Java Script

une approche pratique

basée sur la réalisation étape par étape

des principales utilisations du language

coté client

par Xavier Braive et François Plégades

Java Script
les bases

Table des matières

Introduction

Les bases du langage

Programmation Orientée Objet

Java Script les objets natifs

Java Script les objets du navigateur

Les objets Java Script mis à disposition XML AJAX

Les feuilles de Style

Java Script

Les bases

Introduction

Cette partie doit vous permettre de bien comprendre l'environnement client serveur dans lequel le javascript se situe.

Comprendre le rôle de chaque partie prenante d'un Internet ou d'un Intranet permet de bien concevoir les programmes que vous mettrez en place et d'utiliser les langages appropriés à chaque coté (machine client, serveur, serveur de base de données).

Principes généraux - Rappel

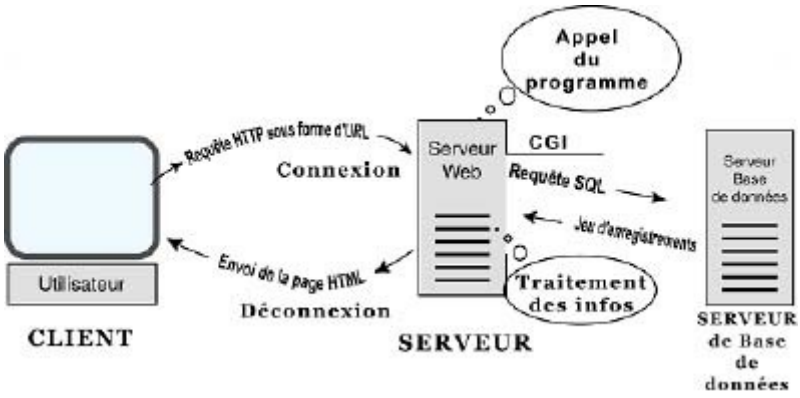
Un site internet est constitué d'un ensemble de pages html, qui sont de simples fichiers au format texte, avec une extension htm ou html.
Dans ces fichiers, des balises html définissent la façon dont l'information est structurée et doit être affichée sur le logiciel client (IE/Firefox). Ce second rôle est aujourd'hui rempli par les feuilles de style (css).
Ces fichiers sont placés sur un serveur web, qui est une machine sur laquelle tourne un logiciel capable de répondre à des requêtes http (ex.: Apache).
Une requête http contient l'adresse (URL) du fichier demandé par le client au serveur.

Demande d'une page DHTML

Une page HTML peut contenir un script programmé en Javascript ou VBScript.
Ce script sera exécuté côté client (par IE/Firefox), et pourra contenir des gestionnaires d'événements (i.e. réagir à un clic de souris, un survol, une validation de champ).
Ces scripts, associés aux calques de html4, seront la base du DHTML. Le Dynamic Html permet ainsi de modifier les propriétés des éléments d'une page, comme la position ou le contenu de ces éléments. Ces scripts sont intégrés en dur dans la page Html et sont exécutables par le client, quand la page est chargée.

Demande d'une page HTML dynamique.

Les fichiers sur serveur ne sont plus des pages html qui seront servies telles quelles, mais des scripts écrits en PHP ou ASP, qui seront exécutés sur demande et qui renverront en résultat une page html écrite à la volée.
Le serveur exécute le script au moment de la connexion. Il peut lire ou écrire des fichiers externes, envoyer une requête à une base de données et intégrer le résultat à la page qu'il envoie ensuite.



Java Script

les bases

Protocole HTTP

Principes généraux d'une relation client/serveur

Dans la relation client -> serveur , serveur -> client, le système va utiliser un mode de communication bien défini afin que n'importe quel serveur internet comprenne n'importe quel client

C'est le protocole HTTP (HyperText Transfer Protocol).
La version 0.9 était uniquement destinée à transférer des données sur Internet (des pages Web écrites en HTML). La version 1.0 du protocole permet désormais de transférer des messages avec des en-têtes décrivant le contenu du message en utilisant un codage de type MIME (Multipurpose Internet Mail Extensions).

Le but du protocole HTTP est de permettre un transfert de fichiers (essentiellement au format HTML) localisés grâce à une chaîne de caractères appelée URL (Uniform Resource Locator) entre un navigateur (le client) et un serveur Web (le site).

Principe d'une requête HTTP

Quand le client désire une page html il envoie au serveur une requête HTTP qui contient l'adresse du fichier demandé par le client au serveur (URL).

Requête HTTP sous forme d'URL

Cette requête permet au client (le visiteur du site) de demander en plus d'une page spécifique et de sa position dans l'arborescence du site, un type de connexion, de s'identifier et de spécifier ses

désirs.

En savoir plus :

D'une manière générale la requête suit la règle
Méthode URI Version-HTTP
En-tête général
En-tête de requête
En-tête d'entité
Ligne blanche obligatoire si un corps d'entité existe
Corps d'entité

- *La ligne 1 indique la méthode que le client utilise, à quel document elle doit s'appliquer et quelle est la version d'HTTP qui est utilisée.*
  Les méthodes peuvent être GET, POST, HEAD, PUT, LINK, UNLINK, DELETE, OPTION et TRACE
- *Les en-têtes généraux donnent des informations générales telles que la date ou le fait que la connexion doit ou ne doit pas être maintenue.*
- *Les en-têtes de requête donnent les informations sur la configuration du client et sur les formats de documents acceptés.*
- *Les en-têtes d'entité décrivent quel est le format des données qui sont envoyées (encodage, taille, type etc). Le client ne les utilise que lorsqu'il envoi un corps d'entité c'est à dire lors des méthodes POST ou PUT.*

ex:

```
GET/index.html#haut ?log=Toto&mdp=123 HTTP/1.1
Accept: image/gif, image/jpeg, */*
Accept-language:  fr,en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: monsite.fr
Connection: Keep-Alive
```

Codage de l'URI

Internet utilise deux termes pour désigner la même chose.
URL (Uniform Resource Locator) et URI (Uniform Resource Identifier).
Les URL sont un sous-ensemble des URI. Une URI doit permettre d'identifier une ressource de manière permanente, même si la ressource est déplacée ou supprimée. Jusqu'à présent, les URL sont les seules URI ayant trouvé une application pratique.
Pratiquement que l'on utilise le terme URL ou le terme URI on parle de la même chose : l'adresse de la page demandée

Soit une URL de la forme :

```
HTTP://www.monsite.fr:80/coucou/index.html?log=Toto&mdp=123#haut
```

Le navigateur va interpréter cette URL de la façon suivante

HTTP://	Protocole de communication utilisé (Ici Hyper Text Transfert Protocol protocole du html). Le protocole défini la manière dont le client et le serveur vont communiquer. HTTP pour une communication sous forme de liens hypertexte (le web) FTP (File Transfert Protocol) pour les transfert entre le client et le serveur. C'est le protocole utilisé pour mettre en place ou récupérer les sites sur le serveur à partir du site développé en interne. NNTP (Network News Transport Protocol), pour les news. C'est le protocole Internet précisant la manière dont les messages des newsgroups sont distribués, récupérés, cherchés et publiés - et donc celui qui décide des interactions entre serveurs de newsgroups et outils de lecture de ces newsgroups. SMTP (Simple Mail Transfert Protocol) pour transférer le courrier d'un serveur à un autre en connexion point à point. POP (post office protocol) pour récupérer le courrier depuis une boite aux lettres électronique jusqu'au client de courrier (thunderbird ou outlook). Le protocole POP actuel est le 3 : POP3
www.monsite.fr	serveur à contacter
:80	port de connexion sur le serveur. En HTTP 80 est le port par défaut et peut donc être omis
/coucou/	répertoire à emprunter pour trouver le document. La racine du site est /
index.html	document demandé au serveur. Par défaut c'est index.html ou index.htm
?log=Toto&mdp=123	arguments sous forme nom=valeur séparés par des & et codé (*). C'est la transmission des variables en méthode GET
#haut	repère d'affichage dans la page (ancre)

* Dans une URL les espaces ou certains caractères particuliers (comme & ou les caractères ascii 128 à 255) sont codés sous forme "%valeur ascii en hexadécimal du caractère" (par exemple ê = ascii 234 ; 234=.EA en hexadécimal donc ê s'écrit %EA)

Une URL ne peut pas dépasser un certain nombre de caractères. C'est une limite de la méthode GET

L'adresse dans l'URL est soit absolue (c'est à dire complète), soit relative (c'est à dire partielle). Quand l'adresse est relative, tout ce qui manque est semblable à l'URL de la page en cours.

Principe d'une réponse serveur en HTTP

Le serveur va répondre en recherchant la ressource demandée, en créant une en-tête de serveur

(header) et en joignant en le séparant des valeurs de l'en-têtes par une ligne blanche le document demandé.

Réponse du serveur (entête + document)

D'une manière générale la requête suit la règle

Version-HTTP - Code d'état - Explication
En-tête général
En-tête de réponse
En-tête d'entité
Ligne blanche obligatoire si le corps d'entité existe
Corps d'entité

- La ligne 1 indique la version d'HTTP qui est utilisée, le code de résultat de la requête (200 requête accomplie avec succès) et une explication humainement compréhensible du code d'état (OK).

- Les en-têtes généraux donnent comme pour le client des informations générales telles que la date ou le fait que la connexion est ou n'est pas maintenue

- Les en-têtes de réponse donnent les informations sur la configuration du serveur, informe le client sur les méthodes supportées et les autorisations nécessaires, ou demande au client de réessayer plus tard.

Principaux Codes d'état :

10x - Message d'information. Ces codes ne sont pas utilisés dans la version 1.0 du protocole

20x - Réussite. Ces codes indiquent le bon déroulement de la transaction
ex : 200 OK
La requête a été accomplie correctement

30x - Redirection. Ces codes indiquent que la ressource n'est plus à l'emplacement indiqué
ex : 301 MOVED
Les données demandées ont été transférées à une nouvelle adresse

40x - Erreur due au client. Ces codes indiquent que la requête est incorrecte

ex :
401 UNAUTHORIZED
Le paramètre du message donne les spécifications des formes d'autorisation acceptables. Le client doit reformuler sa requête avec les bonnes données d'autorisation
403 FORBIDDEN
L'accès à la ressource est tout simplement interdit
404 NOT FOUND
Le serveur n'a rien trouvé à l'adresse spécifiée.

50x - Erreur due au serveur. Ces codes indiquent qu'il y a eu une erreur interne du serveur
ex :
500 INTERNAL ERROR
Le serveur a rencontré une condition inattendue qui l'a empêché de donner suite à la demande

Le navigateur

Le navigateur (ou browser) est un programme tournant sur la machine cliente. Son rôle est de recevoir les pages écrites en HTML et de traduire les balises pour en faire une présentation conforme grâce à son DOM (Document Object Model).
C'est lui qui va gérer la navigation et le dialogue avec le serveur.

Lorsque l'on tape une adresse internet sous forme d'URL dans le navigateur, celui ci appelle le serveur et tente d'établir la connexion.

Si celle ci réussi le serveur envoie une page par défaut (index) ou la page demandée.

Le navigateur va alors lire la page HTML qu'il a reçu du serveur et interpréter les balises qu'il rencontre.
Il affiche au fur et à mesure la page dans sa fenêtre d'affichage.
Lorsqu'il rencontre des éléments définis par une URL comme des images, des feuilles de styles ou des fichiers externes de scripts il les mémorise et une fois la page affichée il appelle le serveur pour récupérer ces éléments.

Un certain nombre d'éléments d'une page nécessite un programme extérieur pour être afficher dans le navigateur (Flash, vidéo, pdf etc.)

Java Script
les bases

Il existe 2 types de programme extérieur.

- Les activeX propre à Windows et à internet explorer sous Windows
- Les plugins pour les autres navigateurs.

Aucune normalisation des interpréteurs présents dans les navigateurs n'ayant été faite, les résultats de la lecture d'une page peuvent dépendre du navigateur et de sa version.

Ces problèmes sont particulièrement sensibles au niveau des interpréteurs de feuilles de style et des interpréteurs de scripts javascript.
Il est donc nécessaire de tester les pages sur les principaux navigateurs du marché

Internet explorer Windows
Firefox Windows et MAC
Internet explorer MAC (abandonné par Microsoft)
Safari Windows et MAC

Le DOM (Document Object Model)

Le Document Object Model (ou DOM) est une recommandation du W3C qui décrit une interface indépendante de tout langage de programmation et de toute plate-forme, permettant à des programmes informatiques et à des scripts d'accéder ou de mettre à jour le contenu, la structure ou le style du documents. Celui ci est alors actualisé et peut ainsi devenir dynamique.

A partir des balises de la page HTML, le DOM va créer des objets qu'il mettra à la disposition des langages (scripts, feuilles de styles). Ceux ci pourront donc agir sur n'importe quel élément de la page.

La construction sous forme arborescente de la page permet une création de chaque élément sous forme d'objet.

Pour bien gérer cela il faut une rigueur d'écriture qui n'était pas mise en place dans le HTML.
C'est pour cette raison que le HTML est aujourd'hui abandonné au profit du XHTML qui est du HTML avec une rigueur de XML.

Les bases du langage

Cette partie doit vous permettre de comprendre les bases du langage javascript.
Vous y verrez la syntaxe ainsi que les différents éléments tel que variables, opérateurs et fonctions.
Grâce à une série d'exercices de base, vous devez savoir, en fin de chapitre, écrire des scripts simples en utilisant les composants du langage javascript.

Formats et conventions

FICHIERS

Les fichiers sont sous forme texte simple, ont une extension .js ou sont inclus directement dans la page html.

BALISES <script>

La balise script indique au navigateur que son contenu est à traiter comme étant des instructions javascript.
Il les exécute à l'aide de son analyseur syntaxique javascript.
Comme l'analyse du code dépend du navigateur, il existe des différences d'interprétation. Le résultat final dépend donc du navigateur.

```
<script language="javascript" type="text/javascript" >
Instructions
</script>
```

ou pour un fichier externe

```
<script language="javascript" type="text/javascript"
src="malib.js"></script>
```

Fin d'instruction :

Les ligne d'instruction se terminent soit par point-virgule (;), soit par un retour à la ligne

Echappement

Le signe d'échappement (\) sert à ne pas interpréter un caractère:

Echappements particuliers
- \n nouvelle ligne (newline)
- \t tabulation
- \r retour chariot (Carriage return)
- \xdd caractère 8bit encodé Latin-1 en hexadecimal dd
- \udddd caractère 16bit encodé Unicode en hexadecimal
- \\ écrit le caractère \ (antislash)

ex :

```
<script language="javascript">
document.write("je dis \"bonjour le Monde !\u0022");
</script>
```

Commentaire :

On peut insérer des commentaires dans le code javascript soit sous forme de bloc allant de /*…….*/ ou soit sous forme ligne en utilisant // en début de ligne ou à partir de l'endroit ou on commente la ligne.

Sortie vers le client :

document.write()
Ex:

```
<script language="javascript">
document.write("je dis \"bonjour le Monde !\"");
</script>
```

Attention document.write() insére dans une page html du texte uniquement lors du chargement de la page.
Si vous utilisez document.write() une fois la page chargée, celle ci est effacée et remplacée par le contenu de document.write()

Variables

Un nom de variable doit commencer par une lettre, le caractère $ ou le caractère souligné (_), suivis de lettres, $ chiffres ou _.
Les variables ne doivent pas être déclarées et ne sont pas fortement typées: elles peuvent contenir une chaine de caractère, une valeur numérique (entière ou décimale) ou booléenne (true = 1 false = 0).
Lors des opérations javascript transtype automatiquement en fonction de l'opération demandée.
Ceci peut créer des résultats complètement éronés

```
a="1"; b=1; c=true;
alert(a+b+c) // donne 11true
alert(b+c+a) // donne 21
```

Attention: les variables sont sensibles à la casse (ville est différent de Ville et de VILLE).

Assignation :

Elle se fait à l'aide de l'opérateur d'assignation =

```
ville = "Bruxelles";    // (string)
ZIP = 85750;            // (integer ou int)
taux = 6.55957;         // (float ou double)
reponse = false;        //(boolean)
```

Transtypage :

On peut forcer le type d'une variable en utilisant les fonctions de transtypage

parseFloat() pour transtyper en nombre décimal
parseInt() pour transtyper en nombre entier
toString() pour transtyper en chaine de caractères
Ex :

```
a = "1"; // type string
b = 1 ; // type integer
document.write( a+b+"<br />"); //affiche 11
a = parseInt(a) ;
document.write( a+b+"<br />"); //affiche 2
```

ou en transformant la variable en objet type
Number()
String()
Boolean()
Array()
Ex :

```
a = "1"; // type string
b = 1 ; // type integer
document.write( a+b+"<br />"); //affiche 11
b = Boolean(b) ;
document.write( a+b+"<br />"); //affiche 1true
```

Variables variables

Les variables variables permettent de spécifier un nom de variable à partir d'une variable. On utilise la fonction eval("chaine de caractères") qui interprète la chaine passé en argument comme du code javascript
Ex :

```
ville1 = "Tokyo";
ville2 = "Nagoya";
ville3 = "Kobe";
numero = 2;
eval("mavar=ville"+numero);
document.write("ville"+numero+" vaut : "+mavar);
```

Donnera Nagoya.

