Atelier N°3	*JavaScript*
Objectifs	• **Initiation au langage Javascript** • **Application**

1 Initiation au langage JavaScript

JavaScript est un langage de programmation, créé par les sociétés Netscape et Sun

Microsystems vers la fin de l'année 1995. Son objectif principal est d'introduire de petits programmes, appelés **SCRIPTS**, dans les pages HTML pour effectuer des traitements simples sur le poste de travail de l'utilisateur.

La possibilité d'inclure des programmes dans les pages HTML et de les exécuter directement sur le poste client est intéressante, car elle permet de décharger le serveur de ce travail et d'éviter les attentes des réponses aux requêtes adressées via le Réseau.

En résumé, voici ses principales caractéristiques :

- JS est un langage de programmation **structurée** qui concourt à **enrichir le HTML**, à le rendre plus "intelligent" et interactif.
- Le code JS est **intégré** complètement dans le **code HTML**, et interprété par le navigateur client
- JS contient des **gestionnaires d'événement** : il reconnaît et réagit aux demandes de l'utilisateur, comme un clic de la souris, une validation de formulaire, etc...

Mais c'est un langage limité :

- C'est un langage de script, dépendant de HTML, c'est une **extension de HTML**. Il est à mi-chemin de HTML et de Java. Il ressemble à Java, mais il est en fait très différent. Java est un langage complet, semi-compilé sur le serveur et complètement autonome du langage HTML
- La syntaxe de JavaScript ressemble à s'y méprendre à celle du langage C aussi.

1.1 Ecriture et exécution du code JS

On place du code JS dans une page HTML généralement à 2 endroits :

1. **Entre les balises <SCRIPT>....</SCRIPT>** dans la section d'en-tête, ou dans le corps de la page.

```
<HEAD>
<!-- HTML4 et (x)HTML -->
<script type="text/javascript" src="source.js">


<!-- HTML5 -->
<script src=".\source.js"></script>
</HEAD>
```

où *source.js* doit être un fichier accessible au moment de l'exécution, dans le répertoire courant ou à une adresse URL précisée. Un tel fichier externe permet de réutiliser le code dans de multiples pages, sans avoir à le réecrire.

Ce code est évalué au début du chargement de la page, mais il n'est pas forcément exécuté à ce moment là.

Le code de la fonction n'est exécuté qu'à la demande explicite de l'utilisateur, le plus souvent en réponse à un événement (voir ci-dessous).

2. **A l'intérieur d'une commande HTML qui gère un événement**.

Le code JS est alors placé sous forme d'une fonction, affectée à un gestionnaire d'événement inclus dans une balise. L'exécution du code est alors provoquée par appel d'une fonction JS (préalablement déclarée) dont l'exécution constitue une réponse à l'événement.

Un événement survient à l'initiative de l'utilisateur, par exemple en cliquant sur un bouton, ou après la saisie du texte dans un champ de formulaire.

Ecriture générale

```
<balise ... onEvenement= "fonction JS">
```

- *balise* est le nom de certaines balises, souvent des composants de formulaire
- *onEvenement* est un nouvel attribut de la balise comme **onClick**
- **fonction JS** est généralement une fonction déclarée auparavant dans une section <HEAD> <SCRIPT>...</SCRIPT> </HEAD>. Mais ça peut être aussi une suite d'instructions **"code JS"**, séparées par des virgules.

- JS fait la distinction entre majuscules et minuscules, contrairement aux balises HTML. C'est une source fréquente d'erreur.
- Pour comprendre le code, inclure des commentaires abondants :
  // pour une simple ligne de commentaires
  /* .....*/ pour les encadrer sur plusieurs lignes.
- Quand on ne définit pas de fonctions, on peut inclure le code directement dans la section <BODY>.
- Il est préférable d'inclure les déclarations des scripts dans l'en-tête du document, dans la section <HEAD>..</HEAD>.

