Minicours JavaScript

Cours hors-cadre de fin d'année

Stéphane Perret

Version 1.30

Table des matières

1 Introduction **1**
1.1 JavaScript et Java sont deux langages différents . . . 1
1.2 L'utilité de JavaScript . . . 1
1.3 Remarque à propos des vieux explorateurs . . . 1

2 Programmation en JavaScript **2**
2.1 Premier programme . . . 2
2.2 Les commentaires . . . 2
2.3 Les caractères spéciaux . . . 3
2.4 L'entête ou le corps d'un document HTML . . . 3
2.5 Les scripts externes . . . 3
2.6 Les variables en informatique . . . 4
2.6.1 L'assignation de variables en informatique . . . 4
2.7 Opérations logiques sur les variables . . . 5
2.7.1 Opérations arithmétiques . . . 5
2.7.2 Opérations de comparaison . . . 5
2.7.3 Opérateur sur les chaînes de caractères . . . 5
2.8 Les fenêtres de dialogue . . . 6
2.8.1 Demande d'information . . . 6
2.8.2 Demande de confirmation . . . 6
2.8.3 Alertes . . . 6
2.8.4 Application : comment trouver une erreur dans le code JavaScript 6
2.9 Les fonctions en JavaScript . . . 7
2.9.1 Fonctions . . . 7
2.9.2 Les fonctions (et constantes) mathématiques . . . 8
2.10 Les structures de contrôles . . . 8
2.10.1 La commande if ... else if ... else . . . 8
2.10.2 Une structure de contrôle condensée . . . 9
2.10.3 La commande switch . . . 10
2.10.4 La commande break . . . 10
2.11 Les boucles . . . 11
2.11.1 La commande while . . . 11
2.11.2 La commande do...while . . . 11
2.11.3 La commande for . . . 11
2.12 Les différents types de données . . . 12
2.12.1 Les booleans . . . 12
2.12.2 Les tableaux . . . 12
2.12.3 Les dates . . . 15
2.12.4 Les chaînes de caractères . . . 15

3 Interactivité en JavaScript **16**
3.1 Les boutons . . . 16
3.2 Les effets sur les images . . . 18
3.3 Les formulaires . . . 20
3.3.1 Un exemple de formulaire sans JavaScript . . . 20

4 Annexes et références **22**

1 Introduction

JavaScript est utilisé dans des millions de pages web afin d'améliorer leur conception. Il s'agit d'une couche de programmation supplémentaire qui vient s'ajouter au langage HTML[1]. Le code HTML est le langage de base que toute page Internet se doit d'utiliser : en plus de son rôle proche d'un traitement de texte, ce langage permet de surfer grâce aux liens hypertextes. Quant à JavaScript, il a été conçu pour donner plus d'interactivité aux pages HTML. Le mot script indique qu'il s'agit d'un langage de programmation simplifié qui s'exécute en local sur l'ordinateur qui est en train de lire la page web. Ce langage, comme l'HTML, ne nécessite l'achat d'aucune licence pour pouvoir l'utiliser.

Initialement, JavaScript a été développé par Netscape, mais maintenant la plupart des explorateurs permettant de naviguer sur Internet sont compatibles avec JavaScript.

1.1 JavaScript et Java sont deux langages différents

Il est important de bien préciser que même si les noms sont très proches, Java et JavaScript sont deux langages bien distincts, autant du point du vue de leurs concepts que de leur conception. Java est développé par Sun Microsystems et est un langage de programmation bien plus puissant et complexe que JavaScript. Java peut se comparer au langage `C++`.

1.2 L'utilité de JavaScript

Les possibilités de JavaScript sont multiples.

1. JavaScript livre aux concepteurs de pages web un outil de programmation avec une syntaxe élémentaire. Contrairement à un vrai langage, presque tout le monde peut insérer un petit bout de code JavaScript dans leur page web.
2. JavaScript permet l'utilisation de textes dynamiques dans une page web. Un texte dynamique est un texte qui change selon les conditions de son environnement (on peut par exemple afficher la date d'aujourd'hui).
3. JavaScript peut réagir à un événement particulier. Par exemple quand une page a terminé son chargement ou quand un utilisateur clique sur un élément HTML.
4. JavaScript peut lire et écrire des éléments HTML.
5. JavaScript peut être utilisé pour valider des données sans utiliser un serveur.
6. Les cookies peuvent être créés et lus par un programme en JavaScript.