Tableaux (Listes)

En javascript les tableaux sont des objets qui sont créés avec new Array() ou l'opérateur [].
Ils ont une seule dimensions.
Pour construire un tableau à n dimensions on crée des tableaux de tableaux .
Ex :

```
dessert = new Array();
dessert[0] = "glace";
dessert[1] = "mousse";
dessert[2] = "fruit";
document.write(dessert[2]);
```

Le constructeur Array permet de créer soit un tableau vide soit un tableau pré rempli

```
jours = new Array("lundi", "mardi", "mercredi", "jeudi");
```

Les éléments sont indicés à partir de zero:

```
jours[0] vaut lundi
```

```
document.write(jours[2]); //donne mercredi
```

Et mélanger pour avoir des tableaux à plusieurs dimensions:

```
menus = new Array(
new Array("salade","steack","tarte"),
new Array("foie de volaille", "homard", "forêt noire"),
new Array("bisque", "saumon","tarte tatin")
);
$menus[1][1] donne "homard"
document.write(menus[2][1]);//donne saumon
```

Ajouter ou modifier un élément :

Pour ajouter un éléments il suffit de donner son indice et sa valeur

```
jours[4] = "vendredi";
```

Pour modifier un élément existant, se référer à son indice

```
menus[0][0]= "jus de mangue";
```

Application :

Créer un tableau pays indicé sur le code téléphonique
(France 33, Belgique 32, Mexique 52, Japon 81)
Créer un index variable i
Demander l'affichage du 52 (Mexique).

Corrigé:

```
i=52;
pays=new Array()
pays[32] = "Belgique";
pays[33] = "France";
pays[52] = "Mexique";
pays[81] = "Japon";

document.write(pays[i]);
document.write("<br />"+pays.length);
document.write("<br />"+pays[10);
```

On constate que le nombre d'items créés dans le tableau est l'indice le plus élevé (82) + 1.
Les items non attribués ont pour valeur **undefined**

Opérateurs

Opérateurs arithmétiques :

+, -, *, /, % (modulo)
remarque : la décimale est le point (.).
exemple:
20%7

donne 6

Incrémentation :

x++; est équivalent à x = x + 1;
x--; est équivalent à x = x - 1;
x += 10; est équivalent à x = x + 10;

D'une manière générale
x (op)=val est équivalent à x=x (op) val

Opérateur de chaîne (concaténation) :

La concaténation permet de coller deux chaînes.
"bon" + "jour"

On peut utiliser l'autoconcaténation (+=)

```
lemot = "Bon";
lemot += "jour";  // lemot contient "Bonjour"
```

Opérateur de comparaison

< plus petit et <= plus petit ou égal
> plus grand et >= plus grand ou égal
== semblable (égalité de valeurs)
=== semblable et de même type (valeurs et types)
!= différent de (a="1";b=1;a!=b renvoi false)
!== non identique à (a="1";b=1;a!==b renvoi true)

Combinaisons Logiques

a && b Vrai si a ET b sont vrais.
a || b Vrai si a OU b est vrai
a ^ b Vrai si a OU b est vrai, mais pas les deux.
! Vrai si a est faux.

Les contrôle de flux (conditionnels et boucles)

Cette partie doit vous permettre d'utiliser les contrôles conditionnels pour orienter vos scripts en fonction des contextes et d'utiliser les différents types de boucles pour automatiser les étapes répétitives en interrogation ou affichage de données.
A la fin de ce chapitre vous connaîtrez les syntaxes et possibilités du javascript au niveau du contrôle de flux.

Contrôles de Flux

Les contrôles de flux sont destinés à modifier le cours d'un programme en fonction de certaines conditions. Ces conditions sont testées sur des vérités logiques:

Test de condition:

Si:

```
if (condition)
{
        instructions…
}
```

Sinon:

```
if (condition)
{
        instructions…
}
else
{
        autres instructions…
}
```

Sinon Si:

```
if (condition)
{
        instructions…
}
else if (condition2)
{
        autres instructions…
}
else
{
        autres instructions…
}
```

Switch case:

Dans certains cas, cette structure est plus efficace que des imbrications de si.
Le break sert à sortir de la structure sans passer par le traitement par défaut:

```
switch(variable)
{
```

```
        case "valeur1":
                instructions1;
                break;
        case "valeur2":
                instructions2;
                break;
        default:
                instructions par défaut;
}
```

Traitements en Boucle

Les traitements en boucles permettent d'exécuter des instructions de manière répétitive. Il en existe plusieurs types:

while (tant que condition vraie)

Peut ne pas être exécutée (si la condition n'est pas vérifiée au départ)

```
while(condition)
{
        instructions;
}
```

Ex:

```
i = 0;
while (i<=20)
{
     document.write(i);
     i++;
}
```

si i = 21 au départ, rien ne sera exécuté

Do .... While

La condition est évaluée en fin de boucle (donc exécutée au moins une fois)

Ex:

```
i = 0;
do
{
     document.write(i);
     i++;
}
while (i <= 20);
```

si i = 21 au départ, la boucle sera quand même exécutée 1 fois.

For

La boucle FOR s'exécute un nombre déterminé de fois.

Ex :

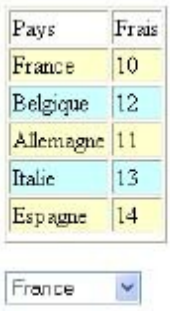
```
for (i = 10; i <= 20; i++)
{
     document.write(i+"<br />") ;
```

```
}
```

APPLICATION

- Créer une variable tableau de frais de port par pays, sous la forme:
tab_pays[0]= ["Allemagne",11] ;
tab_pays[1]= ["France",10] ;
etc.
- Créer une boucle de présentation sous forme de tableau HTML, avec une colonne "pays" et une colonne "Frais".
- Créer la liste déroulante correspondante.
- Alterner les couleurs de lignes (bgcolor de la balise tr)

remarque : la propriété length renvoie la taille d'un tableau. (tab_pays.length)



Correction

```
<html>
<head>
<title>calcul ttc</title>
<script language="javascript" type="text/javascript">
tab_pays = new Array(
["Allemagne",11],
["Belgique",12],
["Espagne",14],
["France",10],
["Italie",13],
["Angleterre",13]
);
coul = new Array("FFFFCC","CCFFFF");
aff_tab = aff_liste = "";

for (i = 0; i < tab_pays.length; i++)
{
     aff_tab += "<tr bgcolor=\"#"+coul[i%2]+"\">" +
            "<td>"+tab_pays[i][0]+"< /td>"+
            "<td>"+tab_pays[i][1]+"</td>"+
            "</tr>" ;
     aff_liste += "<option value=\""+tab_pays[i][1]+"\">"+
            tab_pays[i][0]+
            "< /option>";
}
```

```
</script>
</head>
<body>
<h1>Nos Frais de port</h1>
<table border="1" cellspacing="0">
<tr>
<td>Pays</td>
<td>Frais</td>
</tr>
<script language="javascript" type="text/javascript">
     document.write(aff_tab);
</script>
</table>
<h2>Choisissez votre pays</h2>
<form>
Pays :
<select name="fdp">
<option value="">Choisissez votre pays</option>
<script language="javascript" type="text/javascript">
     document.write(aff_liste);
</script>
</select>
     <input type="submit" value="Pays" />
</form>
</body>
</html>
```

Fonction

Une fonction isole une partie du code pour le rendre réutilisable. Elle utilise des arguments qui lui sont transmis lors de son appel, et peut envoyer un résultat avec return.
Ex :

```
function euros (francs)
{
     resultat = francs / 6.55957;
     return(resultat);
}

document.write( euros(10));
document.write("<br />"+ euros(20));
```

Une fonction peut accepter plusieurs arguments, séparés par une virgule.

Portée

Attention: les variables définies dans une fonction ont une portée globale. (accessibles en dehors de la fonction).
Pour pouvoir ne pas changer les variables externes à la fonction il faut déclarer dans la fonction les variables avec var.
Ex :

```
taux = 6.55957;
```

```
val = 10;
function euros (francs)
{
     val = francs / taux;
     return(val);
}
document.write(euros(val));
document.write("<br />"+euros(val));

taux = 6.55957;
val = 10;
function euros (francs)
{
     var val = francs / taux;
     return(val);
}
document.write(euros(val));
document.write("<br />"+euros(val));
```

Application :

Écrire une fonction qui calcule un prix TTC sur base de 2 arguments:
- le prix HT
- le taux (5.5 ou 19.6 %)

Corrigé:

```
<html>
<head>
<title>calcul ttc</title>
<script language="javascript" type="text/javascript">
function ttc (ht, taux)
{
     var resultat = ht * (1+ taux / 100);
     return(resultat);
}
ht = 158;
tva = 5.5;
ttc = ttc(ht,tva);
</script>
</head>
<body>
<script language="javascript" type="text/javascript">
     document.write(ht+" euros hors taxe donnent avec une TVA de "+tva+" %
"+ttc+" euros TTC.");
</script>
</body>
</html>
```

Librairies et inclusion de fichier

Une librairie est un fichier javascript qui ne contient que des fonctions. Son extension est .js. On y fait appel par la balise script en donnant l'adresse du fichier :

```
<script language="javascript" type="text/javascript" src="ma_librairie.js">
</script>
```

Programmation Orientée Objet

En programmation non objet, le corps des scripts est toujours composé d'une suite d'instructions, faisant éventuellement appel à des fonctions.
La POO a permis de changer notre approche de la programmation en la rapprochant de notre mode de pensée.
Notre pensée fonctionne à travers des abstractions de la réalité qui nous entoure.
Par exemple, pour nous, une table est un objet comportant un certain nombre de propriétés (un support sous forme de un ou plusieurs pieds, un plateau, etc). En plus de ces propriétés elle possède un certain nombre de fonctions. L'ensemble des propriétés et des fonctions lié à l'objet table est propre à toutes les tables. Il existe des tables qui ont en plus des propriétés et fonctions des tables des propriétés et des fonctions qui leur sont propre (par exemple l'objet guéridon ou l'objet bureau).
On a donc une nouvelle définition qui dira que l'objet bureau est un objet table qui possède en plus ….. L'objet bureau hérite donc de l'objet table toutes ses fonctions et propriétés.
Une fois que l'objet à été classifié on peu utiliser des objets qui répondront à la classification définie et qui seront donc des tables ou des bureaux.
Chaque instance de table ou de bureau aura alors sa vie propre et pourra acquérir des propriétés ou des fonctions qui lui seront propre en plus de celle de la classe auquel ils appartiennent

Les avantages de la POO

un code plus clair à lire et plus facile à modifier
une réutilisation des classes pour d'autres scripts

La POO a deux buts principaux:

- faciliter la réutilisation du code que vous avez déjà écrit (héritage).
- encapsuler des données (propriétés) et les traitements (méthodes).

Pour cela l'objet utilise un mode d'emploi de l'objet, la classe, qui indique les variables internes à l'objet (les propriétés) et les fonctions intrinsèques utilisables avec cet objet (les méthodes)

Lors de la création de la variable objet, une fonction pourra initialiser certaine valeur de l'objet. C'est la fonction constructeur

Bien que construit en objet et mettant à disposition du développeur que des objets, le javascript a un modèle objet très basique.

Pour voir comment l'objet fonctionne en javascript prenons un exemple.
Nous allons gérer les commande de pizza.
Le client est généré par son nom et son numéro de table.
Il peu commander, ajouter, demander l'addition.
Chaque commande a comme propriétés le nombre de pizzas royale (pizro), le nombre de pizzas regina (pizre), le prix des royale (prixro) , le prix des regina (prixre), le prix total

Création de la classe

Constructeur

```
function commande_pizza(lenom, num_table) {
this.prixro=12;
this.prixre=8.5;
this.pizro=0;
this.pizre=0;
this.nom = lenom;
this.num = num_table;
}
```

prixro, prixre, pizro, pizre, nom et num sont les propriétés de la classe commande_pizza.

Définir les méthodes d'un objet

Il existe 3 sortes de méthodes en langage objet, les méthodes privées, qui ne sont appelées qu'au sein de la class et inaccessibles en dehors de celle ci, les méthodes héritées qui sont accessible au sein de la class et de ses enfants et les méthodes publiques, qu'on peut appeler à partir de n'importe ou dans le script.

En Javascript tout est publique.

Création des méthodes

Pour créer une méthode on peu la créer lors de l'instancialisation de l'objet (constructeur)

Création lors de l'appel du constructeur

```
function commande_pizza(lenom, num_table)
{
     this.prixro = 12;
     this.prixre = 8.5;
     this.pizro = 0;
     this.pizre = 0;
     this.nom = lenom;
     this.num = num_table;
     this.ajoute = ajoute;
}
function ajoute(quoi,combien)
{
     switch (quoi){
          case "royale":
               this.pizro += combien;
               return true
          case "regina":
               this.pizre += combien;
               return true
     }
```

```
        return false
    }
```

Attention : dans la création de la méthode this.ajoute = ajoute; on n'indique pas les parenthèses (),

Utilisation de la méthode

```
client2.ajoute("regina",5);
```

Création hors appel du constructeur : utilisation de la propriété « prototype »

```
function affiche()
{
    return "Client : " + this.nom + " (Table num&eacute;ro " + this.num +
        ")<br />Nombre de pizza Royale : " + this.pizro +
        "<br />Nombre de pizza Regina : "+ this.pizre ;
}
commande_pizza.prototype.affiche = affiche;
// ou
commande_pizza.prototype.facture = function()
{
    total = (this.pizro*this.prixro)+(this.pizre*this.prixre);
    return this.affiche() +"<br /><br />Total de votre commande : " +
        total + " &euro;";
}
```

Attention :
Si on veut ajouter de nouvelles méthodes à des objets Javascript prédéfinis on ne peut utiliser que le prototypage

L'opérateur «this» indique que les variables sont des variables appartenant à commande_pizza.
Javascript n'a pas de type class ou plus exactement le type class est implicite.
C'est la fonction commande_pizza() qui va initialiser la class commande_pizza en lui donnant des propriétés.
commande_pizza() est le constructeur de la classe commande_pizza.
Javascript n'a besoin que du constructeur pour créer une class.

Création d'une instance de l'objet

On utilise pour ça l'opérateur new constructeur(params) .

```
var client1 = new commande_pizza ("5", "Dupond");
var client2 = new commande_pizza ("2", "Dubois");
//  Utiliser l'objet
Client1.pizro += 2;
Client1.pizre += 5;
Client2.pizro += 3;
Client2.pizre += 1;
document.write(Client1.pizro); // écira "2"
document.write(Client2.pizre); // écrira "1"
```

L'héritage

Il est possible en Javascript d'implémenter des sous-classes d'une classe et de les faire hériter des propriétés et des méthodes de la classe-mère.

Reprenons notre exemple de pizza. On va créer une class mère qui servira à gérer la mise à disposition des pizzas (type de pizza, prix) et qui permettra de rajouter des type de pizza.

La class commande_pizza héritera de cette class pizza et gérera les commandes des clients

La classe pizza

```
function pizza()
{
     this.prixroyale = 12;
     this.prixregina = 8.5;
     this.royale = 0;
     this.regina = 0;
     this.liste_pizza = new Array("royale","regina");
     this.ajoute_piz = ajoute_piz;
}
function ajoute_piz(quoi,prix)
{
     eval("this." + quoi + " = 0");
     eval("this.prix" + quoi + " = " + prix);
     this.liste_pizza[this.liste_pizza.length] = quoi;
     return true
}
```

La sous-classe commande_pizza

```
commande_pizza.prototype = new pizza();
function commande_pizza(num_table, lenom)
{
     this.parent = pizza;
     this.parent();
     this.nom = lenom;
     this.num = num_table;
     this.ajoute = ajoute;
}
function ajoute(quoi, combien)
{
     eval("this." + quoi + " += " + combien);
}
function affiche()
{
     aff = "Client : "+this.nom + " (Table num&eacute;ro " +
         this.num + ")<br />" ;
     for (i = 0; i < this.liste_pizza.length; i++)
     {
          aff += "<br />Nombre de pizza " + this.liste_pizza[i] + " : ";
          eval("aff += this." + this.liste_pizza[i]);
     }
     return aff;
```

```
}
commande_pizza.prototype.affiche = affiche;
commande_pizza.prototype.facture = function()
{
     var total = 0;
     for (i = 0; i < this.liste_pizza.length; i++)
     {
          eval("total += this." + this.liste_pizza[i] +
          " * this.prix" + this.liste_pizza[i] + ";");
     }
     return this.affiche() + "<br /><br />Total de votre commande : " +
          total + " &euro;";
}
```

On fait hériter la class commande_pizza de la class pizza grâce à
`commande_pizza.prototype = new pizza(); .`
ainsi que par les 2 premières lignes du constructeur commande_pizza ().

On crée le client en donnant son numéro de table et son nom
`client1 = new commande_pizza("5", "Dupond");`
On utilise les propriété de pizza dans commande_pizza

```
client1.royale += 3;
client1.regina += 1;
client1.ajoute("regina",5);
```

on utilise la méthode ajoute_piz de pizza pour créer une nouvelle pizza en donnant son nom et son prix via commande_pizza
`client1.ajoute_piz("napo",6.5);`
On utilise cet ajout dans commande_pizza

```
client1.ajoute("napo",3);
document.write(client1.facture());
```

donnera :

Client : Dupond (Table numéro 5)
Nombre de pizza royale : 3
Nombre de pizza regina : 6
Nombre de pizza napo : 3
Total de votre commande : 106.5 €

Propriétés et méthodes statiques

Les propriétés et les méthodes sont dites statique lorsqu'elles n'appartiennent pas aux instances objets mais à la classe elle même. On peut donc les utiliser sans créer d'objets avec new.

L'exemple type de classe statiques est l'objet Math de javascript qui met à disposition les méthodes et propriétés mathématiques.