Pour ce tp on inclura le code script directement dans le head et le body

1.2 Apprendre le JavaScript par l'exemple

1.2.1 Afficher la date et l'heure

Exemple 3.1 :

/***/
/* Ce code permet d'afficher la date de jour ainsi que l'heure, la minute et la seconde */
/***/

```
<!doctype html>
<BODY>
<SCRIPT>
var date = new Date();
//déclaration d'une variable de type Date
//la variable date contient la date du jour
var mois = date.getMonth() + 1;
//la variable mois contient le N° du mois (le résultat de la méthode
//getMonth() appliquée à l'objet date, donne un nombre de 0 à 11)
var jour = date.getDate() ;
var annee = date.getFullYear();
document.write( "Date de ce jour : " + jour + " / "+ mois + " /" + annee, "<BR />");
document.write( "Heure ", date.getHours(), " : ",date.getMinutes()," : ", date.getSeconds() );
</SCRIPT>
</BODY>
</HTML>
```

A la lecture de ce code JS, comprendre :

- La déclaration var date = new Date(); déclare un objet appelé date, de type Date, initialisé à la date du système.
- Les fonctions getDate(), getMonth() etc... s'appliquent à la variable date. Ces fonctions ne peuvent s'adresser qu'à ce type de variable Date. On dit que ce sont des méthodes de l'objet date. La notation générale est : objet.méthode()
- la *concaténation* (mise "bout à bout") de variables de type texte et de constantes "texte" est réalisée avec l'**opérateur** +
- l'instruction **document.write("texte")** est l'instruction générale d'affichage de "texte" dans le même document.
- On peut séparer les éléments à afficher par l'**opérateur** + ou plus simplement par la **virgule** ,.

1.2.2 Appel de fonction dans un formulaire

Le langage JavaScript, permet l'insertion de petits programmes dans les pages HTML, les rendant ainsi interactives. Il est ainsi possible d'effectuer des traitements à partir des données entrées par l'utilisateur, les traiter, les valider avant de les transférer au serveur.

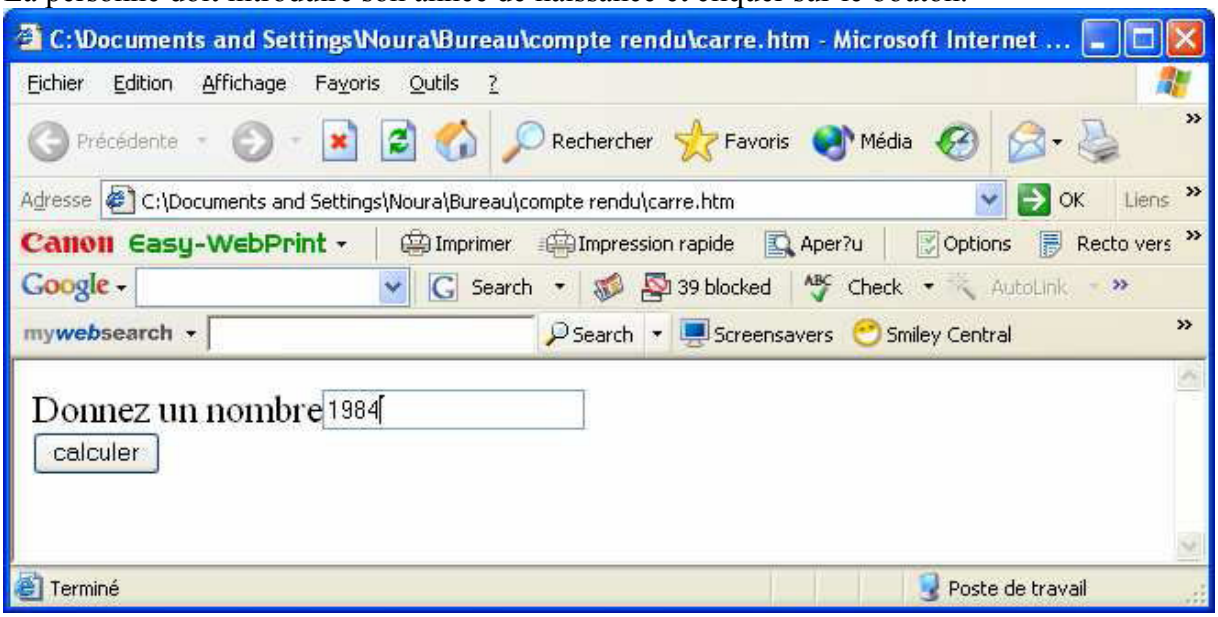
Exemple 3.2 :

```
/*****************************************************************************/
/* Ce code permet de calculer l'âge d'une personne comme */
/* La personne entre son annee de naissance dans la zone texte et elle clique sur le bouton calculer */
/* alors son age sera affiché */
/*****************************************************************************/
```

