0 à 9.
Le nom de la variable ne peut pas commencer par un chiffre et ne peut pas être constitué uniquement de mots-clés utilisés par le JavaScript. Par exemple, vous ne pouvez pas créer une variable nommée **var** car vous allez constater que ce mot-clé est déjà utilisé, en revanche vous pouvez créer une variable nommée var_.

Pour déclarer une variable, il vous suffit d'écrire la ligne suivante :

Code : JavaScript

```
var myVariable;
```

b) Affectation :

Code : JavaScript

```
var myVariable;
myVariable = 2;
```

Le signe = sert à attribuer une valeur à la variable ; ici nous lui avons attribué le nombre 2. Quand on donne une valeur à une variable, on dit que l'on fait une **affectation**, car on affecte une valeur à la variable.

c) Les types de variables :

Contrairement à de nombreux langages, le JavaScript est un langage typé *dynamiquement*. Cela veut dire, généralement, que toute déclaration de variable se fait avec le mot-clé **var** sans distinction du contenu, tandis que dans d'autres langages, comme le C, il est nécessaire de préciser quel type de contenu la variable va devoir contenir.
En JavaScript, nos variables sont typées dynamiquement, ce qui veut dire que l'on peut y mettre du texte en premier lieu puis l'effacer et y mettre un nombre quel qu'il soit, et ce, sans contraintes.

Les trois types principaux en Javascript :

Le type numérique (alias *number*) :
Il représente tout nombre, que ce soit un entier, un négatif, un nombre scientifique, etc.
Pour assigner un type numérique à une variable, il vous suffit juste d'écrire le nombre seul :

var number = 5;
Le Javascript reconnaît plusieurs écritures pour les nombres, comme l'écriture décimale **var** number = 5.5; ou l'écriture scientifique **var** number = 3.65e+5; ou encore l'écriture hexadécimale **var** number = 0x391;

Les chaînes de caractères (alias *string*) :
Ce type représente n'importe quel texte. On peut l'assigner de deux façons différentes :

Code : JavaScript

```javascript
var text1 = "Mon premier texte"; // Avec des guillemets
var text2 = 'Mon deuxième texte'; // Avec des apostrophes
```

Les booléens (alias *boolean*) :
Un booléen est un type à deux états qui sont les suivants : **vrai** ou **faux**.

Code : JavaScript

```javascript
var isTrue = true;
var isFalse = false;
```

d) Tester l'existence de variables avec typeof

Il se peut que vous ayez un jour ou l'autre besoin de tester l'existence d'une variable ou d'en vérifier son type. Dans ce genre de situations, l'instruction **typeof** est très utile, voici comment l'utiliser :

Code : JavaScript

```
var number = 2;
alert(typeof number); // Affiche : « number »

var text = 'Mon texte';
alert(typeof text); // Affiche : « string »

var aBoolean = false;
alert(typeof aBoolean); // Affiche : « boolean »
```

Simple non ? Et maintenant voici comment tester l'existence d'une variable :

Code : JavaScript

```
alert(typeof nothing); // Affiche : « undefined »
```

Voilà un type de variable très important ! Si l'instruction **typeof** vous renvoie **undefined**, c'est soit que votre variable est inexistante, soit qu'elle est déclarée mais ne contient rien.

e) Les opérateurs arithmétiques

Maintenant que vous savez déclarer une variable et lui attribuer une valeur, nous pouvons entamer la partie concernant les opérateurs arithmétiques. Vous verrez plus tard qu'il existe plusieurs sortes d'opérateurs mais dans l'immédiat nous voulons faire des calculs, nous allons donc nous intéresser exclusivement aux opérateurs arithmétiques. Ces derniers sont à la base de tout calcul et sont au nombre de cinq :

Opérateur	Signe
addition	+
soustraction	-
multiplication	*
division	/
modulo	%

Concernant le dernier opérateur, le modulo est tout simplement le reste d'une division. Par exemple, si vous divisez 5 par 2 alors il vous reste 1 ; c'est le modulo !

Exemple de Calcul :

Voici le code JavaScript permettant d'additionner 3 et 2

Code : JavaScript

```
var result = 3 + 2;
alert(result); // Affiche : « 5 »
```

f) Concaténation et à la conversion des types :

Certains opérateurs ont des particularités cachées. Prenons l'opérateur + ; en plus de faire des additions, il permet de faire ce que l'on appelle des **concaténations** entre des chaînes de caractères.

Code : JavaScript

```
var hi = 'Bonjour', name = 'toi', result;
result = hi + name;
alert(result); // Affiche : « Bonjourtoi »
```

g) La fonction prompt() :

Voici la première interaction avec l'utilisateur grâce à la fonction prompt().