1.3 Remarque à propos des vieux explorateurs

Certains vieux explorateurs n'ont pas les mêmes capacités d'interprétation du code JavaScript que ceux d'aujourd'hui. Ainsi, un bon script JavaScript devrait contenir des commandes qui indiquent si la version de JavaScript n'est pas à jour. Néanmoins, dans ce document et ceci dans le but d'alléger le code présenté, on se passera de ces commandes. Le lecteur trouvera tout de même certaines de ces commandes en annexe.

[1] HyperText Markup Language

2 Programmation en JavaScript

2.1 Premier programme

Voici le contenu complet d'un fichier `hello.html` qui affiche le texte `Hello World!`. Pour créer ce programme, if faut :

1. Un navigateur Internet (par exemple Internet Explorer ou FireFox).
2. Un éditeur de texte (comme notepad (sur PC) ou simpletext (sur Mac)).

Dans le fichier appelé `hello.html`, on tape le texte suivant (depuis l'éditeur de texte). Ensuite en l'ouvrant dans l'explorateur, le programme s'exécute tout seul. Voici le code : pour découvrir son effet, il suffit de suivre la démarche ci-dessus.

```
<html>
<body>
<script language="javascript">
  document.write("Hello World!");
</script>
</body>
</html>
```

La balise `<html> ... </html>` indique qu'il s'agit d'un document `HTML`. La balise `<body> ... </body>` indique qu'il s'agit du corps du document `HTML`.

La balise `<script language="javascript"> ... </script>` encadre le code JavaScript.

2.2 Les commentaires

Lorsqu'on programme, il faut décrire à l'aide de commentaires ce que le programme fait. Si on laisse de côté du code pendant 6 mois, il est peu probable que l'on se souvienne exactement de ce qu'on a fait (à quoi correspond ce nom de variable, que fait cette fonction de 30 lignes de codes).

Or, un commentaire est facile à mettre : tout le texte qui suit `//` n'est pas interprété par JavaScript.

Or, `//` ne fonctionne que pour une ligne. Si on veut écrire un commentaire de plusieurs lignes, on peut aussi utiliser `/*` pour commencer le commentaire et `*/` pour le finir. Voici les deux façons d'écrire un commentaire sur plusieurs lignes.

```
/* Ceci est un commentaire
sur plusieurs lignes */
```

```
// Ceci est un commentaire
// sur plusieurs lignes
```

Attention à ne pas confondre avec les balises de commentaires du langage qui sont les suivantes et qui fonctionnent indépendamment du nombre de lignes.

```
<!-- Ceci est un commentaire en HTML -->
```

Règle d'or des commentaires

Il n'est jamais inutile de commenter un programme!!!

2.3 Les caractères spéciaux

Dans le premier exemple, on a utilisé le script suivant afin d'écrire "Hello World !".

```
document.write("Hello World!");
```

Si on désire écrire des guillemets, on doit faire comprendre à JavaScript qu'il s'agit d'un caractère à afficher et non à interpréter. On utilise pour cela le caractère \.

```
document.write("Le titre est \"l\'algue \& le poisson\".");
```

2.4 L'entête ou le corps d'un document HTML

Lorsqu'on veut un programme qui doit s'exécuter avant que la page web s'affiche, on met le code JavaScript dans l'entête du document HTML (balise `<head> ... </head>`).

```
<html>
<head>
<script language="javascript">
  lignes de code
</script>
</head>
```

Lorsque le code JavaScript doit générer un bout de page web, on doit l'insérer dans le corps du document HTML.

```
<html>
<head>
  ...
</head>
<body>
<script language="javascript">
  lignes de code
</script>
</body>
```

Les deux types de scripts peuvent être mélangés (des bouts de scripts peuvent être mis dans l'entête et d'autres dans le corps d'une page web).

2.5 Les scripts externes

On peut mettre les scripts dans un fichier à part de la manière suivante. L'avantage de cette manière de procéder est que le script peut être utilisé sur plusieurs pages HTML de manière simple et efficace. Cela simplifie la lecture des pages HTML et cela permet aussi de ne modifier qu'un seul fichier dans le cas où le code est amélioré !

Voici le fichier `externe.js` :

```
document.write("ce script est externe")
```

Voici le code HTML :

```
<html>
<body>
<script src="externe.js"></script>
</body>
</html>
```

En page 6, se trouve un fichier externe, appelé `msgerreur`, qui renvoie une alerte en cas d'erreur dans le code JavaScript.