Pour rendre une propriété ou une méthode statique il suffit de l'ajouter au constructeur

Ex :

```
function maClass()
{
     this.maProp = "prop1";
```

```
        this.maMethode = function(){....}
}
//création d'une propriété statique sur la classe maClass
maClass.maPropStat = PropStat1 ;
//création d'une méthode statique sur la classe maClass
maClass.maMethodeStat = function(){....}
```

Utilisation de l'objet

```
//instancialisation de l'objet
maVarObj = new maClass() ;
//utilisation de l'objet
alert(maVarObj.maProp) ;
maVarObj. MaMethode() ;
//utilisation des méthodes et propriétés statiques
alert(maClass.maPropStat);
maClass.maMethodeStat();
```

Java Script

Les objets natifs

Les objets du javascript

Cette partie doit vous permettre de connaître les grandes fonctions du javascript. Ces fonction sont mise à disposition sous forme d'objets possédant des propriétés et des méthodes
Vous y verrez la syntaxe des principales fonctions mathématiques, de texte et de date et leur utilisation.
En fin de chapitre vous devez connaître les possibilités offertes par le javascript pour gérer les calculs, l'écriture et les dates.
(Il ne s'agit pas de connaître par cœur ces fonctions mais de savoir quelles possibilités sont mises à disposition pour écrire des scripts en javascript)

Objets javascript internes

Objets de typage et de manipulation des types:

Object
Array
String
Number
Boolean

Objets de manipulations

this
Function
arguments

Objets d'opérations

Date
Math
RegExp

Objets de typage et de manipulation des types

Tous les types, du plus simple (boolean) au plus complexe (object) sont des objets.
Certain comme Boolean, Number, Object n'ont que peu d'intérêt en dehors d'initialiser un type ou de convertir en un type.
D'autre comme Array et String mettent à disposition du développeur une suite de propriétés et de méthodes permettant d'utiliser et de manipuler ces types

L'objet Array

Il est initialiser via new Array() ou []

```
Mon_tableau=new Array(val1,val2,val3) ;
Mon_tableau=[val1,val2,val3];
```

Les propriétés

length renvoie la taille du tableau

```
Mon_tableau.length //renvoit 3
```

Les méthodes

concat(tableau) concaténe deux tableaux

```
tableau_result = tableau1.concat(tableau2);
```

Ex :

```
tab1 = Array("dimanche","lundi","mardi","mercredi");
tab2 = Array("jeudi","vendredi","samedi");
tab = tab1.concat(tab2);
```

join("separateur") converti le tableau en une chaine de caractères en utilisant "séparateur" comme séparateur de valeur (par défaut c'est la virgule)

Ex :

```
tab = Array("jeudi","vendredi","samedi");
alert(tab.join("µ"));
```

push(élément1,élément2) ajoute un ou des éléments à la fin d'un tableau

Ex

```
tab = Array("dimanche","lundi","mardi","mercredi");
tab.push("jeudi","vendredi","samedi");
```

pop() enlève le dernier élément et le renvoit.
shift() enlève le premier élément et le renvoit,
unshift(élément1,élément2) ajoute un ou des éléments au début d'un tableau.
slice(debut, longueur) renvoie un sous-tableau commençant à l'indice debut de longueur éléments. Le tableau d'origine n'est pas changé

```
soustab = tab.array_slice(1, 2)
```

renverra dans soustab (indice 1, longueur 2) soit Array("lundi","mardi")
splice(debut, longueur, remplacement) Remplace les "longueur" éléments à partir de l'indice "debut" par "remplacement" et retourne les valeurs supprimées
reverse() inverse l'ordre des éléments
sort(type_tri) opère un tri alphabétique dans l'ordre croissant. Les nombre sont considérés comme des chaines de caractères (1258 sera donc placé avant 2)
toString() renvoie le tableau sous forme de chaine de caractères

L'objet String

Les chaine de caractères sont automatiquement convertie en objet String

Les propriétés

length renvoie la longueur de la chaine

```
alert("toto".length) // renvoie 4
```

Les méthodes

charAt(pos) renvoie le caractère situé à pos dans la chaine. Attention le premier caractère est à la position 0
charCodeAt(pos) renvoie la valeur unicode du caractère situé à pos dans la chaine.
fromCharCode(code) renvoie le caractère correspondant au code unicode envoyé
indexOf(rech,pos) renvoie la position du caractère "rech" à partir de la position "pos" dans une chaine. Si pos n'est pas donnée la recherche commence au premier caractère de la chaine. Si le caractère n'est pas trouvé indexOf retourne -1
Ex :

```
machaine = "toto";
machaine.indexOf("o");// retourne 1
machaine.indexOf("o",2);// retourne 3
```

lastIndexOf(rech,pos) Même chose mais à partir de la fin de la chaine
concat() concaténe deux chaine (comme +)
slice(debut,nombre) extrait "nombre" caractères à partir du caractères positionné à "debut"
Ex

```
adr = "francois.plegades@wanadoo.fr"
pos = adr.indexOf("@");
if (pos != -1)
{
     domaine = adr.slice(pos + 1); //wanadoo.fr
     titulaire = adr.slice(0, pos); //francois.plegades
}
```

substr(debut,longueur) Retourne une partie de la chaîne, avec index de départ et longueur en paramètres.

```
"bonjour".substr(3, 2); // donne jo
```

substring(debut,fin) Retourne une partie de la chaîne, avec index de départ et index de fin.

```
"bonjour".substring(3, 5); // donne jo
```

split("separateur") coupe une chaine sur la base de "separateur" et retourne un tableau. Le séparateur peut être une expression régulière
match(regExp) recherche l'expression régulière "regExp" dans une chaine et retourne le résultat dans un tableau. Si l'expression régulière n'est pas trouvée renvoie null
search(regExp) recherche l'expression régulière "regExp" dans une chaine et retourne la position du premier caractère trouvé. Si l'expression régulière n'est pas trouvée renvoie -1
replace(regExp,remplacement) recherche l'expression régulière "regExp" dans une chaine et la remplace par "remplacement". Si l'expression régulière n'est pas trouvée ne fait rien
Ex :

```
recherche = /[ ./-]/g;
tel = "01-47 12.52/30";
tel_nettoyé = tel.replace(recherche, "")
```

toLowerCase() Met tous les caractères en minuscules.
toUpperCase() Met tous les caractères en majuscules.
toString() transtype en string les nombres ou les booléen

Objets de manipulations

L'objet Function

Il est initialiser via

```
mafonction = new Function("arg","code");
```

ou

```
function mafonction(arg)
{
     code;
}
```

ou

```
mafonction = function(arg)
{
     code ;
}
```

Les propriétés

arguments[] tableau des objets arguments
constructor renvoie le constructeur

L'objet arguments

Cet objet représente sous forme de tableau la valeur des arguments passés à une fonction. On peu ainsi récupérer des arguments non déclaré lors de l'instentialisation de la fonction

Les propriétés

callee renvoie le contenu complet de la fonction.
Cette propriété permet de rappeler la fonction avec de nouveau arguments.(appel récursif)

```
function ecrit_recursif(x)
{
     if (x <= 3)
     {
          document.write(x + "<br />")
          ecrit_recursif.arguments.callee(x + 1);
     }
}
ecrit_recursif(0) ;
```

length nombre d'argument passé

Objets d'opérations :

Les expressions régulières

Le but de cette partie n'est pas de vous faire maitriser les expressions régulières mais de vous rappeler leur principe.
Les expressions régulières permettent d'analyser une chaine en la décrivant ou en décrivant une

partir de celle-ci.
Par exemple, une adresse IP est constitué de quatre groupes de nombres compris entre 0 et 255 et séparé par un point.
Si on doit vérifier la validité d'une adresse IP ou la retrouver dans un texte on pourra définir un expression régulière qui retournera null ou l'adresse suivant qu'elle est trouvé ou pas (match(expReg)).

Testons de 0 à 255
Pour tester un nombre entre 0 et 255 avec une expression régulière il faut découper le nombre en blocs analysable puis décrire les différentes possibilités liées par des ou | .
bloc1 de 0 à 199

```
expReg ([01]?[0-9]?[0-9])
```

soit
1 ou pas de (0 ou 1)
1 ou pas de (0 ou 1 ou 2 ou 3 ou .....ou 9)
1 obligatoire de (0 ou 1 ou 2 ou 3 .....ou 9)
bloc2 de 200 à 249

```
expReg (2[0-4][0-9])
```

soit 2 obligatoire
1 obligatoire de (0 ou 1 ou 2 ou 3 ou 4)
1 obligatoire de (0 ou 1 ou 2 ou 3 .....ou 9)
bloc3 de 250 à 255

```
expReg (25[0-5])
```

soit 2 et 5 obligatoire
1 obligatoire de (0 ou 1 ou 2 ou 3 ou 4 ou 5)

Soit pour tester de 0 à 255

```
([01]?[0-9]?[0-9])|(2[0-4][0-9])|(25[0-5])
```

L'adresse IP est donc
(un jeu de bloc + un point) 3 fois + (un jeu de bloc)

```
expReg = /(([0-1]?[0-9]?[0-9])|(2[0-4][0-9])|(25[0 -5])[.]){3}([0-1]?[0-9]?
[0-9])|(2[0-4][0-9])|(25[0-5])/
```

Avant de tester quelques explications
L'expression est encadré par des slashes // à la place des guillemets

Les commutateurs

On peu après le dernier slash ajouté des commutateurs
i pour ne pas tenir compte de la casse
g pour une recherche globale (retour de toutes les expressions trouvées et pas seulement de la première)
m recherche multiligne (ne s'arrete pas au caractère \n nouvelle ligne)

Les indicateurs de position

^ Recherche à partir du début de texte
$ Recherche jusqu'en fin de texte
/^ expReg $/ le texte est semblable à l'expression régulière

Les quantificateurs

{m,n} de m à n caractères successifs semblables à ...
{m,} m ou + caractères successifs semblables à ...
{m} exactement m caractères successifs semblables à ...
* 0 ou + caractères successifs semblables à ...
+ 1 ou + caractères successifs semblables à ...
? 0 ou 1 caractères successifs semblables à ...

Les classes de caractères

. n'importe quel caractère (pour trouver le point il faut l'échapper \. ou le mettre entre crochets [.])
[...] n'importe quel caractère présent entre les crochets
[^...] n'importe quel caractère a l'exception de ceux présents entre les crochets
Attention : ^ n'a pas le même sens entre les crochet et en dehors de ceux-ci.
[a-zA-Z] n'importe quel lettre
\w tout mot (semblable à [a-zA-Z0-9_])
\W tout sauf un mot (semblable à [^a-zA-Z0-9_])
\s tout espace blanc (semblable à [\r\n\t\v])
\S tout sauf un espace blanc (semblable à [^\r\n\t\v])
\d tout nombre (semblable à [0-9])
\D tout sauf un nombre (semblable à [^0-9])

L'objet RegExp

Il est initialiser via new RegExp() ou //

```
variable = new RegExp("motif", "Commutateur") ;
variable = /motif/Commutateur ;
```

Les propriétés

$' ou rightContext renvoie le texte à droite du dernier motif trouvé
$& ou lastMatch renvoie le dernier motif trouvé
$_ ou input renvoie le texte dans lequel la recherche à été faite
$` ou leftContext renvoie le texte à gauche du dernier motif trouvé
$+ ou lastParen renvoie le dernier bout de texte trouvé par le dernier sous motif (défini par les parenthèse dite capturante)
$n renvoi le sous motif n (n compris entre 1 et 9)
index renvoie la position du premier caractère du motif trouvé
lastIndex renvoie la position à partir de laquelle la prochaine recherche sera faite
source renvoie le motif de l'expression régulière

Les méthodes

compile() remplace le motif par un nouveau motif
exec() renvoie un tableau des résultats de la recherche
test() renvoie true ou false suivant le résultat de la recherche

Exercice
Faire un formulaire qui envoie une adresse IP à une fonction javascript verifie() qui dira si l'adresse est valide ou pas.
Le contenu du champ nommé "ip" du formulaire est récupéré avec document.forms[0].ip.value
Correction

```
<html>
<head>
<title>calcul ttc</title>
<script language="javascript" type="text/javascript">
function verifie()
{
     expReg = /^((([01]?[0-9]?[0-9])|(2[0-4][0-9])|(25[0 -5]))[.]){3}(([0-
1]?[0-9]?[0-9])|(2[0-4][0-9])|(25[0 -5])){1}$/
     leip = document.forms[0].ip.value;
     tab = leip.search(expReg);
     if (tab!=-1)
     {
          alert("Adresse IP valide");
     }
     else
     {
          alert("Adresse IP fausse");
     }
}
</script>
</head>
<body>
<form>
IP : <input type="text" name="ip" /><br />
<input type="button" value="Voir" onclick="verifie()" />
</form>
</body>
</html>
```

Objet Math :

Cet objet n'a pas de constructeur. On l'utilise sous la forme Math.propriété ou Math.fonction()
Math.abs(nombre) -- Valeur absolue
Math.abs(-10) => 10
Math.acos(valeur) -- arc cosinus (radians) -1<=valeur<=1
Math.acos(1) => 0 (angle nul)
Math.asin(valeur) -- arc sinus (radians) -1<=valeur<=1
Math.asin(1) => 1.5707963267949 (soit PI / 2, ou 90°)
Math.atan(valeur)-- arc tangente (radians)
Math.cos(angle) -- cosinus angle en radians

Math.sin(angle) -- Sinus angle en radians
Math.tan(angle) -- Tangente angle en radians
Math.ceil(nombre) -- Arrondit à l'entier supérieur
Math.ceil(2.8) => 3
Math.floor(nombre) -- Arrondit à l'entier inférieur
Math.floor(2.8) => 2
Math.round(nombre) -- Arrondit à l'entier le plus proche Math.round(1.55) => 2
Math.max(nombre1,nombre2) -- Renvoie la plus grande valeur des 2 nombres. Math.max(2,5) => 5
Math.min(nombre1,nombre2) -- Renvoie la plus petite valeur. Math.min(2,5) => 2
Math.random()-- Génère une valeur aléatoire comprise entre 0 et 1.
Pour avoir une valeur aléatoire entre 2 nombres a et b tel que a<b
x=Math.round(Math.random()*(b-a))+a
Math.pow(nombre,puissance) -- Puissance
Math.pow (2,3) => 23 => 8
Math.sqrt(nombre) -- Racine carrée.
Math.sqrt(9) => 3

L'objet Date

Il est initialiser via new Date().
On peut lui passer en argument une date sous forme d'un timestamp Unix.
Par défaut il utilise la date en seconde depuis le 1er janvier 1970 0h 0mn 0s

Les méthodes

Pour récupérer les données de date
getFullYear() renvoie l'année sur quatre chiffres
getMonth() renvoie le numéro du mois (de 0 pour janvier à 11 pour décembre)
getDate() renvoie le jour du mois
getDay() renvoie le jour de la semaine sous forme de nombre (attention 0 = dimanche)
getHours() renvoie l'heure (0 à 23)
getMinutes() renvoie les minutes
getSeconds() renvoie les secondes
getMilliseconds() renvoie les millisecondes

même chose avec getUTC à la place de get qui renvoie la date en temps universel UTC

getTime() renvoie le timestamp en millisecondes (date en milliseconde depuis le 1er janvier 1970)
getTimezoneOffset() renvoie la diférence en minutes entre l'heure GMT et l'heure locale. (sur une machine réglé sur l'heure en France 60 en heures d'hivers et 120 en heures d'été)

Pour modifier une date :
parse() convertie en timestamp en milliseconde une date sous forme de chaine de caractères en anglais
mois jour,année heures:minutes:seconde

```
Date.parse("February 12,2005 18:25:12")
```

setFullYear() modifie l'année sur quatre chiffres
setMonth() modifie le numéro du mois (de 0 pour janvier à 11 pour décembre)
setDate() modifie le jour du mois (argument entre 1 et 31)
setHours() modifie l'heure (0 à 23)
setMinutes() modifie les minutes (0 à 59)
setSeconds() modifie les secondes (0 à 59)
setMilliseconds() modifie les millisecondes

même chose avec setUTC à la place de set qui modifie la date en temps universel UTC

setTime() modifie la date (argument nouvelle date en timestamps en millisecondes)

Pour afficher une date sous forme de chaine de caractères
toDateString() renvoie la date en anglais
toGMTString() renvoie la date et l'heure GMT en anglais
toLocaleDateString() renvoie la date au format défini par la machine sur laquelle le script est exécuté
toLocaleString() renvoie la date et l'heure au format défini par la machine sur laquelle le script est exécuté
toLocaleTimeString() renvoie l'heure au format défini par la machine sur laquelle le script est exécuté
toTimeString() renvoie l'heure au format anglais
toString() renvoie la date et l'heure en anglais
toUTCString renvoie la date et l'heure UTC au format anglais
ValueOf() renvoie la date et l'heure en millisecondes

Java Script

Les objets du navigateur

Les objets fournis par le navigateur

Les navigateurs fournissent à travers leur DOM (document object model) un lot d'objets avec leurs propriétés et leurs méthodes. Cet ensemble d'objets est hiérarchisé et arborescent.

Au sommet on trouve l'objet *window* (la fenêtre) qui contient les objets *document*, *navigator*, *event*, *screen*, *history* et *location*

Chaque objet peut être lié à des événements qui déclencheront des actions précises

C'est à travers les propriétés, les méthodes et les événements qui caractérisent les objets du navigateur que l'on pourra rendre interactif et dynamique les pages html.

A la fin de cette partie vous saurez utiliser les différentes possibilités offertes par les objets du navigateur

La hiérarchie des objets du navigateur

- window
  - document
    - anchors[]
    - classes[]
    - ids[]
    - forms[]
      - elements[]
        - button
        - checkbox
        - fileupload
        - hidden
        - password
        - radio
        - reset
        - select
        - submit
        - text
        - textarea
    - navigator
      - mimeTypes
      - plugins
    - event
    - screen
    - frames[]
    - history
    - location

Les événements

Les événements sont des actions produites soit par l'utilisateur (touche du clavier enfoncée, relâchée, bouton de la souris pressé, pointeur de la souris entrant dans un objet, le survolant, en sortant, etc...) soit par le programme lui même (la fenêtre est redimensionnée, la page est chargée, l'objet s'active ou se déactive, etc...)