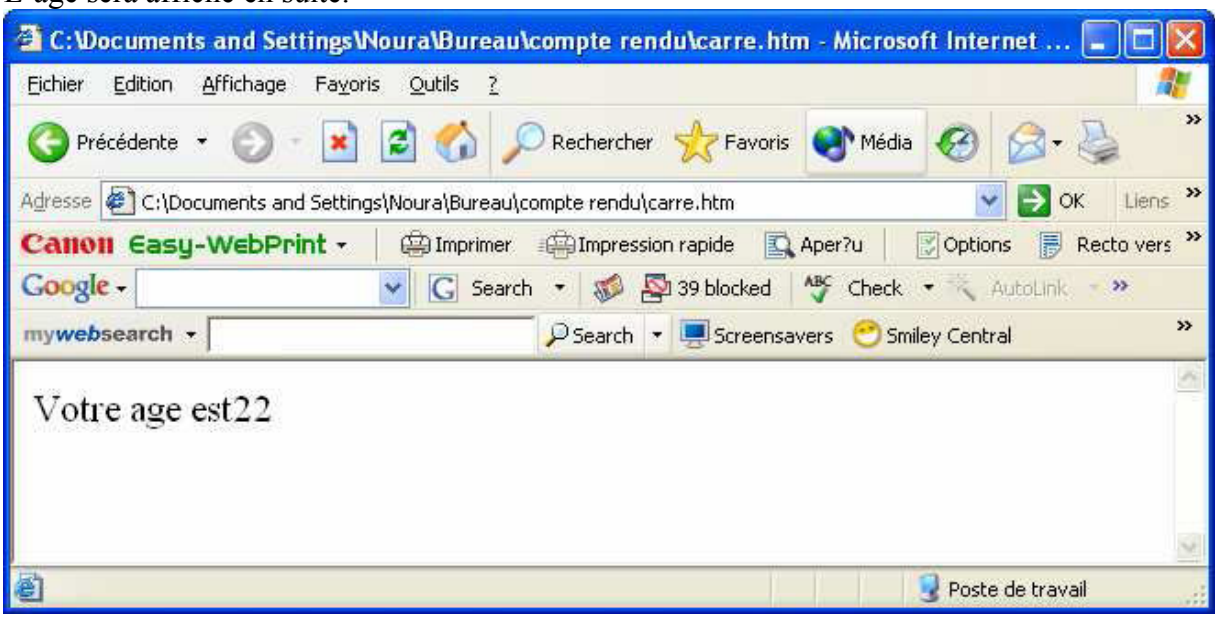
```
<!doctype html>
<head>
        <script>
        //declaration de la function age
        function age(frm)
        {
        var date = new Date();
        //date contient la date système
        var annee = date.getFullYear();
        // annee contient l'annee actuelle
        var age2=annee - frm.naiss.value;
        //calcul de l'age en accedant au champs naiss du formulaire
        //form du document en cours
        document.write("Votre age est",age2);
        //Affichage de l'age
        }
        </script>
</head>

<body>
<form>
        Donnez l'annee de votre naissance : <input type="text" name="naiss">
        <br />
        <input type="button" value="calculer" onclick="age(this.form)">
</form>
</body>
</html>
```

L'exécution de cet exemple sera sur deux étapes:
La personne doit introduire son année de naissance et cliquer sur le bouton.



L'âge sera affiché en suite.