Code : JavaScript

```
var userName = prompt('Entrez votre prénom :');
alert(userName); // Affiche le prénom entré par l'utilisateur
```

La fonction prompt() s'utilise comme alert() mais a une petite particularité. Elle renvoie ce que l'utilisateur a écrit sous forme d'une chaîne de caractères.
Ainsi, le texte tapé par l'utilisateur se retrouvera directement stocké dans la variable userName

Tapez ceci :

Code : JavaScript

```
var start = 'Bonjour ', name, end = ' !', result;

name   = prompt('Quel est votre prénom ?');
result = start + name + end;
alert(result);
```

h) La fonction parseInt() :

Essayons maintenant de faire une addition avec des nombres fournis par l'utilisateur :

Code : JavaScript

```
var first, second, result;

first  = prompt('Entrez le premier chiffre :');
second = prompt('Entrez le second chiffre :');
result = first + second;

alert(result);
```

Voilà le problème, tout ce qui est écrit dans le champ de texte de prompt() est récupéré sous forme d'une chaîne de caractères, que ce soit un chiffre ou non. Du coup, si vous utilisez l'opérateur +, vous ne ferez pas une addition mais une concaténation !
C'est là que la conversion des types intervient. Le concept est simple : il suffit de convertir la chaîne de caractères en nombre. Pour cela, vous allez avoir besoin de la fonction **parseInt()** qui s'utilise de cette manière :

Code : JavaScript

```
var text = '1337', number;

number = parseInt(text);
alert(typeof number); // Affiche : « number »
alert(number); // Affiche : « 1337 »
```

Maintenant que vous savez comment vous en servir, on va pouvoir l'adapter à notre code.

```
var first, second, result;

first = prompt('Entrez le premier chiffre :');
second = prompt('Entrez le second chiffre :');
result = parseInt(first) + parseInt(second);

alert(result);
```

5) Les conditions :

a) Les opérateurs de comparaison :

Comme leur nom l'indique, ces opérateurs vont permettre de comparer diverses valeurs entre elles. En tout, ils sont au nombre de huit, les voici :

Opérateur	Signification
==	égal à
!=	différent de
===	contenu et type égal à
!==	contenu ou type différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

Code : JavaScript

```
var number1 = 2, number2 = 2, number3 = 4, result;

result = number1 == number2; // Au lieu d'une seule valeur, on en
écrit deux avec l'opérateur de comparaison entre elles
alert(result); // Affiche « true », la condition est donc vérifiée
car les deux variables contiennent bien la même valeur

result = number1 == number3;
alert(result); // Affiche « false », la condition n'est pas
vérifiée car 2 est différent de 4

result = number1 < number3;
alert(result); // Affiche « true », la condition est vérifiée car 2
est bien inférieur à 4
```

Code : JavaScript

```
var number = 4, text = '4', result;

result = number == text;
alert(result); // Affiche « true » alors que « number » est un
nombre et « text » une chaîne de caractères

result = number === text;
alert(result); // Affiche « false » car cet opérateur compare aussi
les types des variables en plus de leurs valeurs
```

Les conditions « normales » font des conversions de type pour vérifier les égalités, ce qui fait que si vous voulez différencier le *nombre* 4 d'une *chaîne de caractères contenant le chiffre* 4 il vous faudra alors utiliser le triple égal ===.

b) Les opérateurs logiques :

Opérateur	Type de logique	Utilisation
&&	ET	valeur1 && valeur2
||	OU	valeur1 || valeur2
!	NON	!valeur

Exemple :

Code : JavaScript

```
var result = true || true;
alert(result); // Affiche : « true »

result = true || false;
alert(result); // Affiche : « true »

result = false || false;
alert(result); // Affiche : « false »
```

Combiner les opérateurs :

Code : JavaScript

```
var condition1, condition2, result;

condition1 = 2 > 8; // false
condition2 = 8 > 2; // true

result = condition1 && condition2;
alert(result); // Affiche « false »
```

c) La condition « if else »

Il existe trois types de conditions, nous allons commencer par la condition **if else** qui est la plus utilisée.

Code : JavaScript

```
if (true) {
    alert("Ce message s'est bien affiché.");
}

if (false) {
    alert("Pas la peine d'insister, ce message ne s'affichera
pas.");
}
```

Constitution de la condition :

La structure conditionnelle **if** est constituée :

De parenthèses qui contiennent la condition à analyser, ou plus précisément le booléen retourné par les opérateurs conditionnels ;

D'accolades qui permettent de définir la portion de code qui sera exécutée si la condition se vérifie.