Les événements sont détectés par les objets qui les acceptent.
Ils peuvent servir de déclencheur pour des scripts qui leurs sont liés ou faire appel à des fonctions.
L'événement est récupéré par un appel qui commence par on
Cet appel est :
soit une propriété d'une balise html donc de l'objet créé par le DOM,
soit un gestionnaire d'événement lié à l'objet

Ex:
Sous forme de propriété de la balise

```
<html>
<head>
<title>javascript</title>
<script language="javascript" type="text/javascript">
     function clic()
     {
          alert("Bouton cliqué");
     }
</script>
</head>
<body onload = "alert('Page chargée')">
<input type="button" value="clic" onclick="clic()" />
</body>
</html>
```

Sous forme de gestionnaire lié

```
<html>
<head>
<title>javascript</title>
</head>
<body>
<input type="button" id='bouton' value="clic" />
</body>
</html>
<script language="javascript" type="text/javascript">
alert('page chargée');
document.getElementById('bouton').onclick = function()
{
     alert("Bouton cliqué");
}
</script>
```

Principaux événements

Relatif au chargement de la page

onload
onbeforeunload
onunload
onabort

Relatif à l'activation de l'objet (focus)

onbeforeactivate
onactivate
onbeforedeactivate
ondeactivate
onchange
onblur
onbeforeeditfocus
onfocus
onfocusin
onfocusout
oncontrolselect

relatif à l'impression

onbeforeprint
onafterprint

relatif à la mise à jour

onbeforeupdate
onafterupdate

relatif au copié collé

onbeforecopy
oncopy
onbeforecut
oncut
onbeforepaste
onpaste

relatif au drag and drop

ondrag
ondragdrop
ondragstart
ondragenter
ondragover
ondragleave
ondragend

ondrop
onmove
onmovestart
onmoveend

relatif aux erreurs

onerror

relatif à l'utilisation du clavier

onkeydown
onkeyup
onkeypress
onhelp

relatif à l'utilisation de la souris

onclick
ondblclick
onmousedown
onmouseup
onmouseenter
onmousemove
onmouseover
onmouseout
onmouseleave
onmousewheel
oncontextmenu

relatif à l'utilisation des ascenseur

onscroll

relatif aux changement de propriétés

onpropertychange

relatif à l'envoi de formulaire ou à sa réinitialisation

onreset
onsubmit

relatif au redimentionnement

onresize
onresizestart
onresizeend

relatif à l'utilisation de la sélection

onselect
oncellchange
onselectionchange

onselectstart

relatif à l'arrêt de la page par le bouton arrêter du navigateur

onstop

Attention : Les événements dépendent des objets et des navigateurs. Ils peuvent donc fonctionner dans un navigateur et pas dans un autre

L'objet window

L'objet window représente le navigateur dans lequel le document html est affiché. L'objet est créé chaque fois qu'une nouvelle page est affichée. On ne peut donc pas stoquer des informations dans l'objet window sous forme de propriétés.

La fenêtre principale du navigateur est la fenêtre mére. Les fenêtres ouvertes à partir de la fonction open() sont référencées par la valeur retournée par la fonction.
On peut communiquer d'une fenêtre à une autre par leur nom ou par la valeur de la fenêtre ouvrant si elle existe (opener).

Les fonctions définies dans les scripts du document html appartiennent à la fenêtre dans lesquelles elles sont définies. Ce sont donc des méthodes de la fenêtre et on peut les appeler de n'importe quelle fenêtre en utilisant le nom de la fenêtre ou si c'est le parent direct opener

Les objets enfants de window

document	C'est la page html affichée
navigator	C'est le navigateur et ses réglages
event	C'est l'événement généré
screen	C'est l'écran d'affichage et ses réglages
frames[]	C'est les frames du recordset
history	C'est l'historique
location	C'est l'URL

Les propriétés

defaultStatus : contient le texte par défaut qui apparaît dans la barre d'état.
status : contient le texte qui doit apparaître dans la barre d'état. Prend le dessus sur defaultStatus.
screenLeft : contient l'abscisse du coin supérieur gauche.
screenTop : contient l'ordonnée du coin supérieur gauche.
innerHeight contient la dimension verticale en pixels du document (exclu les menus et barres de scrolling de la fenêtre)
innerWidth contient la dimension horizontale en pixels du document (exclu les menus et barres de scrolling de la fenêtre)
outerHeight contient la dimension verticale en pixels du document incluant les menus et barres de scrolling de la fenêtre.

outerWidth contient la dimension horizontale en pixels du document incluant les menus et barres de scrolling de la fenêtre.
length renvoie le nombre de frames contenus dans la fenêtre
name renvoie le nom de la fenêtre
opener C'est l'objet fenêtre qui a ouvert la fenêtre active, donc le parent si il existe
screenLeft Distance en pixels entre le bord gauche de l'écran et le bord droit de la fenêtre (IE)
screenTop Distance en pixels entre le bord haut de l'écran et le bord haut de la fenêtre (IE)
screenX Distance en pixels entre le bord gauche de l'écran et le bord droit de la fenêtre (firefox)
screenY Distance en pixels entre le bord gauche de l'écran et le bord droit de la fenêtre (firefox)

Les méthodes

alert(texte) : Affiche un message à l'utilisateur
confirm(texte) : Pose une question binaire à l'utilisateur
prompt(texte) : Propose une saisie de texte à l'utilisateur
print() : Imprime la page
back() retour arriere dans l'historique
forward() deplacement avant dans l'historique
home() Chargement de la page d'accueil
stop() Arret du chargement en cour
focus() : Donne le focus sur la page
blur() : Retire le focus de la page
open(url,nom,options) : Ouvre un nouvelle fenêtre. On lui passe en arguments l'URL de la page à afficher, le nom de la fenêtre et un jeu d'options

options de fenêtre

fullscreen met en plein écran
height=nb hauteur en pixels
innerHeight=nb hauteur du doc en pixels sans menus
outerHeight=nb hauteur du doc en pixels avec menus
width=nb largeur en pixels
innerWidth=nb largeur du doc en pixels sans menus
outerWidth=nb largeur du doc en pixels avec menus
left=nb position par rapport au bord gauche
top=nb position par rapport au haut de l'écran
directories=yes/no affiche ou pas
location=yes/no affiche ou pas la barre d'adresse
menubar=yes/no affiche ou pas la barre menus
scrollbars=yes/no affiche ou pas les ascenseurs
toolbar=yes/no affiche ou pas la barre outils
status=yes/no affiche ou pas la barre status
resizable=yes/no permet ou pas le redimentionnement

close() : Ferme une fenêtre

moveBy(x,y) : Déplacement relatif
moveTo(x,y) : Déplacement vers un point précis
resizeBy(x,y) : Redimensionnement relatif

resizeTo(x,y) : Redimensionnement à une taille fixe
scrollBy(x,y) : Scrolle la fenêtre en relatif
scrollTo(x,y) : Scrolle à une position définie

setTimeout(action,delai) : Déclenche une action apres "delai" milliseconde et retourne un timer
clearTimeout(timer) : Supprime le timer retourné par setTimeout
setIntervale(action,delai) Déclenche une action répétitive tout les "delai" et retourene un timer
clearInterval(timer) Supprime le timer retourné par setInterval

Ex :

créer une page ouvrant un popup au bout de 10 secondes.
Le popup se referme automatiquement après 5 secondes.
A la fermeture le popup lance une fonction dans la page principale

Corrigé:

Fenêtre principale

```
<html>
<head>
<title>objet window</title>
<script language="javascript" type="text/javascript">
function decompte(val)
{
     if (val == 0)
     {
          lex = 250;
          ley = 150;
          fenetre = window.open("enfant.html", "popup", "location=no,
menubar=no, resizable=no, scrollbar=no, status=no, toolbar=no, top=" + ley
+ ", left=" + lex + ", width=300, height=200");
     }
     else
     {
          doc = document.getElementById("leaff");
          doc.innerHTML = "<h1>" + val + "</h1>";
          val--;
          chrono = setTimeout("decompte(" + val + ")",1000)
     }
}

function enfant_m_appelle(quoi)
{
     doc = document.getElementById("leaff");
     doc.innerHTML = "<h1>" + quoi + "</h1>";
}

function appel()
{
     fenetre.appel();
}
```

```
</script>
</head>
<body >
Lancement de la fenêtre popup
<input type="button" value="go" onclick="decompte(10)" />
<div name="leaff" id="leaff" ></div>
<br /><br />
<input type="button" value="Arrêter le lancement"
onclick="clearTimeout(chrono)" />
<br /><br />
<input type="button" value="appel à popup" onclick="appel()" /><br />
</body>
</html>
```

Fenêtre popup

```
<html>
<head>
<script language="javascript" type="text/javascript">
val = 5;
function decompte()
{
     if (val == 0)
     {
          lex = 250;
          ley = 150;
          this.close()
     }
     else
     {
          doc = document.getElementById("aff");
          doc.innerHTML = "<h1>" + val + " secondes</h1>";
          val--;
     }
}

function go()
{
     doc = document.getElementById("aff");
     doc.innerHTML = "<h1>" + val + " secondes</h1>";
     val--;
     dec = setInterval('decompte()',1000);
}

function ferme()
{
     this.opener.enfant_m_appelle("l'enfant est fermé");
}

function appel()
{
     clearInterval(dec);
     this.focus();
     doc = document.getElementById("aff");
     doc.innerHTML = "<h1>On m'appelle!</h1>";
```

```
}
</script>
</head >
<body onload="go()" onunload="ferme()">
Je suis la fenêtre enfant. Je me ferme dans
<div name="aff" id="aff" ></div>
<br /><br />
<input type="button" value="Arrêter la fermeture"
onclick="clearInterval(dec)" />
</body>
</html>
```

Les objets history, screen et location

history

history.length renvoie le nombre d'entrée dans l'historique
history.back() déplacement arriére dans l'historique
history.forward() déplacement avant dans l'historique
history.go(nb) déplacement avant ou arriére dans l'historique de "nb" pages

screen

C'est l'écran de l'ordinateur client
availHeight hauteur de l'écran
availWidth largeur de l'écran
height hauteur totale de l'ecran
width largeur totale de l'ecran
colorDepth ou **pixelDepth** nombre de bits par pixel pour l'affichage de la couleur

location

Cet objet contient les informations relative à l'URL
rappel de la structure d'une URL

```
protocole://serveur:port/chemin/document?variables#ancre
```

hash contient la partie de l'URL derrière # (ancres)
host contient le nom du serveur et le port
hostname contient le nom du serveur
href contient l'URL complète
pathname contient le chemin d'accès au document
port contient le port d'entrée (80 par defaut)
protocol contient le protocol utilisé (en général http)
search contient la partie de l'URL derrière ? (passage des variables en méthode get)
assign(URL) modifie l'URL du document (charge la nouvelle page)
reload() recharge la page en cours
replace(URL) remplace l'URL en cours par une nouvelle URL (l'historique ne crée pas de nouvelle entrée)

L'objet navigator

contient 2 objets enfants sous forme de tableau d'objet
mimeTypes[]
plugins[]
C'est l'objet qui vous renvoie les renseignements sur le navigateur utilisé par le client

Les propriétés :

appCodeName nom du code du navigateur (Mozilla pour IE et Firefox)
appName nom du navigateur (Microsoft Internet Explorer pour IE et Netscape pour Firefox)
userAgent renvoie la totalité des renseignements sur le navigateur (données userAgent du protocole http)
platform renvoie le type de plateforme (Wn32)
appVersion renvoie le numéro de version du navigateur et le système d'exploitation
cookieEnabled renvoie true ou false suivant que les cookies sont acceptés ou pas

Pour IE

systemLanguage langue du navigateur
userLanguage langue du navigateur

Pour Firefox

language langue du navigateur

mimeTypes

renvoie les informations sur les types de fichiers reconnus par le navigateur. Internet explorer utilisant sous windows les actives X ignore l'objet mimeTypes.

Propriétés

length renvoie le nombre de type mime reconnu
name renvoie le nom du type mime (undefined la plupart du temps)
type renvoie le type du type mime
suffixe renvoie les extentions du type mime
description renvoie la description du type mime
enabledPlugin renvoie [object Plugin] si le plugin qui sert à la lecture du type mime est actif.

plugins

L'objet plugins ne fonctionne que pour les navigateurs utilisant les plugins. Internet explorer utilise sous windows les activeX donc plugins ne générera pas d'erreurs mais sera inefficace

Propriétés

length renvoie le nombre de plugins
name renvoie le nom du plugin

filename renvoie le nom l'emplacement du fichier code du plugin
description renvoie la description du plugins

Ex de code renvoyant les infos sur les plugins du navigateur (pas pour IE qui utilise activeX)

```
<html>
<head>
<title>objet window</title>
<script language="javascript" type="text/javascript">
function init()
{
     var txt = "";
     /* pour IE this.navigator.plugins.length renvoie 0
     donc txt restera vide */
     for (i = 0; i < this.navigator.plugins.length; i++)
     {
          txt += "<br />" + this.navigator.plugins[i].name +
          " ( " + this.navigator.plugins[i].filename + " ) : " +
          this.navigator.plugins[i].description ;
     }
     ecrit(txt);
}

function ecrit(txt)
{
     doc=document.getElementById("letexte")
     doc.innerHTML=txt
}
</script>
</head>
<body onload = "init();">
<div id="letexte" name="letexte"></div>
</body>
```

L'objet event

L'objet Event est l'objet créé par javascript chaque fois qu'un événement est récupéré.
Il permet donc d'accéder aux propriétés de l'événement et à son déroulement.

Propriétés

button renvoie le bouton de la souris pressé (pour IE : 1 gauche, 2 droit, 4 milieu et pour firefox : 0 gauche, 1 milieu et 2 droit)

clientX position horizontale de l'événement
clientY position verticale de l'événement
screenX position verticale de l'événement dans l'écran
screenY position horizontale de l'événement dans l'écran

type renvoie le type d'événement
cancelBubble indique si l'événement peut se propager aux objets parents. Par defaut est à false et se propage. Il faut le basculer à true pour empécher sa transmission.

altKey renvoie true si la touche alt est enfoncée
ctrlKey renvoie true si la touche Ctrl est enfoncée
shiftKey renvoie true si la touche maj est enfoncée

Pour IE

keyCode renvoie le code unicode de la touche enfoncée

Pour Firefox

keyCode renvoie le code unicode de la touche enfoncée avec keypress si le caractère n'est pas affichable (touche f1,..f12 par ex) ou avec keydown
charCode renvoie le code unicode de la touche enfoncée avec keypress si le caractère est affichable

Pour IE

srcElement objet sur lequel est intervenu l'événement

Pour Firefox

target objet sur lequel est intervenu l'événement

Technique de récupération de l'objet événement

cas 1 : transmission via une propriété de balise html

on le transmet dans les arguments d'appel de la fonction en utilisant la variable event (objet event)

```
<html>
<head>
<title>javascript</title>
<script language="javascript" type="text/javascript">
function ecrit(txt)
{
     doc = document.getElementById("letexte") ;
     doc.innerHTML = txt ;
}

function clic(quoi,evt)
{
     var txt = quoi + "<br />Position du clic donnée par body : " +
evt.clientX + ' - ' + evt.clientY;
     ecrit(txt);
}
</script>
</head>
<body onclick = "clic('toto',event)">
<div id="letexte" name="letexte" style="background-color:#FFCCCC;"></div>
</body>
</html>
```

cas 2 : gestionnaire d'événements

On le récupère dans la fonction appelé.

Chez IE il est mis à disposition dans une variable event

Chez Firefox il est transmis dans les arguments de l'appel en première position

```
<html>
<head>
<title>objet window</title>
<script language="javascript" type="text/javascript">
function monEvenement()
{
     if (arguments[0])  //cas firefox
     {
     evt = arguments[0];
     }
     else               // cas IE
     {
     evt = event;
     }
     var txt = "Position du clic :" + evt.clientX + " - " + evt.clientY;
     for (val in evt)
     {
      txt += "<br />" + val + " : " + eval("evt." + val) ;
     }
     ecrit(txt);
}
function ecrit(txt)
```

```
{
     doc = document.getElementById("letexte") ;
     doc.innerHTML = txt ;
}

function init()
{
document.getElementById('letexte').onclick = monEvenement;
}
</script>
</head>
<body onload="init()">
<div id="letexte" name="letexte">clic</div>

</body>
<script language="javascript" type="text/javascript">
//pour initialiser l'événement dans IE
//document.link[0].onclick = evenement;
</script>
</html>
```

l'objet document

C'est la page html dans sa totalité (de <html> à </html>).
Via le DOM, les éléments de la page définis dans les balises html sont considérés comme des objets enfant de document.

Ils sont accessibles via document.getElementById("id_de_objet") si on a donné un id à la balise html. L'id doit être unique.
Ils sont accessibles via document.getElementsByName("nom") ou document.getElementsByTagName("nom_balise") qui renvoi un tableau des objets portant le nom "nom" ou un tableau des objets dont le nom de balise est "nom_balise".