2.6 Les variables en informatique

Les variables sont les éléments clés d'un langage de programmation. On conçoit une variable comme un tiroir contenant une information. La valeur de la variable peut être appelée à changer. Pour se référer à la variable, on y donne un nom.

Les noms de variables obéissent aux règles usuelles suivantes.

1. Les noms de variables doivent commencer par une lettre ou éventuellement le caractère `_`.
2. Il faut faire attention aux majuscules et aux minuscules. Par exemple `X` et `x` sont deux noms de variables différents (cette contrainte est générale à toute programmation HTML).

En langage JavaScript, on crée une variable de la même manière que l'on change sa valeur.

`var strname = valeur` ou `strname = valeur`

2.6.1 L'assignation de variables en informatique

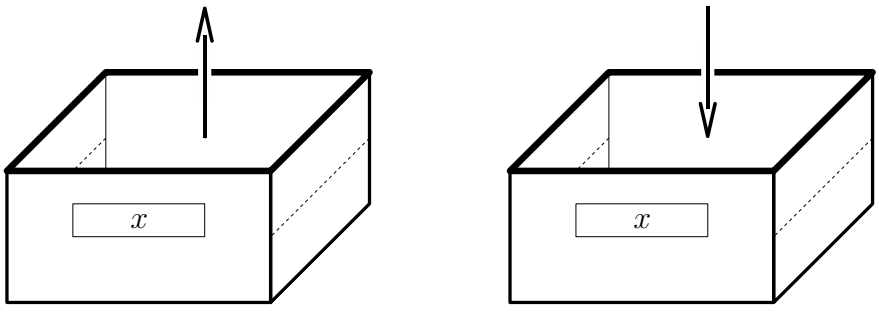
Attention, contrairement au symbole = en mathématique, dans un langage de programmation le symbole = ne doit se lire que dans un sens. Ci-dessus, il faut comprendre que le tiroir appelé `strname` contient l'objet `valeur`. Il faut bien faire la différence entre le tiroir (le nom de la variable) et son contenu (la valeur de la variable).

Par exemple si $x = 5$ (il faut penser que dans le tiroir appelé x, il y a la valeur 5), alors le code suivant va faire en sorte que $x = 6$ (il faut penser que dans le tiroir appelé x, il y a la valeur 6).

```
<html>
<head>
<script language="javascript">
  x = 5;
  x = x + 1;
</script>
</head>
```

Voici l'image mentale qu'il faut avoir :

$x = 5 \quad \Longrightarrow \quad x + 1 = 5 + 1 = 6$



Les variables déclarées dans une fonction seront détruites après que la fonction se soit exécutée (il s'agit de variables locales). Les variables déclarées en dehors d'une fonction sont conservées pour toutes les fonctions de la page web. Ces variables sont détruites lorsque la page web est fermée.

2.7 Opérations logiques sur les variables

2.7.1 Opérations arithmétiques

On retrouve les opérations mathématiques de base. L'opération modulo permet de calculer le reste de division. Par exemple, 7 modulo 2 est le reste de division de 7 par 2, il vaut 1 (car $7 = 2 * 3 + 1$)

Opération	Nom	Exemple
`+`	addition	`x = 7 + 2`
`-`	soustraction	`x = 7 - 2`
`*`	multiplication	`x = 7*2`
`/`	division	`x = 7/2`
`%`	modulo	`x = 7 % 2`
`++`	ajoute 1	`x++`
`--`	retire 1	`x--`

La puissance est dans l'extension `Math`, elle s'obtient par la commande `Math.pow` (attention à la majuscule). Voici un petit code qui affiche le résultat de 3^{30} :

```
document.write( Math.pow(3,30) );
```

D'autres commandes mathématiques se trouvent en page 8.

2.7.2 Opérations de comparaison

Une opération de comparaison renvoie la valeur de vérité (*true* pour vrai et *false* pour faux) de la proposition correspondante. L'opération de négation renverse les valeurs de vérité (vrai devient faux et vice-versa).

Opération	Nom	Exemple
`==`	est égal à	`2 == "2"`
`===`	est égal à et a le même type	`2 === "2"`
`!=`	n'est pas égal à	`2 != 2`
`>`	est plus grand que	`2 > 3`
`<`	est plus petit que	`2 < 3`
`>=`	est plus grand ou égal à	`2 >= 3`
`<=`	est plus petit ou égal à	`2 <= 3`
`&&`	ET logique	`(2 > 3) && (2 < 3)`
`||`	OU logique	`(2 > 3) || (2 < 3)`
`!`	négation (NOT)	`!(2 < 3)`

le XOR (ou exclusif) paraît manquant, mais on peut utiliser `^` qui livre 1 (au lieu de *true*) ou 0 (au lieu de *false*). On peut aussi utiliser les ET ou OU logique pour reconstruire le XOR. On peut ainsi programmer le XOR facilement.