Comme vous pouvez le constater, le code d'une condition est exécuté si le booléen reçu est **true** alors que **false** empêche l'exécution du code.

d) Petit intermède : la fonction confirm() :

On lui passe en paramètre une chaîne de caractères qui sera affichée à l'écran et elle retourne un booléen en fonction de l'action de l'utilisateur :

Code : JavaScript

```
if (confirm('Voulez-vous exécuter le code Javascript de cette page ?
')) {
    alert('Le code a bien été exécuté !');
}
```

Comme vous pouvez le constater, le code s'exécute lorsque vous cliquez sur le bouton OK et ne s'exécute pas lorsque vous cliquez sur Annuler. En clair : dans le premier cas la fonction renvoie **true** et dans le deuxième cas elle renvoie **false**. Ce qui en fait une fonction très pratique à utiliser avec les conditions.

e) La structure else pour dire « sinon » :

Admettons maintenant que vous souhaitiez exécuter un code suite à la vérification d'une condition et exécuter un autre code si elle n'est pas vérifiée. Il est possible de le faire avec deux conditions **if** mais il existe une solution beaucoup plus simple, la structure **else** :

Code : JavaScript

```
if (confirm('Pour accéder à ce site vous devez avoir 18 ans ou plus,
cliquez sur "OK" si c\'est le cas.')) {
    alert('Vous allez être redirigé vers le site.');
}

else {
    alert("Désolé, vous n'avez pas accès à ce site.");
}
```

Comme vous pouvez le constater, la structure **else** permet d'exécuter un certain code si la condition n'a pas été vérifiée, et vous allez rapidement vous rendre compte qu'elle vous sera très utile à de nombreuses occasions.

f) Les conditions « switch » et ternaire :

Ce sont les 2 derniers types de condition. On les découvrira plus tard. Tout peut être fait avec if et else.

g) Tester l'existence de contenu d'une variable :

Pour tester l'existence de contenu d'une variable, il faut tout d'abord savoir que tout se joue au niveau de la conversion des types. Vous savez que les variables peuvent être de plusieurs types : les nombres, les chaînes de caractères, etc. Eh bien ici nous allons découvrir que le type d'une variable (quel qu'il soit) peut être converti en booléen même si à la base on possède un nombre ou une chaîne de caractères.

Code : JavaScript

```
var conditionTest = 'Fonctionnera ? Fonctionnera pas ?';

if (conditionTest) {
    alert('Fonctionne !');
} else {
    alert('Ne fonctionne pas !');
}
```

Le code nous affiche le texte « Fonctionne ! ». Pourquoi ? Tout simplement parce que la variable conditionTest a été convertie en booléen et que son contenu est évalué comme étant vrai (**true**).

Les contenus faux : un nombre qui vaut zéro, une chaîne de caractères vide ou la valeur **undefined**.
Bon, après il est possible d'évaluer des attributs, des méthodes, des objets, etc.

h) Le cas de l'opérateur OU :

Encore un cas à part : l'opérateur OU ! Celui-ci, en plus de sa fonction principale, permet de renvoyer la première variable possédant une valeur évaluée à **true** ! Exemple :

Code : JavaScript

```
var conditionTest1 = '', conditionTest2 = 'Une chaîne de
caractères';

alert(conditionTest1 || conditionTest2);
```

Au final, ce code nous retourne la valeur « Une chaîne de caractères ». Pourquoi ? Eh bien parce que l'opérateur OU va se charger de retourner la valeur de la première variable dont le contenu est évalué à **true**.

EXERCICE

Présentation de l'exercice

But du programme : fournir un commentaire selon l'âge de la personne.
Vous devez fournir un commentaire sur quatre tranches d'âge différentes qui sont les suivantes :

Tranche d'âge	Exemple de commentaire
1 à 17 ans	« Vous n'êtes pas encore majeur. »
18 à 49 ans	« Vous êtes majeur mais pas encore senior. »
50 à 59 ans	« Vous êtes senior mais pas encore retraité. »
60 à 120 ans	« Vous êtes retraité, profitez de votre temps libre ! »

Le déroulement du code sera le suivant :
L'utilisateur charge la page Web ;
Il est ensuite invité à taper son âge dans une fenêtre d'interaction ;
Une fois l'âge fourni l'utilisateur obtient un petit commentaire.

L'intérêt de cet exercice n'est pas spécialement de sortir un commentaire pour chaque tranche d'âge, mais surtout que vous cherchiez à utiliser la structure conditionnelle la plus adaptée et que vous puissiez préparer votre code à toutes les éventualités.

Cours résumé de https://openclassrooms.com et le livre "*Dynamisez vos sites web avec JavaScript*"