Chaque objet de la page donne accès en lecture ou en écriture à ses propriétés, autant celle de base que celle qui ont été rajoutés.
Les propriétés de base de ces objets sont les arguments des balises HTML

La propriétés **innerHTML** des objets du document permet de lire ou d'écrire à la volée dans les balises de la page du code HTML

Principaux objets enfants de document

anchors[] links[] tableau des ancres et liens de la page
forms[] tableau des formulaires de la page
styleSheets[] tableau des feuilles de style de la page
images[] tableau des images de la page

Les propriétés

LinkColor, alinkColor, vlinkColor, bgColor, fgColor

title permet de lire ou modifier la balise <title> de la partie <head>
body permet de travailler sur le contenu de la partie body du document sans toucher à la partie head
Ex

```
<html>
<head>
<title>objet window</title>
<script language="javascript" type="text/javascript">
memoire = "";

function init()
{
     var txt = "";
     memoire = document.formu.texte.value;
     txt = memoire.replace(/\n/g,"<br />") ;
```

```
     txt += "<br /><input type=\"button\" value=\"Voir\"
onclick=\"retour()\" />" ;
     document.body.innerHTML = txt;
}

function retour()
{
     var txt = "<form name=\"formu\" id=\"formu\" >" ;
     txt += "<textarea name=\"texte\" cols=\"50\" rows=\"10\" >" + memoire
+ "</textarea>" ;
     txt += "<br /><input type=\"button\" value=\"Voir\" onclick
= \"init()\" />" ;
     txt += "</form>" ;
     document.body.innerHTML = txt;
}
</script>
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<form name="formu" id="formu" >
<textarea name="texte" cols=50 rows=10 >Entrer du texte</textarea>
<br /><input type="button" value="Voir" onclick="init()" />
</form>
</body>
</html>
```

cookie si navigator.cookieEnabled vaut true on peut mémoriser des valeurs ou des variables dans un cookie enregistré chez le client.
Cette technique permet de mettre à disposition de différentes pages ou du serveur des variables.
Attention : un cookie ne sait que mémoriser des chaines de caractères. Si on veut mettre en mémoire des tableau ou des objets il faudra les transformer en chaine de caractère (sérialisation) et les remettre sous forme de tableau ou d'objet à la récupération (dé-sérialisation)

Structure d'un cookie (seul nom=valeur est obligatoire)

```
nom=valeur;expire=date;domain=domaine;path=chemin;secure
```

- *nom* est le nom de la variable cookie et *valeur* son contenu
- *expire* contient la date jusqu'à quand le cookie est valable. Passé cette date le navigateur peut l'effacer
- *domain* contient le domaine à partir duquel le cookie sera récupérable. En dehors de ce domaine il ne sera pas possible de récupérer le cookie
- *path* contient le chemin à l'intérieur du domaine à partir duquel le cookie est récupérable.
- *secure* true pour exiger que le cookie voyage avec une connexion sécurisée

Voyons le fonctionnement des cookies à travers deux fonctions ecrit_cookie() et lit_cookie()

```
<html>
<head>
<title>objet window</title>
<script language="javascript" type="text/javascript">
if (navigator.cookieEnabled==false){
```

```
var txt="Vous avez interdit les cookies sur votre machine.\n Ce programme
risque d'\u00eatre gravement perturb\u00e9 par cela.\nD\u00e9sol\u00e9"
alert(txt)
}
function ecrit_cookie(txt){
var letexte=escape(txt);
var date=new Date()
var ladate=new Date()
ladate.setTime(date.getTime()+3600);
document.cookie="texte="+letexte+";expires="+ladate.toGMTString()
+"domain=localhost;path= /"
document.cookie="toto="+letexte+";expires="+ladate.toGMTString()
+"domain=localhost;path= /"
}
function lit_cookie(quoi){
var lecookie=document.cookie
tab_cookie=lecookie.split(";")
for (i=0;i<tab_cookie.length;i++){
tmp=tab_cookie[i].split("=");
if (tmp[0]==quoi){return(tmp[1])}
}
return false
}
function init(){
var txt="";
ecrit_cookie(document.formu.texte.value)
txt=document.formu.texte.value.replace(/\n/g,"<br />")
document.body.innerHTML=txt+"<br /><input type=\"button\" value=\"Retour\"
onclick=\"retour()\" />";
}
function retour(){
memoire=lit_cookie("texte");
var txt="<form name=\"formu\" id=\"formu\" >"
txt+="<textarea name=\"texte\" cols=50 rows=10 >"+unescape(memoire)
+"</textarea>"
txt+="<br /><input type=\"button\" value=\"Voir\" onclick=\"init()\" />"
txt+="</form>";
document.body.innerHTML=txt;
}
</script>
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<form name="formu" id="formu" >
<textarea name="texte" cols=50 rows=10 >Entrer du texte</textarea>
<br /><input type="button" value="Voir" onclick="init()" />
</form>
</body>
</html>
```

referrer renvoie l'URL de la page appelant la page en cour (semblable à historique -1)

Les méthodes

getElementById("id") renvoie l'élément portant l'id "id"
getElementsByName("name") renvoie un tableau des éléments portant le nom "name"
getElementsByTagName("balise") renvoie un tableau des éléments dont le nom de balise est "balise"
createStyleSheet("URL",index) charge la feuille de style dont l'URL est "URL" dans la page. Cette feuille de style est positionnée dans le tableau de feuilles de style à l'index "index" ou en fin de tableau si "index" n'est pas spécifié
elementFromPoint(x,y) renvoie l'élément en position x,y
getSelection() renvoie le texte selectionné par l'utilisateur. Sur Internet Explorer sous windows on utilise document.selection.createRange() et sur safari window.getSelection()
write("texte") envoie texte à l'affichage

Il est aussi possible de travailler sur un document html en utilisant le DOM XML.

Exercice
Faire un champs de saisie (textarea) qui met à jour le texte d'un calque au fur et à mesure de la saisie.

```
<html>
<head>
<title>objet window</title>
<link href="style.css" rel="stylesheet" type="text/css" />
<script language="JavaScript" type="text/JavaScript">
<!--
function met_en(quoi)
{
     var deja = false ;
     if (window.getSelections)
     {
      txt = window.getSelection();
     }
     else if (document.getSelection)
     {
      txt = document.getSelection() ;
     }
     else if (document.selection)
     {
      txt = document.selection.createRange().text;
     }
     deja = nettoie_bal(txt,quoi) ;
     if (deja)
     {
      txt_balise = txt
      switch(quoi)
      {
            case "g":
                  txt = "<b>" + txt + "</b>" ;
                  break;
            case "i":
                  txt = "<i>" + txt + "</i>";
                  break;
```

```
                case "s":
                      txt = "<u>" + txt + "</u>";
                      break;
          }
         }
         else
         {
          switch(quoi)
          {
                case "g":
                      txt_balise = "<b>" + txt + "</b>";
                      break;
                case "i":
                      txt_balise = "<i>" + txt + "</i>";
                      break;
                case "s":
                      txt_balise = "<u>" + txt + "</u>";
                      break;
          }
         }

         document.formu.texte.value =
document.formu.texte.value.replace(txt,txt_balise);
         init(document.formu.texte.value)
}

function nettoie_bal(txt,quoi)
{
         var saisie = document.formu.texte.value;
         var deja = false
         pos = saisie.indexOf(txt);
         if ((pos-2) != 0)
         {
          bal = saisie.substr((pos - 3), 3);
          switch(bal)
          {
                case "<b>":
                      if (quoi == "g"){deja = true;}
                      return deja
                case "<i>":
                      if (quoi == "i"){deja = true;}
                      return deja
                case "<u>":
                      if (quoi == "s"){deja = true;}
                      return deja
          }
         }
}

function init(txt)
{
         document.getElementById("voirtxt").innerHTML = txt.replace( /\n/g,"<br
/>")
}
//-->
```

```
</script>
</head>
<body>
<form name="formu" id="formu" >
<textarea name="texte" cols=50 rows=10 onkeyup="init(this.value)" >Entrer
du texte</textarea>
<br />
<input type="button" value="Voir"
onclick="init(document.formu.texte.value)" />
</form>
<div id="voirtxt" style="position:absolute; left:460px; top:19px;
width:384px; height:179px; z-index:1; background-color: #FFFFFF; layer-
background-color: #FFFFFF; border: 1px none #000000;"></div>
<div id="boutons" style="position:absolute; left:460px; top:202px;
width:384px; height:18px; z-index:2" align="center">
<input name="g" id="g" type="button" value="Gras"
onclick="met_en(this.name)" />
<input name="i" type="button" id="i" value="Italique"
onclick="met_en(this.name)" />
<input name="s" type="button" id="s" value="Souligne"
onclick="met_en(this.name)" />
</div>
</body>
</html>
```

Manipulation des styles dans le document

Dans une page HTML le style peut être soit dans une feuille de style externe ou interne, soit en tant qu'argument style de balise.

Dans le premier cas on peut accéder à la feuille via la propriété styleSheets qui retourne un tableau d'objets feuille de style indexant les feuilles dans l'ordre de leur chargement dans la page

```
<html>
<head>
<title>page principale</title>
<link href="css/mafeuille.css" rel="stylesheet" type="text/css">
<style type="text/css">
<!--
#test {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 20px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}

#toto {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 10px;
     color: #006699;
```

```
        height: 150px;
        width: 200px;
        overflow: auto;
        position: absolute;
        left: 150px;
        top: 201px;
    }
    -->
    </style>
    </head>
    <body>
     …....
    </body>
    </html>
```

Dans cet exemple on a
document.styleSheets.length qui renvoie 2

document. StyleSheets[0].href renvoie css/mafeuille.css

Objet styleSheet

Les propriétés

href : adresse de la feuille de style externe. Permet de charger ou changer une feuille de style externe

disabled : true ou false pour activer ou désactiver la feuille de style

rules (IE) ou cssRules (Firefox) : Tableau des règles de la feuille de style indexé dans l'ordre de l'écriture

Les méthodes

IE

addRule("règle css") : insère une nouvelle règle de style en fin de tableau rules

deleteRule(index) : supprime la règle de style du tableau rules à l'index 'index'

Firefox

insertRule("règle css",index) : insère une nouvelle règle de style dans le tableau cssRules à l'index 'index'

removeRule(index) : supprime la règle de style du tableau cssRules à l'index 'index'

Ex :

Changer dans la classe test la couleur (ff0000) et la taille (10 pixels) de la police

```
<html>
<head>
```

```
<title>page principale</title>
<script language="javascript">
<!--
function change_prop(quoi)
 {
     try {
      //IE
      tab_regles = document.styleSheets[1].rules[0] ;
     }catch(e)
     {
      //les autres
      tab_regles = document.styleSheets[1].cssRules[0] ;
     }

     tab_regles.style.color = '#ff0000';
     tab_regles.style.fontSize = '10px';
 }
//-->
</script>

<link href="css/mafeuille.css" rel="stylesheet" type="text/css">

<style type="text/css">
<!--
#test {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 20px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}

#toto {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 10px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}
-->
</style>
</head>
<body>
    <div id="test" onclick="change_prop(this)">
      <p>ceci est un test de </p>
      <p>JavaScript</p>
    </div>
```

```
    <h1> </h1>
</body>
</html>
```

Manipulation des classes de style

Si les styles sont gérés via des classes de style (.maClasseDeStyle) on peut obtenir des transformation du style des éléments de la page HTML par changement de classe.

className est la propriété d'attribution de classe de style pour un élémént (attribut class de la balise)

Ex On attribut une classe test à un div puis on la change pour une classe toto

```
<script language="javascript">
<!--
function change_prop(quoi)
 {
     quoi.className = 'toto'
 }
//-->
</script>

<link href="css/mafeuille.css" rel="stylesheet" type="text/css">

<style type="text/css">
<!--
.test {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 20px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}

.toto {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 10px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}
-->
</style>
</head>
<body>
    <div id="test" onclick="change_prop(this)" class="test">
      <p>ceci est un test de </p>
```

```
        <p>JavaScript</p>
      </div>
      <h1> </h1>
  </body>
                                        </html>
```

Manipulation de l'attribut style

Si le style est géré par l'attribut style d'une balise il devient objet style dans le DOM.

On peut donc le manipuler directement en utilisant document.getElementById('id').style

Les règles de style sont elles même enfant objet de style. Elles s'écrive de la même manière que dans une feuille de style avec le remplacement du tiret – par la mise en majuscule de la première lettre du texte suivant le tiret.

La règle de style *background-color* devient l'objet *backgroundColor*

On peut donc changer la couleur de fond d'une balise

```
<div id='aff' style='background-color:#FFFFCC;'>Bla bla bla</div>
```

avec le javascript

```
document.getElementById('aff').style.backgroundColor = '#CCFFFF' ;
```

Attention : Les dimension de style oblige à utiliser px pour stipuler que ces dimension sont en pixels. Si on veut augmenter cette dimension de x pixel il faut la convertir en entier avec parseInt(), ajouter x et ajouter px en fin

```
x = 1 ; //valeur d'incrémentation ;
posX = parseInt(document.getElementById('id).style.left);
posX += x;
document.getElementById('id).style.left = posX + 'px')
```

Attention : la propriété style est issu de l'attribut style. Pour pouvoir récupérer une règle pour la modifier il faut l'avoir implémenté dans l'attribut style

Ex :

```
<html>
<head>
<title>page principale</title>
<script language="javascript">
<!--
function change_prop(quoi)
 {
     var posX = parseInt(quoi.style.left);
     posX += 100;
     quoi.style.left = posX + 'px' ;
 }
//-->
</script>

<link href="css/mafeuille.css" rel="stylesheet" type="text/css">
```

```
<style type="text/css">
<!--
#test {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 20px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}

#toto {
     font-family: Arial, Helvetica, sans-serif;
     font-size: 10px;
     color: #006699;
     height: 150px;
     width: 200px;
     overflow: auto;
     position: absolute;
     left: 150px;
     top: 201px;
}
-->
</style>
</head>
<body>
    <div id="test" onclick="change_prop(this)"
style="position:absolute;left:100px;top=100px;">
      <p>ceci est un test de </p>
      <p>JavaScript</p>
    </div>
    <h1> </h1>
</body>
</html>
```

Application : le drag and drop

Si on découpe une séquence de drag and drop on a:
Le drag and drop commence lors de l'événement mousedown
Le déplacement de l'objet suit la position de la souris lors de l'événement mousemove
Le drag and drop se termine lors de l'événement mouseup
Pour gérer les événements on place des gestionnaires d'événements au niveau du document (page HTML) et on identifie l'objet à déplacer via son id.
On récupère la position du clic et la position de l'objet au départ du drag and drop (mousedown)
On met une variable isdrag à true. Elle restera à true jusqu'à la fin (mouseup)
A chaque déplacement de la souris on récupère la position de la souris et on recalcule la position de l'objet *posObjet = posObjetDepart + (posSouris – posSourisDepart)*
On repositionne l'objet via son style.left et style.top

```
<html>
<head>
<title>drag and drop</title>
<script language="javascript" type="text/javascript">
<!--
var isdrag = false;
var x, y;
var dobj;

function movemouse(ev)
{
  if (isdrag)
  {
  var lex, ley
     //pour ie qui transmet pas l'evenement
    if (!ev)
     {
          ev = event;
     }

     lex = tx + ev.clientX - x ;
     ley = ty + ev.clientY - y ;

     if (lex>0){
          dobj.style.left = lex
     }
     if (ley>0){
          dobj.style.top  = ley
     }
    return false;
  }
}

function selectmouse(ev)
{
     if (ev)
     {
          var fobj = ev.target;
     }
     else
     {
          var fobj = event.srcElement;
          ev = event;
     }

     x = ev.clientX;
     y = ev.clientY;

     if (fobj.id =="palette")
     {
          isdrag = true;
          dobj = fobj;
          tx = parseInt(dobj.style.left);
          ty = parseInt(dobj.style.top);
          document.onmousemove = movemouse;
```

```
            return false
        }
}

document.onmousedown = selectmouse;
document.onmouseup = function(){isdrag = false;};
//-->
</script>
</head>
<body bgcolor="#FFFFCC">

<div id="palette" style="position:absolute;
     left:0px;
     top:0px;
     width:100px;
     height:300px;
     z-index:100;
     background-color: #CCCCFF;
     padding:10px;
     border:solid 1px black;
     text-align:center;" >
<span style="font-weight:bold;">Changer la couleur de fond</span>
<br /><br />
<span onClick="document.getElementById('aff').style.backgroundColor =
'red';" style="cursor:pointer">Rouge</span><br />
<span onClick="document.getElementById('aff').style.backgroundColor =
'green';" style="cursor:pointer">Vert</span><br />
<span onClick="document.getElementById('aff').style.backgroundColor =
'lightblue';" style="cursor:pointer">bleu</span><br />
<span onClick="document.getElementById('aff').style.backgroundColor =
'yellow';" style="cursor:pointer">Jaune</span><br />
<span onClick="document.getElementById('aff').style.backgroundColor =
'#FFCCFF';" style="cursor:pointer">Magenta</span><br />
<span onClick="document.getElementById('aff').style.backgroundColor =
'#CCFFFF';" style="cursor:pointer">Cyan</span><br />
</div>
<div id="aff" style="position:absolute;
     left:150px;
     top:100px;
     width:500px;
     height:350px;
     z-index:1;
     background-color: #CC0000;" >
</div>

</body>
</html>
```

L'objet images

On peut lire ou changer les propriétés d'une image via le tableau des images du document document.images[index] ou l'index est le numéro d'ordre du classement des images dans la page ou plus simplement en nommant l'image (name="monimage") puis par document.monimage.
On peut aussi accéder aux propriétés via document.getElementById('monid')
Toutes les propriétés de l'image sont lisible et transformable particulièrement sa src qui permet de changer l'image à la volée
Ex :

```
<html>
<head>
<title>images</title>
<script language="JavaScript" type="text/JavaScript">
<!--
function change(quoi,comment){
var la_url = new Array("images/img1.jpg", "images/img2.jpg") ;
quoi.src = la_url[comment] ;
}
//-->
</script>
</head>
<body>
<div id="voirtxt">
<img src="images/img1.jpg" name="image1" onmouseover="change(this,1)"
onmouseout="change(this,0)"/>
</div>
</body>
</html>
```

Il est aussi possible de créer un objet image avec new Image() et de le rajouter avec appendChild(Obj_Image) dans le DOM et ainsi créer une image dans la page

```
<html>
<head>
<title>images</title>
<script language="javascript" type="text/javascript">
function aff_img()
{
     img = new Image() ;
     img.src = "images/ciceron.jpg" ;
     img.width = 188 ;
     img.height = 237 ;
     document.getElementById('aff').appendChild(img)  ;
}
</script>
</head>
<body >
<div id="debug" style="cursor:pointer;" onclick="aff_img()">voir
image</div>
<div id="aff" ></div>
</body>
```

```
</html>
```

l'objet forms

C'est un tableau de tous les objets formulaire de la page.
On peut les récupérer soit dans le tableau via son index de chargement dans la page, soit directement en le nommant via son nom ou par son id

Soit un formulaire

```
<form name="formu" action="toto.com" method="post" enctype="multipart/form-data" target="_SELF" id="formu">
<!-- éléments du formulaire -->
</form>
```

Chaque élément du formulaire est lisible et transformable soit par document.form[0] soit par le nom document.formu soit via document.getElementById('formu')

les propriétés :

action lit ou change l'URL appelée par le formulaire
enctype lit ou change l'encodage du formulaire
method lit ou change la methode d'envoi
target lit ou change le mode d'affichage du retour.