L'objet document possède une propriété forms, qui est une collection contenant les objets formulaires présents dans la page. On peut alors récupérer l'objet formulaire désiré

document.forms['mon_formulaire']

où mon_formulaire est la valeur de l'attribut "id" de la balise form, ou bien de l'attribut "name". Il faut savoir que si la balise form a les 2 attributs, c'est la valeur de l'attribut "id" qu'il faudra utiliser pour récupérer votre objet formulaire.

Ensuite, un objet formulaire contient une propriété elements qui est une collection contenant les éléments de saisie du formulaire. On récupère donc un élément du formulaire soit la méthode **getElementById**() de l'objet **document** soit directement par la collection des éléments du formulaire de cette façon :

document.forms['mon_formulaire'].elements['mon_element']

où mon_element est la valeur de l'attribut "id" ou "name" de l'élément. Même remarque que pour les formulaires si il y a les 2 attributs, il faudra utiliser la valeur de l'attribut "id" pour récupérer l'élément.

this représente l'objet javascript en cours. Il permet de raccourcir ce chemin d'accès.
Grâce à *this.form*, on peut accéder au formulaire de l'élément en cours.

```
this.form.elements['mon_element']
```

Une fois que l'élément est atteint, il est possible de manipuler ces propriétés.
Par exemple, pour placer dans la zone de texte le mot "NOUVEAU", il faut juste écrire :

```
this.form.elements['mon_element'].value="NOUVEAU"
```

Dans le bouton, l'événement onClick reçoit le code javascript à exécuter lors du clic sur le bouton.
Le code javascript doit se mettre entre " ou entre '. Il faut faire très attention à alterner les " et '. On peut écrire : onClick='alert("Bonjour")' ou onClick="alert('Bonjour')"

Tableau récapitulatif des événements

Gest.événement	**provoqué par l'utilisateur qui ...**	**sur les objets ...**
onBlur	enlève le focus du composant	text, textarea, select
onChange	change la valeur d'un texte ou d'un composant à options	text, textarea, select
onClick	clique sur un composant ou un hyperlien	button, checkbox, radio, reset, submit
onFocus	donne le focus au composant	text, textarea, select
onLoad	charge la page dans le navigateur	balises BODY, FRAMESET
onMouseOut	la souris quitte un lien ou une ancre	balises <A HREF..>, <AREA HREF..>
onMouseOver	bouge la souris sur un lien ou une ancre	balises <A HREF..>, <AREA HREF..>
onReset	efface les saisies d'un formulaire	bouton reset
onSelect	sélectionne une zone d'édition d'un formulaire	text, textarea
onSubmit	soumet un formulaire	bouton submit
onUnload	quitte la page	balises BODY, FRAMESET

Voir les bases du javascript dans le document ci_joint :

« Les_Bases du_javascript.pdf »

Exercices

Exeercice-3-1 :

Améliorer l'exemple précédent en ajoutant une condition sur l'age :
Si l' age<26 un message s'affiche "Vous avez droit au tarif jeune étudiant.";
Si un Message "Vous n'avez droit à aucun tarif réduit." qui s'affiche

Exeercice-3-2 :

Créez une fonction **Moyenne(frm)** et un **formulaire** avec 6 champs de saisie (3 Notes et leur coefficients) et un bouton submit**.**
La fonction Moyenne calcule et affiche la moyenne en utilisant Alert
Essayiez de Traiter toutes les possibilités d'emplacement d'une fonction :
- dans <HEAD> et </HEAD> *(exercice-3-1-1.html)*
- dans le <BODY> et </BODY> *(exercice-3-1-2.html)*
- appel de fichier externe « script3-1-3.js » *(exercice-3-1-3.html)*

Exeercice-3-3 :

En utilisant deux images « eat_me.png » et « drink_me.png fournie dans ce tp, créez un script qui permet de changer la taille (horizontale et verticale) de l'image « alice.png », agrandissant l'image si on clique sur le premier et la réduisant si on clique sur le second.
Rappelez-vous que d'une façon similaire à ce qui a été fait jusqu'à présent, on peut changer le style CSS d'un élément HTML en utilisant la syntaxe x.style.propriété = "valeur";

Exeercice-3-4 :