Les méthodes :

reset() remet sous forme initiale le formulaire (comme le bouton reset)
submit() envoie le formulaire (comme le bouton submit)

Ex:

```
<html>
<head>
<title>formulaire</title>
<link href="style.css" rel="stylesheet" type="text/css" />
<script language="JavaScript" type="text/JavaScript">
<!--
function prop()
{
     var txt = "";
     with (document.formu)
     {
      txt += "Action :" + action + "<br />";
      txt += "Methode :" + method + "<br />";
      txt += "Codage :" + enctype + "<br />";
      txt += "Cible :" + target + "<br />";
      txt += "Nom :" + name + "<br />";
      txt += "Nombre d'éléments :" + length + "<br />";
     }
     document.getElementById("voirtxt").innerHTML = txt + "<br /><input type=\"button\" value=\"Envoyer\" onclick=\"envoie()\" />"
}
```

```
function change(){
     with (document.formu)
     {
      action = "#";
      method = "get";
      enctype = "";
      target = "_BLANK";
     }
}

function envoie()
{
     document.formu.submit()
}
//-->
</script>
</head>
<body>
<div id="voirtxt" name="voirtxt" style="position:absolute; left:0px;
top:100px; width:400px; height:200px;">
propriétés du formulaire
</div>
<form name="formu" action="toto.com" method="post" enctype="multipart/form-
data" target="_SELF">
<!-- éléments du formulaire -->
<input type="text" name="texte" value="toto" />
<br /><input type="button" name="propriété" value="propriétés"
onclick="prop()" />
<br /><input type="button" name="ch" value="change_prop" onclick="change()"
/>
</form>
</body>
</html>
```

Les objets de formulaire

Tous les objets d'un formulaire sont accessible via le tableau elements[] de forms en donnant son index ou son nom document.forms[0].elements[0] ou directement via les noms donnés aux éléments document.formu.texte

On trouve 3 grands type de champs dans un formulaire : input, select et textarea

Les objets input

<input /> (button, submit, reset, checkbox, radio, fileupload, text, password, hidden)
Les propriétés des éléments <input /> sont récupérables et transformables

Propriétés

align alignement du champ (left, center, ou right)
checked coche du champ checkbox ou radio (true ou false)

defaultChecked coche par défaut du champ checkbox ou radio (true ou false)
defaultValue valeur par défaut du champ
value valeur du champ. Permet de récupérer les valeurs entrées ou cochées dans le champ
disabled desactive le champ (true ou false)
readOnly desactive le champ de saisie (true ou false)
size ou **width** largeur en pixels du champ
status renvoie l'état des champs à cocher (true ou false)
name nom du champ
type type du champ (attribut type de la balise input)
form formulaire père du champ
maxLength nombre maximum de caractères saisissable dans le champ

méthodes

blur() rend le champ inactif (enlève le focus)
focus() rend le champ actif (donne le focus)
click() envoie un clic au bouton
select() selectionne le contenu du champ

les objets textarea

L'objet textarea permet de lire ou changer les propriétés de la balise <textarea>

Propriétés

defaultValue valeur par défaut du champ
value valeur du champ. Permet de récupérer les valeurs entrées ou cochées dans le champ
disabled desactive le champ (true ou false)
readOnly desactive le champ de saisie (true ou false)
cols nombre de colonne du champ (attribut cols)
rows nombre de lignes du champ (attribut rows)
wrap donne le réglage des retours à la ligne
size largeur en pixels du champ
name nom du champ
form formulaire père du champ
maxLength nombre maximum de caractères saisissable dans le champ

méthodes

On peut via les méthodes de l'objet activer, désactiver selectionner ou cliquer le champ
blur() rend le champ inactif (enlève le focus)
focus() rend le champ actif (donne le focus)
click() envoie un clic au bouton
select() selectionne le contenu du champ

les objets select

L'objet select permet de lire ou changer les propriétés de la balise <select>.

L'objet select possède des objets option disponible via le tableau options[]

Propriétés

disabled desactive le champ (true ou false)
form formulaire père du champ
length Nombre de balise option de la liste
multiple indique si la liste est à choix multiple ou unique (true ou false)
name nom du champ
selectedIndex donne l'index du choix dans la liste
size largeur en pixels du champ
value valeur du champ. Permet de récupérer les valeurs entrées ou cochées dans le champ

Méthodes

blur() rend le champ inactif (enlève le focus)
focus() rend le champ actif (donne le focus)
add(element,avant) , option.add(element,avant) insère l'élément "element" avant l'élément "avant" dans la liste des options. Si "avant" est absent met l'élément "element" en fin de liste
remove(rang) , option.remove(rang) supprime l'élément indéxé dans la liste par "rang"

L'objet option

Il permet de gérer les options d'une liste.
constructeur new Option("etiquette","valeur")
Le constructeur permet de rajouter des options à la volée dans une liste déroulante <select>

```
document.formulaire.liste_deroulante.options[index_nouveau] = new
Option("etiquette","valeur")
```

Propriétés

defautSelected true si l'option est la selection par défaut
selected true si l'option est selectionnée
disabled true si l'option est désactivée
index index de l'option dans le tableau des options
text texte situé dans la balise (étiquette)
value attribut value de la balise
form formulaire d'appartenance

Les objets Java Script

mis à disposition

XML

AJAX

L'objet XML

Les navigateurs actuels mettent à disposition des objets basé sur le DOM XML.
Si votre document est compatible XML (XHTML) vous pouvez naviguer dans la page html via le DOM XML.
C'est une solution qui peut être très pratique pour transformer dynamiquement les pages HTML
Pour utiliser l'objet XML on crée l'objet puis on charge le document XML dans l'objet créé.

Manipulation du noyau du document XML

Création d'un objet XML

Sous IE

```
var xmlDoc = new
ActiveXObject("Microsoft.XMLDOM") ;
xmlDoc.async = "false" ;
```

Pour charger un fichier

```
xmlDoc.load('Document.xml');
xmlObj = xmlDoc.documentElement;
```

Pour charger une variable contenant du XML sous forme texte

```
xmlDoc.loadXML(MaVarXML);
xmlObj = xmlDoc.documentElement;
```

Sous Mozilla

```
var xmlDoc =                                          ↓
document.implementation.createDocument("", "",    ↓
null);
```

(2 premiers paramètres facultatifs : namespace et racine du doc.
Le 3ème n'existe pas)

Pour charger un fichier (comme sous IE)

```
xmlDoc.load('Document.xml');
xmlObj = xmlDoc.documentElement;
```

Pour charger une variable contenant du XML sous forme texte il faut passer par un analyseur de XML (DOMParser) que l'on crée comme objet puis utiliser la méthode parseFromString('chaine à parser','typage mime de la chaîne')

```
var parser = new DOMParser();
```

```
xmlDoc = parser.parseFromString(MaVarXML,"text/xml");
```

L'objet DOM XML

Le DOM met à disposition de javascript la page HTML sous forme d'un objet XML.
On peut donc travailler le document comme un objet XML avec l'ensemble des propriétés et des méthodes lié aux objets XML

Méthodes

createElement(nom_du_tag) crée un noeud nommé "nom_du_tag"
createTextNode("text") crée un noeud texte contenant le texte "text"
getElementById("test") retourne l'élément dont l'id est "test"
getElementsByTagName("nom") retourne un tableau des éléments dont le nom de balise est "nom"
getElementsByName("nom") retourne un tableau des éléments dont le nom est "nom"

Propriétés

DocumentElement : retourne l'objet racine du document.
nodeType retourne le type de l'élément (1 noeud, 2 attribut, 3 texte, 9 document)
nodeName retourne le nom du noeud (si type est 1 nom de la balise, si type est 2 nom de l'attribut, sinon #text ou #document)
nodeValue retourne la valeur de l'élément (si type est 2 ou 3 retourne la valeur, sinon rien)
tagName retourne le nom de balise de l'élément si type = 1 (utiliser plutôt nodeName)
parentNode : retourne l'objet parent de l'élément

Navigation dans le l'arbre du document

Propriétés

childNodes tableau des enfants du noeud courant (childNodes[1] est le second enfant)
children tableau des enfants éléments du noeud courant (children[1] est le second enfant)
firstChild premier noeud enfant
lastChild dernier noeud enfant
nextSibling le noeud enfant suivant
parentNode le noeud parent
previousSibling le noeud enfant précédent
sourceIndex le numéro d'index du noeud dans la page source

Manipulation des noeud

Méthodes

appendChild(y) ajoute le noeud y au noeud courant
y.cloneNode(true | false) crée une copie du noeud y (si true tout l'arbre y est copié si false seul la racine de y est copié)

insertBefore(y,z) insert y dans le noeud courant avant z
removeChild(y) supprime le noeud y
replaceChild(y,z) remplace le noeud z par le noeud y

Manipulation des données (Le noeud courant doit être un noeud texte)

Méthodes

appendData("Mon texte à moi") ajoute le texte au noeud courant
deleteData(4,3) supprime 3 caractères du texte dans le noeud courant à partir du cinquième
insertData(4,"ajout de texte") insert après le quatrième caractère du texte du noeud courant le texte "ajout de texte" (début en position 5)
replaceData(4,3,"du nouveau texte ") remplace 3 caractères du texte du noeud courant après le quatrième caractère par le texte "du nouveau texte"
substringData(4,3) retourne 3 caractères à partir du cinquième du texte du noeud courant

Propriété

data retourne le texte (pareil que nodeValue)

Manipulation des attributs (cle = "valeur")

Méthodes

getAttribute("cle") renvoie la valeur de l'attribut "cle" du noeud courant
setAttribute("cle","valeur") met l'attribut "cle" à la valeur "valeur" IE ne change pas les style et supprime les événements
removeAttribute("cle") supprime le ou les attributs "cle" du noeud courant IE ne supprime pas les événements

x.getAttributeNode("cle") renvoie le noeud attribut "cle" du noeud courant
x.hasAttribute("cle") vrais si le noeud courant a un attribut "cle" pas chez IE
x.hasAttributes() vrais si le noeud courant a un ou des attributs pas chez IE
name renvoie le nom de l'attribut

Propriétés

attributes retourne un tableau des attributs interrogeable par index ou par étiquette (attributes[0] ou attributes["cle"]) pas terrible chez IE
value renvoie la valeur de l'attribut

Divers

Méthodes

createDocumentFragment() crée un bout de document auquel on peu ajouter autant de noeud que

l'on veut et ensuite on l'insert dans le document

```
var x = document.createDocumentFragment();
var y = x.appendChild(document.createElement(`h4'));
y.appendChild(document.createTextNode(`Hello'));
```

hasChildNodes() vrais si le noeud courant a des enfants
document.implementation.hasFeature("XML","1.0") vrais si le browser supporte XML 1.0
item(x) retourne le sous-objet dont la position dans le tableau des sous-objets et à l'index x (attention pas de crochets mais des parenthèses)
splitText(5) coupe le texte du noeud texte courant au sixième caractère. Le noeud courant contient donc les caractères 0 à 5. Le reste du texte est mis dans un nouveau noeud en position `noeud courant'.nextSibling

Propriétés de position et dimension d'un élément

La position d'un élément (coordonnées left et top) et donnée par offset par rapport à l'élément parent (la balise qui contient cette élément)
offsetLeft distance de l'élément au bord gauche de la balise parente
offsetTop distance de l'élément au bord haut de la balise parente
offsetParent retourne l'élément parent
offsetWidth largeur de l'élément
offsetHeight hauteur de l'élément

DOM XML appliqué aux tableaux

Les tableaux étant constament présent dans les pages HTML et présentant une construction différente entre IE et firefox il a été créé un objet tableau qui facilite la manipukation des contenus lignes et cellules.

Manipulation d'un tableau via DOM XML

Si on identifie via un id la balise <table> on peut récupérer l'arbre XML de construction du tableau avec document.getElementById('id').

La structure de cet arbre fait apparaître un enfant 'TBODY' premier enfant de 'TABLE'

TABLE

TBODY

TR

TD …

Lorsque l'on veut insérer un nouvel élément il faut en tenir compte avec IE.

Firefox accepte que l'on n'en tienne pas compte.

Attention : Le DOM XML de Firefox tient compte des noeux texte créés par les retours à la ligne du code

Pour que le code soit passe-partout on récupère TBODY via getElementsByTagName

```
var obj_aff = document.getElementById('tab_aff');
obj_aff = obj_aff.getElementsByTagName('TBODY')[0];
```

On crée ensuite un élément TR

```
letr =  document.createElement("tr");
```

Puis on crée les éléments TD et on lui ajoute le contenu texte

```
letd =  document.createElement("td");
contenu = document.createTextNode(txt);
letd.appendChild(contenu);
```

Et on ajoute l'élément TD à l'élément TR

```
letr.appendChild(letd);
```

Une fois la ligne construite on l'ajoute à l'objet tableau

```
obj_aff.appendChild(letr);
```

Ex :
A partir d'un fichier XML de sites marchand "boutique.xml" on ajoute dans un tableau HTML les lignes de chaque marchand du fichier en affichant le nom, le message publicitaire et l'adresse du site

Fichier "boutique.xml"

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<articles>
<marchand id="m1">
    <nom_marchand>Dupond et Fils</nom_marchand>
    <publicite><![CDATA[Dupond et Fils annoncent leurs soldes <b>Tout
<i>doit</i> partir!</b>  moins 60%!]]></publicite>
     <site>dupond-fils.fr</site>
</marchand>
<marchand id="m2">
    <nom_marchand>Tartenpion</nom_marchand>
    <publicite><![CDATA[Chez Tartenpion <b>remise permanente</b> de moins
30%!]]></publicite>
     <site>tartenpion.fr</site>
</marchand>
<marchand id="m3">
    <nom_marchand>Moins Cher</nom_marchand>
    <publicite><![CDATA[tout est moins cher chez <b><i>Moins
Cher</i></b>!]]></publicite>
     <site>moinscher.com</site>
</marchand>
<marchand id="m4">
    <nom_marchand>CD Discount</nom_marchand>
    <publicite><![CDATA[CD Discount cassent les prix <b>sur tous <i>les
CDs</i>!</b> de moins 15 &agrave; moins 50 %!]]></publicite>
     <site>CD-Discount.com</site>
</marchand>
</articles>
```

Page HTML

```
<html>
<head>
```

```
<title>Dom</title>
<script language="javascript">
<!--
     function lit_xml()
     {
      if (document.implementation &&
document.implementation.createDocument)
      {
            xmlDoc = document.implementation.createDocument("", "", null);
            xmlDoc.onload = recupDonnees;
            xmlDoc.load("boutique.xml");
      }
      else if (window.ActiveXObject)
      {
            xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
            xmlDoc.async = "false";
            xmlDoc.load("boutique.xml");
            recupDonnees();
      }
      else
      {
            alert('Probl&egarve;me avec le chargement du fichier de
donn&eacute;es!');
            return false;
      }
     }

     function recupDonnees()
     {
             var aff = '';
            var obj_aff = document.getElementById('tab_aff');
            obj_aff = obj_aff.getElementsByTagName('TBODY')[0];
            var marchand  = xmlDoc.getElementsByTagName('marchand');
            for (i = 0; i < marchand.length; i++)
            {
                  letr =  document.createElement("tr");
                  var donnee = marchand[i].childNodes;
                  for (j = 0; j < donnee.length; j++)
                  {

                        var valeur = donnee[j].childNodes;
                        if (valeur.length != 0)
                        {
                              txt = valeur[0].data
                              letd =  document.createElement("td");
                              contenu = document.createTextNode(txt);
                              letd.appendChild(contenu);
                              letr.appendChild(letd);
                        }

                  }
                  obj_aff.appendChild(letr);
            }

     }
```

```
-->
</script>
</head>
<body onload="lit_xml()">
<div id="aff" >
<table border="1" cellspacing = "0" cellpadding = "5" id = "tab_aff">
<tr><th>Boutique</th><th>Message</th><th>site</th></tr>
</table>
</div>
</body>
```

Manipulation via l'objet Table

L'objet Table est une classe ayant comme enfant l'objet ROW qui lui même est père de l'objet CELL

Classe Table :

Propriétés
rows : Tableau d'objets TableRow des lignes du tableau HTML (balise <tr>)
ElementCaption : Titre du tableau HTML (balise <caption>)

Méthodes
ElementCreateCaption() : Crée ou récupère le titre du tableau HTML (balise <caption>)
deleteCaption() : supprime le titre du tableau HTML (balise <caption>)
insertRow(index_avant) : ajoute une ligne avant celle d'index "index_avant" dans le tableau rows et la retourne sous forme d'un objet TableRow. Le tableau HTML gagne une ligne vide
deleteRow(index) : supprime la ligne d'index "index" dans le tableau rows

Classe TableRow :

Propriétés
cells : Tableau d'objets TableCell des cellules de la ligne (balise <td>)
rowIndex : index de la ligne dans le tableau rows de l'objet Table

Méthodes
insertCell(index_avant) : ajoute une cellule avant celle d'index "index_avant" dans le tableau cells et la retourne sous forme d'un objet TableCell. Le ligne du tableau HTML gagne une cellule vide
deleteCell(index) : supprime la cellule d'index "index" dans le tableau cells

Classe TableCell :

Propriétés
cellIndex : index de la cellule dans le tableau cells de l'objet TableRow
innerHTML : Texte sous forme HTML de la cellule

Ex :
Changer la page HTML de l'exercice précédent pour utiliser l'objet Table

```
function recupDonnees()
{
     var aff = '';
```

```
        var obj_aff = document.getElementById('tab_aff');
        obj_aff = obj_aff.getElementsByTagName('TBODY')[0];

        var marchand  = xmlDoc.getElementsByTagName('marchand');
        for (i = 0; i < marchand.length; i++)
        {
            letr =  obj_aff.insertRow(i+1); //insert une ligne
            var donnee = marchand[i].childNodes;
            letd =  "";
            for (j = 0, pos = 0; j < donnee.length; j++)
            {
                 var valeur = donnee[j].childNodes;
                 rech = /[\s]+/g
                 if (valeur.length != 0)
                 {
                      txt = valeur[0].data
                      if (txt.replace(rech,'')!='')
                      {
                           letd = letr.insertCell(pos);
                           letd.innerHTML = txt;
                           pos++;
                      }
                 }
            }
        }
}
```

objet de communication client/serveur (AJAX)

De nouvelles applications web ont vu le jour récemment: recherche avec complétion automatique, sauvegarde instantanée d'informations, interface mail hyperdynamique, cartographie, etc. mais aucune ne fait appel à des plugins tiers comme Flash ou Java. On parle alors souvent de méthode AJAX qui s'articule essentiellement autour de Javascript et d'un objet (dans le sens de la programmation orientée objets) en particulier: XMLHttpRequest.

Créé par Microsoft pour Internet Explorer, l'objet XMLHttpRequest a été adopté par les navigateurs Mozilla, Konqueror, Safari et récemment Opéra. Bien que largement implémentée dans les navigateurs récents, **cette technologie n'est pas un standard du W3C** , lequel propose des fonctionnalités similaires à travers la recommandation Document Object Model (DOM) Level 3 Load and Save Specification.

Cet objet permet de faire des requêtes HTTP afin de récupérer des données au format XML qui pourront être intégrées à un document. Cela peut être très utile pour mettre à jour des données sans pour autant recharger la page.

Les avantages possibles :

- Diminution de la bande passante : seules les données sont chargées et non plus tout le document ;
- Interactivité accrue : plus de rechargement de la page ;
- Rationalisation du code : des routines (de vérification par exemple) n'ont plus à être écrites et maintenues dans deux langages (côté client et côté serveur).

Les inconvénients possibles :

- Ne fonctionne pas sans Javascript, ni dans les navigateurs les plus anciens, ni trop sécurisés ;
- Ne fonctionne qu'avec HTTP : il est impossible de récupérer des données sur un disque local (ce qui est normal) ;
- Les requêtes en dehors du domaine provoquent un avertissement de sécurité ;
- Peut empêcher des comportements habituels du navigateur :
  - Marques-pages et liens vers la page ;
  - Enregistrement des pages (moteurs de recherche);
  - Bouton retour.

Pour créer une communication entre le client et le serveur internet on ouvre un flux avec un objet le liaison.
Suivant les navigateurs on utilise

- Soit l'activeX Microsoft.XmlHttp,
- Soit l'objet XMLHttpRequest.

Une fois la liaison établie on ouvre la communication en méthode get ou post puis on envoie la requête au serveur qui retourne une page XML ou une donnée texte (tableau JSON ou texte de type CSV).
Il suffit d'utiliser le DOM XML dans le cas de page XML pour mettre dynamiquement à jour la page HTML.
Attention
Le rafraîchissement de la page se faisant via le javascript, il n'y a pas de rechargement de la page.
Comme la page n'est pas rechargée, on ne peut pas l'indexer (favoris) et les moteurs de recherche n'ont accès qu'à une seule page dans l'état dou elle s'affiche au chargement.

Méthode de l'objet XMLHttpRequest

XMLHttpRequest.open() : Initie une requête XMLHttpRequest
XMLHttpRequest.send() : Exécute XMLHttpRequest()
XMLHttpRequest.abort() : Abandonne la requête XMLHttpRequest
XMLHttpRequest.setRequestHeaders() : permet de spécifier une valeur d'en-tête HTTP pour la liaison au serveur.
XMLHttpRequest.getAllResponseHeaders() : Récupère tous les entêtes de la requête XMLHttpRequest au serveur
XMLHttpRequest.getResponseHeader() : Récupère uniquement les entêtes 'status' et 'readyState' de la requête XMLHttpRequest au serveur

Propriétés de l'objet XMLHttpRequest

XMLHttpRequest.onreadystatechange : Gestionnaire d'événements pour les changements d'état. Pour une connexion asynchrone il faut lui assigner une fonction de traitement
XMLHttpRequest.readyState : Statut de l'état de l'échange avec le serveur
0 non initialisé
1 ouverture de la connexion suite à l'appel open()
2 requête envoyé suite à l'appel send()
3 requête en cours
4 terminé (le serveur à renvoyé une réponse à la requête)
XMLHttpRequest.statusText : Description textuelle du code de retour.
XMLHttpRequest.responseTest : Chaîne de caractères retourné par le serveur en réponse à la demande
XMLHttpRequest.responseXML : Objet DOM XML retourné par le serveur en réponse à la demande

Java Script
les bases

Créer l'objet

Pour internet explorer sous windows on utilise l'activeX XmlHttp

```
xmlhttp = new ActiveXObject("Microsoft.XmlHttp");
```

Pour les autres browser on crée un objet XMLHttpRequest

```
xmlhttp = new XMLHttpRequest();
```

Ex :

```
function creer_obj_requete()
{
var xmlhttp; //variable objet ajax de requête
try
      {
       /* microsoft 1 */
       xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
      }
      catch (e1)
      {
       try
       {
             /* microsoft 2 */
             xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
       }
       catch (e2)
       {
             if (typeof XMLHttpRequest != "undefined") {
                   /* les autres */
                   xmlhttp = new XMLHttpRequest();
             }
             else
             {
                   /* L'objet xmlHttpRequest n'est pas implémenté */
                   return = false;
             }
       }
      }
return xmlhttp; //retourne l'object
}
```

Ouvrir l'objet

Une fois l'objet xmlhttp créé :
On ouvre la liaison en donnant la méthode, l'adresse de la page du serveur à interroger et true pour une liaison asynchrone

```
xmlhttp.open("post", adr, true);
```

On crée les entêtes de liaison

```
xmlhttp.setRequestHeader('Content-Type' ,'application/x-www-form-
urlencoded');
```

On envoie la requête

```
xmlhttp.send(val);
```

et on donne la fonction de récupération pour traiter le retour

```
xmlhttp.onreadystatechange = recup_result;
```

Ex:

```
function appel(val,adr)
{
     xmlhttp.open("post", adr, true); //ouverture asynchrone
     xmlhttp.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');
     xmlhttp.send(val);
     xmlhttp.onreadystatechange = recup_result;
}
```

Remarque :
Avec une requête synchrone, le script attend la réponse avant de continuer, donc la réponse est disponible dès la ligne suivante.
Si la requête est asynchrone, tous les changements d'états de la requête seront signalés à la fonction de retour déclarée, et le récupération se fera donc à ce niveau.

Requête Synchrone

```
xmlHTTP.open('get', uri, false);
xmlHTTP.send(null);
alert(xmlHTTP.responseText);
```

Requête Asynchrone

```
xmlHTTP.open('get', uri, true);
xmlHTTP.onreadystatechange = retour
xmlHTTP.send(null);

function retour()
{
     if(xmlHTTP.readyState == 4) {
      alert(xmlHTTP.responseText);
     }
}
```

Récupération du document de retour (XML)

On vérifie grâce à la propriété readyState que le document est chargé en entier (retourne 4):
0: non initialisé
1: charge
2: à chargé
3: interactif
4: terminé

```
if (xmlhttp.readyState == 4)
```

on vérifie le statut de la connexion de retour du serveur (200 pou ok)

```
if (xmlhttp.status == 200)
```

Si le retour est fini et que la page a été retourné (status 200) on charge le document XML de retour dans l'objet

- **soit à partir du texte brut avec la propriété responseText de retour**
  Pour internet explorer sous windows (activeX) en créant un objet XML et en mettant le texte de retour dedans

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
xmlDoc.async = "false" ;
xmlDoc.loadXML(xmlhttp.responseText);
```

  Pour les autres en inplémentant un objet XML et en lui assignant le retour sous forme XML

```
parser = new DOMParser();
var  xmlDoc = parser.parseFromString(xmlhttp.responseText,"text/xml");
```

- soit directement en tant qu'objet XML avec la propriété responseXML (entêtes XML obligatoires)

Ex avec XML sous forme texte :

```
function recup_result()
{
     if (xmlhttp.readyState == 4) // 4 : état "complete"
    {
        if (xmlhttp.status == 200) //200 : code HTTP pour OK
        {
           //Traitement de la réponse.
            try
            {     // cas microsoft IE
                  var   xmlDoc      =     new ActiveXObject
( "Microsoft.XMLDOM" ) ;
                  xmlDoc.async = "false" ;
                  xmlDoc.loadXML(xmlhttp.responseText);
             }catch(e){        // autres cas comme firefox
                  parser = new DOMParser();
                  var   xmlDoc =
parser.parseFromString(xmlhttp.responseText,"text/xml");
             }
          return  xmlDoc
        }
    }
}
```

Ex avec XML sous forme document XML

```
function recup_result()
{
     if (xmlhttp.readyState == 4) // 4 : état "complete"
    {
        if (xmlhttp.status == 200) //200 : code HTTP pour OK
        {
           //Traitement de la réponse.
           var xmlDoc = xmlhttp.responseXML;
        }
    }
     return  xmlDoc
}
```

:
On a alors un objet XML (xmlDoc) que l'on peut traiter avec le DOM XML du browser

Code complet :
A l'initialisation l'objet de requête est créé et le navigateur détecté via try catch.
La fonction appel(valeur,adresse,type_de_retour) est appelé pour créer le flux de requête réponse en passant comme attribut la valeur des variables et l'adresse de la page appelée sur le serveur.
On travaille en mode synchrone

Fichier xml appelé (doc.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<log><nom>Toto</nom><prenom>Jean</prenom><mdp>123</mdp></log>
```

Page html

```
<html>
<head>
<title>formulaire</title>
<link href="style.css" rel="stylesheet" type="text/css" />
<script language="JavaScript" type="text/JavaScript">
<!--
type_retour = "XML";

function creer_obj_requete()
{
     var xmlhttp ;
     try
     {
      /* microsoft 1 */
      xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
     }
     catch (e1)
     {
      try
      {
            /* microsoft 2 */
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch (e2)
      {
            if (typeof XMLHttpRequest != "undefined") {
                  /* les autres */
                  xmlhttp = new XMLHttpRequest();
            }
            else
            {
                  /* L'objet xmlHttpRequest n'est pas implémenté */
                  xmlhttp = false;
            }
      }
     }
```

```
     return  xmlhttp;
}

function appel(val,adr)
{
     xmlhttp.open("post", adr, false); //ouverture asynchrone
     xmlhttp.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');
     xmlhttp.send(val);
     /*xmlhttp.onreadystatechange = recup_result;
}

function recup_result()
{*/
     if (xmlhttp.readyState == 4) // 4 : état "complete"
     {
      if (xmlhttp.status == 200) //200 : code HTTP pour OK
      {
            //Traitement de la réponse.
            var xmlDoc ;
            switch (type_retour)
            {
            case  "XML" :
                  xmlDoc = xmlhttp.responseXML;
                  break ;
            case  "txt" :
                  try
                  {     // cas microsoft IE
                        xmlDoc      =     new ActiveXObject
( "Microsoft.XMLDOM" ) ;
                        xmlDoc.async = "false" ;
                        xmlDoc.loadXML(xmlhttp.responseText);
                 }catch(e){         // autres cas comme firefox
                        var parser = new DOMParser();
                        xmlDoc =
parser.parseFromString(xmlhttp.responseText,"text/xml");
                  }
                  break ;
            }
            affiche(xmlDoc);
      }
     }
}

function affiche(xmlDoc)
{
     aff = xmlDoc.documentElement.childNodes;
     aff_tab = "<table border=\"1\" cellspacing=\"0\" width=\"250\">";
     for (i=0; i<aff.length; i++)
     {
     var mavar = aff[i].nodeName;
     var maval = aff[i].childNodes[0].nodeValue
     aff_tab += "<tr><td>" + mavar + "</td><td>" + maval + "</td></tr>";
     }
```

```
     aff_tab += "</table>";
     document.getElementById("affiche").innerHTML = aff_tab;
}

function voir()
{
     //type_retour = "txt";
     xmlhttp = creer_obj_requete();
     if (type_retour == 'XML')
     {
          appel('','./doc.xml');
          alert('XML')
     }
     else
     {
          appel('','./doc.txt');
          alert('txt')
     }
}
//-->
</script>
</head>
<body>
<div id="affiche" name="affiche" style="position:absolute; left:50px;
top:100px; width:400px; height:200px;">
</div>
<input type="button" name="propriété" value="appel" onclick="voir()" />
</body>
</html>
```

Les feuilles de Style

Java Script
les bases

Principe des feuilles de style

Une feuille de style définit une ou plusieur règles qui décrivent comment un ou des éléments d'une page HTML doivent être affichés.
La règle s'écrit sous forme
sélecteur {déclaration}
ex

```
sélecteur   déclaration
     h1     {color:#FF0000;}
```

Cette règle écrira en rouge tout les titre en h1

Le sélecteur :

il peut être :

- sélecteur d'élément.

Il fait référence à un type de balise et définit sa présentation
ex : Pour définir le gras en blanc sur un fond rouge

```
b {color: #FFFFFF;background-color: red;}
```

- sélecteur contextuel.

Il ne s'applique que dans le contexte défini (ensemble de balises imbriquées)
ex : Pour définir le gras dans un paragraphe p en blanc sur un fond rouge

```
p b {color: #FFFFFF;background-color: red;}
```

- sélecteur de classe

Il définit une classe en lui donnant un nom. La classe peut être appelé à l'intérieur de n'importe quel balise avec l'attribut style
ex : Pour définir une classe attention en blanc sur un fond rouge

```
.attention {color: #FFFFFF;background-color: red;}
```

On appelle dans n'importe quel balise la classe attention via l'attribut style

```
<p style="attention">Bla bla bla ....</p>
```

- sélecteur d'id

On nomme un identifiant (id) qui sera utilisé dans une balise. On lie une déclaration à cet identifiant et le style s'applique à la balise identifiée de cette manière.
ex

```
#toto {
     color: red;
     background-color: #FFFFFF;
}
```

Et dans la page

```
<p id="toto">bla bla ....</p>
```

- sélecteur de pseudo-classes et pseudo-éléments

Il existe pour la balise <a> en ensemble de sélecteurs liés aux états de la balise et séparé de a par :
C'est ce qu'on appelle une pseudo-classe

a:link	correspond à l'attribut link de body
a:visited	correspond à l'attribut vlink de body
a:hover	correspond au survol du lien
a:active	correspond à l'attribut alink de body

De même on peut définir des états du contenu liés aux balises de mise en page comme first-letter ou first-line.
C'est ce qu'on appelle un pseudo-élément
ex

```
p:first-letter {font-size: 150%;}
```

La déclaration :

Elle est entourée d'accolade {.....}
Elle est composée de couple propriété valeur séparé par :
Chaque couple propriété:valeur est terminé par ;
ex

```
body {
     font-family: Arial, Helvetica, sans-serif;
     color: #000044;
     background-color: #FFFFCC;
     font-size: 10px;
}
```

La mise en place du style :

Les règles de style peuvent être appliquées aux documents de 3 manières

- en ligne
  On utilise la balise en lui adjoignant via l'attribut style la définition du style. Ce style ne s'applique qu'a cette balise
  ex

```
<p style="font-size:12px;color:red;">Bla bla ..  </p>
```

- en feuille de style interne
  On utilise la balise <style> dans la partie <head> du document et on défini les styles à utiliser. Ces styles ne s'appliquent qu'à cette page.
  ex

```
<style type="text/css">
<!--
body {font-family: Arial, Helvetica; color: #000044; background-
color: #FFFFCC; font-size: 10px;}
p:first-letter {font-size: 150%;}
-->
</style>
```

- en feuille de style externe

Les styles sont définis dans un fichier externe et liés via une balise <link> à la page
La feuille de style peut de cette manière être utilisée sur l'ensemble des pages du site.
ex :