Pour changer le valeur d'un attribut on utilise la syntaxe x.attribut = "valeur".
Soit le code HTML suivant :

```
<img src="une_image.png" alt="Une image" id="dessin">
```

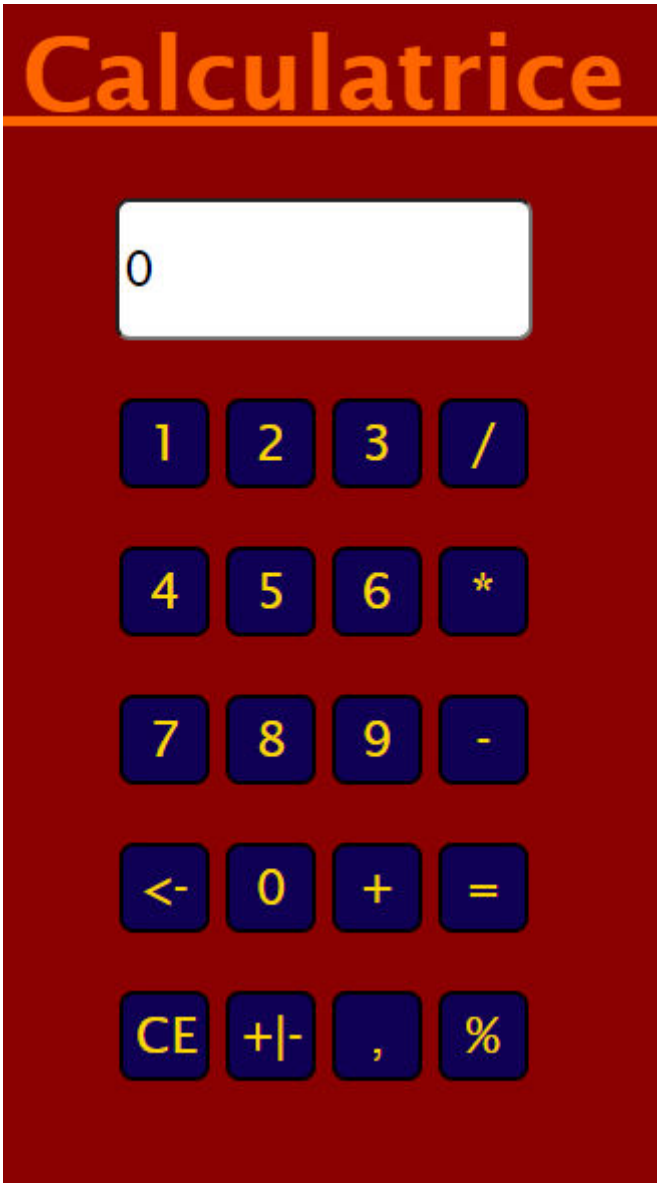
le code JavaScript :

```
document.getElementById("dessin").src = "une_autre_image.png";
```

L'image « une_image.png » sera remplacée par l'image « une_autre_image.png ».

Appliquez cet exemple en utilisant les images « **soleil.png** » et «**lune.png** » fournie dans ce TP

Application 3 :



A. Ecrire le code d'un document HTML, intitulé **calculatrice** qui ressemble à l'image ci-dessus :
   1. une zone de saisie (id= « resultat ») qui permet d'entrer une fonction à calculer à partir des boutons
   2. 10 boutons pour les chiffres de 0 à 9 et un pour , (la valeur du bouton sera automatiquement concaténer dans la zone de saisie résultat à chaque clique)
   3. 6 boutons pour les opérateurs , (la valeur du bouton sera automatiquement concaténer dans la zone de saisie résultat à chaque clique)
   4. Un bouton CE pour initialiser le champs de saisie résultat.
   5. un bouton (=) qui fait appel à une fonction **calculer (frm)** directement lorsqu'on clique dessus et qui affiche le résultat dans la zone de saisie résultat

Rq : penser à faire appel à **eval** (expression) dans la fonction calculer(frm) pour retourner le résultat de la fonction saisie.

B. Créer un fichier de style style3-1.css pour rendre la calculatrice plus agréable.