```
<link href="Monstyle.css" rel="stylesheet" type="text/css" />
```

Les valeurs dans les CSS :

Il existe 5 types de valeurs de propriétés

mots-clés

Ce sont des valeurs explicites telles que dotted pour border-style ou bolder pour font-weight

valeurs de longueur

Elles peuvent être absolues (obligation de mentionner les unités)

- Pouces (in)
- Centimètres (cm)
- Millimètres (mm)
- Points il y a 72 points dans 1 pouce (pt)
- Picas 1 picas = 12 points (pc)
- Pixels (px)

Elles peuvent être relative

- hauteur-em (em) hauteur relative à la police en cours 1.2em=120%
- hauteur-X (ex) c'est la hauteur de l'oeil de la lettre (valeur d'une minuscule).

Approximativement on considère que 1ex = 0.5em

valeurs en pourcentage

C'est une valeur qui est calculée en pourcentage de la valeur courante
font-size:200% affiche une police au double de la taille courante

couleurs

Les couleurs peuvent être spécifier de 5 manières différentes

Hexadécimal #rrvvbb c'est comme les couleur html

```
ex    noir              #000000
```

Hexadécimal abrégé #rvb les valeurs r v et b sont doublées. #B8C est équivalent à #BB88CC

rgb(rx%,gx%,bx%) les valeurs de rouge, vert, bleu sont exprimées en % allant de 0;00% à 100.00%

```
ex    bleu pur          rgb(0%,0%,100%)
```

rgb(rrr,ggg,bbb) les valeurs de rouge, vert, bleu sont exprimées par un nombre compris entre 0 et 255

```
ex    blanc             rgb(255,255,255)
```

Mots-clés C'est les mots-clés des couleur du WEB

```
ex    rouge       red
```

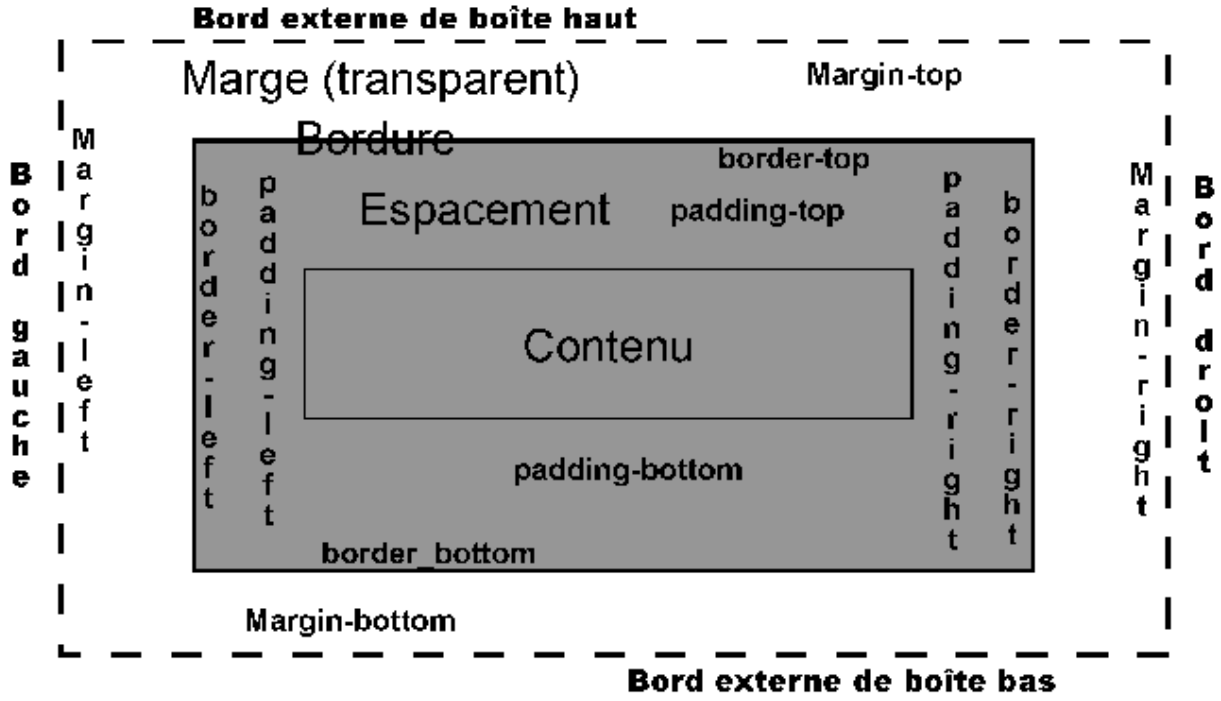
url

Elles sont au format suivant : url(adresse_de_fichier)

```
ex : background-image: url(img/fond.jpg);
```

Boites :

Les éléments des styles CSS appliqués aux balises de type bloc génèrent une boîte rectangulaire appelée boîte de l'élément : c'est l'espace total occupé par l'élément



La boite est composée :

du bord externe, c'est le bord qui se trouve en contact avec le bord externe d'un autre élément consécutif

d'une marge (margin) c'est l'espace entre deux éléments consécutifs

d'une bordure c'est la limite de l'élément (limite de la couleur de fond)

d'un espacement (padding) c'est la distance entre la bordure et le contenu

d'un contenu

Comportement de flottement :

Le flottement permet de faire défiler du texte autour d'un élément à sa gauche ou à sa droite. Avec une propriété "float", un élément sort du flux normal des autres éléments et acquiert un format de type bloc. Par exemple, en donnant la valeur ‘left’ à la propriété ‘float’ d’une image, celle-ci est repoussée vers la gauche jusqu’à buter sur les marges, espacements ou bordures d’un autre élément de type bloc. Le flux normal se déroule sur le côté droit de l’image. Ses marges, bordures et espacements sont respectés, cependant les marges ne fusionnent jamais avec celles des éléments adjacents.

Ex

Pour créer une lettrine

```
p:first-letter {
     font-size: 200%;
     font-family: Arial, Helvetica, sans-serif;
     padding: 2px;
     background-color: #336600;
```

```
        color: #CC9900;
        float: left;
    }
```

Références des propriétés CSS

Pseudo-classes et pseudo-événements

:link	correspond à l'attribut link
:visited	correspond à l'attribut vlink
:hover	correspond au survol par la souris
:active	correspond à l'enfoncement de l'élément
:focus	correspond à l'activation de l'élément

Les pseudo-classes des liens sont à utilisées à la place des arguments de liens dans la balise body. En théorie il peuvent s'appliquer à n'importe quel éléments gérant un lien. En pratique ils ne sont reconnu par les navigateurs que pour <a>

:first-line	première ligne d'un texte
:first-letter	première lettre de l'élément
:first-child	première balise enfant (inclue) de l'élément
:before {content: "texte"}	écrit "texte" avant l'élément
:after {content: "texte"}	écrit "texte" apres l'élément

Références de boite

marge (margin) **valeur de la marge**

margin	la valeur s'applique à toutes les marges
margin-top	valeur de la marge haute
margin-right	valeur de la marge droite
margin-bottom	valeur de la marge basse
margin-left	valeur de la marge gauche

bordure (border)

border-width valeur ou mots-clés : thin, medium, thick

border-style mots-clés none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

border-color valeur de couleur

Comme pour margin on peut définir la bordure pour 1 coté ex border-top-width: 1px

espacement (padding)

padding valeur de l'espacement contenu bordure

padding-top etc

fond (background)

background-color	valeur de couleur
background-image	url("adresse de l'image")
background-repeat	repeat ou repeat-x ou repeat-y ou no-repeat
background-attachement	scroll ou fixed
background-position	valeur de position x position y

contenu et dimensions

height	valeur de la hauteur
width	valeur de la largeur
color	valeur de la couleur d'avant plan. Si le contenu est du texte c'est la couleur de texte

les références de boites s'applique sur tout éléments de type bloc

références de présentation (curseur ascenseur)

Ascenseur (propre à IE)

scrollbar-face-color	valeur de couleur
scrollbar-arrow-color	valeur de couleur
scrollbar-track-color	valeur de couleur
scrollbar-3dlight-color	valeur de couleur
scrollbar-darkshadow-color	valeur de couleur
scrollbar-shadow-color	valeur de couleur
scrollbar-highlight-color:	valeur de couleur

Curseur

cursor permet de définir le curseur de la souris

- pointer Le curseur représente un doigt indiquant un lien.
- hand Le curseur représente un doigt indiquant un lien.
- move Indique un objet qu'on peut déplacer.
- e-resize Curseur pointant vers l'est.
- ne-resize Curseur pointant vers le nord-est.
- nw-resize Curseur pointant vers le nord-ouest.
- n-resize Curseur pointant vers le nord.
- se-resize Curseur pointant vers sud-est.
- sw-resize Curseur pointant vers le sud-ouest.
- s-resize Curseur pointant vers le sud.

- w-resize Curseur pointant vers l'ouest.
- text Indique qu'on peut sélectionner le texte. Souvent une barre en I.
- wait Indique une progression. Souvent une montre ou un sablier.
- help Souvent un point d'interrogation ou une bulle.
- default Curseur par défaut selon la plate-forme. Souvent une flèche.
- crosshair Une marque en croix (ex. deux traits formant un signe "+").
- url Spécifie un fichier spécial. On donne l'adresse de l'image .cur
- auto Le navigateur détermine quel curseur prendre selon le contexte.

Une application des CSS : Les calques

Les feuilles de styles permettent de positionner un élément dans une page.
Ce positionnement peut être absolu , relatif ou statique

position: static :

L'élément est placé à la position normale dans le flux d'affichage

position: relative :

L'élément est positionné en x et y par rapport à sa position normale dans le flux d'affichage
ex
style="position:relative;left:100px;top:150px;" positionne la balise ayant cet attribut style et son contenu à 150 pixels
en dessous et 100 pixels à droite de l'endroit ou il se serait
positionné s'il n'y avait pas eut d'attribut style.

position: absolute :

L'élément est positionné en x et y par rapport au bord haut gauche de l'élément parent. S'il n'a pas de parent il est positionné par rapport au coin haut gauche de la fenêtre d'affichage (C'est le parent de tous les éléments de la page)
ex

```
<body>
<h1 style="position:absolute ;left:100px;top:150px;">
Hello
</h1>
```

positionne le titre Hello à 150 pixels en dessous et 100 pixels à droite du coin haut gauche de la fenêtre d'affichage.

Via les style on peut positionner un élément dans une pile grâce à la propriété z-index et gérer la visibilité via la propriété visibility que l'on peut mettre à visible ou à hidden.
ex

```
style="position:absolute
;left:100px;top:150px;visibility:visible;z-index:5"
```

crée un style comme attribut d'une balise élément positionné à 100 pixels du bord gauche et

150 pixels du bord haut de l'élément parent qui sera visible et au dessus des éléments de z-index inférieur à 5 et en dessous des autres éléments de z-index supérieur à 5

Si on applique ce style à une balise qui est la balise HTML qui sert à tout on obtient un conteneur positionnable.
C'est le principe adopté pour les calques
Ex

```
<body>
<div id="aff" style="position:absolute; width:301px;
height:300px; z-index:1; visibility: visible; left:
200px; top: 150px;" >
<img src="logo.gif" width="301" height="77" />
</div>
</body>
```

Le positionnement peut se faire soit directement dans la balise à l'aide de l'attribut style="style écrit sous forme css"
Il peut être externalisé afin de séparer le code du corps de la page de la présentation. On utilise la référence sur l'id via # pour appliquer le style à la balise
Ex

```
<head>
<style type="text/css">
<!--
#aff {
     position:absolute;
     left:200px;
     top:150px;
     width:301px;
     height:300px;
     z-index:1;
     visibility: visible;
}
-->
</style>
</head>
<body>
<div id="aff">
     <img src="logo.gif" width="301" height="77" />
</div>
</body>
```

La propriété de style display permet de gérer l'affichage de la balise en définissant sont type

- none pour sortir la balise de l'affichage
- block pour lui donner un comportement de type bloc
- inline pour lui donner un comportement de type en ligne
- inline-block pour lui donner un comportement de type bloc se positionnant en ligne (ne marche pas avec firefox)

La propriété de style overflow permet de gérer les dépassement du contenu par rapport au conteneur

- visible : Le contenu est complètement visible. Il n'est pas rogné par les dimensions de son conteneur.

- hidden : Le contenu est rogné par les dimensions de son conteneur. Tout ce qui dépasse est caché
- auto : Le conteneur est doté d'ascenseurs horizontaux et verticaux si le contenu dépasse les dimension du conteneur.
- scroll : Le conteneur est doté d'ascenseurs horizontaux et verticaux

References css de texte

font-family	nom de la police ou liste de police
font-size	taille de la police
font-style	normal ou italique ou oblique
font-variant	normal ou small-caps
font-weight	normal ou bold ou bolder ou lighter ou nombre (100 200 300 400 500 600 700 800 900)

400 = normal, 700= bold

letter-spacing	valeur d'espacement
word-spacing	valeur d'espacement
line-height	valeur de hauteur
text-decoration	none ou underline ou overline ou line-through ou blink
text-indent	valeur d'indentation
text-transform	capitalize uppercase lowercase ou none
text-align	right left center ou justify
vertical-align	valeur ou baseline sub super top text-top middle bottom ou text-bottom
white-space	normal pre ou nowrap

Style de liste css :

les listes sont des list-style

list-style-image	url("adresse url de l'image qui doit servir de puce")
list-style-position	inside ou outside suivant que la puce ou la numérotation est interne à l'alignement ou externe à celui-ci
list-style-type	disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha ou none

Aides

Balises HTML

LES BALISES HTML

balise Definition

attributs rôle

<html> </html> Engobe le document html dans sa totalité

<head> </head> Détermine la zone d'en-tête

<title> </title> Titre du document

<script> </script> Insert un script (Javascript ou VBscript)

<body> </body> Détermine la zone d'affichage

background = "url d'image" image de fond

bgcolor="#FFFFFF" Couleur du fond en hexadecimal

link="#0000FF" Couleur des liens

text="#000000" Couleur du texte

Vlink="#00FF00" Couleur des liens visités

<h1> </h1> Niveau de titre dans le texte (de 1 à 6)

** ** obsolète remplacé par un style

color="#FF0000" Couleur du texte

face="verdana, arial, helvetica, sans-serif" Police à utiliser

size = "3" Taille du texte (entre 1 et 7)

** ** Texte en gras (Bold)

** ** Texte en italique (Italie)

<u> </u> Texte souligné (Underline)

<s> </s> Texte barré (Stnike)

**** Mis en indice (Subscn t)

**** Mis en exposant (Su rscri t)

** ** Marque de paragraphe (retour + saut de ligne>

**
** Retour à la ligne

<blockquote> </blockquote> Mise en retrait d'un paragraphe

<img /> Insertion d'une image

src="url" URL de l'image (absolue ou relative)

alt Texte de remplacement

border Bordure d'image (à éviter)

height Hauteur de l'image

width Largeur de l'image

Align=(left right top middle bottom) Alignement de l'image par rapport au texte

hspace Espace vertical entre image et texte (en largeur)

vspace Espace horizontal entre image et texte (en hauteur)

**<a> ** Lien

href URL du lien

name Étiquette d'ancrage (référence d'une ancre)

<object> </object> Insertion d'un objet (animation, vidéo, son) activeX

data URL de l'objet

border ; width ; height ; Align; hspace ; vspace *Comme pour <img>*

<param /> Paramètre de <object>

<embed> </embed> Insertion d'un objet multimédia

src URL du fichier multimédia

align height width Comme <img>

Loop=(true ou false) Répétition (true) ou lecture unique (false)

** ** Liste à puces

type =("disc" ou "circle" ou "square") Type de puce

** ** Liste ordonnée - -

Type= ("a" ou "A" ou "1" ou 'I') Type de numérotation

** ** Valeur d'un élément de la liste

value Valeur initiale de la numérotation

** ** Isole un sous ensemble pour lui appliquer des paramètres. Balise de type inline

tableau

<table> </table> Mise en page d'un tableau

border Epaisseur en pixels de la bordure (0 par défaut)

width Largeur du tableau (en pixels ou en % de la fenêtre)

align Alignement dans la largeur de la page

bgcolor Couleur de fond du tableau

cellpadding Espace entre le contenu d'une cellule et les bords

cellspacing Espace entre les cellules

<tr> </tr> Ligne de cellules d'un tableau

align=(left center right) Alignement horizontal

valign=(top middle bottom) alignement vertical

bgcolor couleur des cellules de la ligne

<td> </td>

Align valign bgcolor

height

width

colspan Fusion des cellules de la même ligne

rowspan Fusion des cellules de la même colonne

formulaire

<form> </form> Création d'un formulaire

name Nom du formulaire

method Mode d'envoi du formulaire (POST ou GET)

action URL du programme de traitement du formulaire

<input /> Contrôle dans un formulaire

Type=(text, password, checkbox. radio, submit, reset, button, file, image, hidden)
Type du contrôle mis à disposition de l'utilisateur

checked="checked" Pour les types radio ou checkbox coche le bouton par défaut

Maxlength Pour les types text ou password nombre de caractères maximum saisissable

name Nom du contrôle

value Valeur initiale du contrôle

<select> </select> Liste déroutante

name Nom de la liste

multiple = (true ou false) Permet les choix multiples

<option /> Intitulé d'un élément de liste déroulante

value Valeur de l'élément (souvent différente de l'intitulé)

Selected="selected" Sélectionné par défaut

<textarea> </textarea> Zone de saisie de texte acceptant plusieurs lignes

name Nom de la zone

cols Largeur de la zone en nombre de caractères

rows Hauteur de la zone en nombre de lignes calques

** ** Balise de regroupement utilisée pour les calques. Balise de type block

id Référence du calque

style = "position:absolute; left:41px; top:465px: width: l77px: height:74px; z-index: 1"
mise en forme et positionnement du calque. c'est le style qui permet de mettre précisément les objets dans la page. Le z-index est l'ordre d'empilement des objets.