2.7.3 Opérateur sur les chaînes de caractères

L'opérateur `+` permet aussi de concaténer deux chaînes de caractères. Par exemple `"bon"+"jour"` donne `"bonjour"`. Remarquons que l'assignation `x=21+"pommes"` mettra la chaîne de caractères `"21pommes"` dans la variable `x` (elle transforme le nombre `21` en chaîne de caractères). C'est aussi comme cela que l'on converti des nombres en chaîne de caractères (en utilisant la chaîne vide `""`).

2.8 Les fenêtres de dialogue

2.8.1 Demande d'information

Un programme requiert bien souvent une information de la part de l'utilisateur.

```
reponse = prompt("question","réponse par défaut");
document.write(reponse);
```

Les commandes parseInt et parseFloat

La réponse de la commande `prompt` est toujours une chaîne de caractère. Si on désire la convertir en nombre il faut utiliser la commande `parseInt` (pour la convertir en un nombre entier) ou `parseFloat` (pour la convertir en nombre à virgule flottante). Par exemple :

```
n = parseInt(prompt("Que vaut l'entier n ?","0"));
x = parseFloat(prompt("Que vaut la valeur de x ?","3.1416"));
```

2.8.2 Demande de confirmation

On peut aussi demander une confirmation à l'utilisateur (choix entre `ok` ou `annuler`).

```
confirmation = confirm("demande de confirmation");
document.write(confirmation);
```

2.8.3 Alertes

La commande suivante permet d'ouvrir une fenêtre d'alerte contenant un certain message, afin d'avertir l'utilisateur (pendant que la fenêtre d'alerte est affichée l'utilisateur doit d'abord cliquer sur OK avant de continuer).

```
alert("message d'alerte");
```

2.8.4 Application : comment trouver une erreur dans le code JavaScript

Voici un script externe qui peut s'avérer très utile pour traquer une majorité d'erreurs.

```
// onerror est une commande qui permet de récupérer les erreurs
onerror=messageErreur

function messageErreur(msg,url,line)
{
txt = "Il y a une erreur sur cette page :\n\n"  //la commande \n
txt = txt + "Erreur: " + msg + "\n"             //permet d'effectuer
txt = txt + "URL: " + url + "\n"                //un saut de ligne
txt = txt + "Ligne: " + line + "\n\n"           //dans la fenêtre de
                                                //dialogue
alert(txt)
return true // afin de dire à l'explorateur qu'on gère l'erreur
}
```

Il est conseillé d'enregistrer ce script dans un fichier `msgerreur.js` que l'on peut appeler par la ligne de code HTML suivante (à insérer avant tout script dans une page HTML) :

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

2.9 Les fonctions en JavaScript

Une fonction est un morceau de code qui peut être exécuté lors d'un événement ou d'un appel. Elles peuvent être définies dans l'entête (de balises `<head> ... </head>`) afin d'être utilisées dans le corps (de balises `<body> ... </body>`) de la page web.

2.9.1 Fonctions

Voici comment on implémente une fonction.

```
function mafonction(argument1,argument2,...)
  {
  lignes de code
  return sortie;  //commande optionnelle
  }
```

Une fonction sans argument doit tout de même comporter des parenthèses.

```
function mafonction()
  {
  lignes de code
  }
```

Voici une fonction qui additionne deux nombres et qui renvoie le résultat de l'addition.

```
function addition(a,b)
  {
  c=a+b;
  return c;
  }
```

Voici une fonction qui affiche un message d'alerte.

```
function danger()
  {
  alert("fonction danger exécutée");
  }
```

Pour appeler une fonction, on utilise simplement le nom de la fonction. Lorsque la fonction renvoie un nombre, on peut l'assigner dans une variable. Ici, on assigne le résultat de l'addition de 2 et de 3 dans la variable 5.

```
somme = addition(2,3);
```

Si la fonction n'a pas d'arguments, on met tout de même les parenthèses. Le script suivant ouvre la fenêtre d'alerte définie dans la fonction danger().

```
danger();
```

2.9.2 Les fonctions (et constantes) mathématiques

Dans ce cas, les variables contiennent des nombres. On peut appliquer des fonctions mathématiques à ces nombres. Les fonctions mathématiques les plus célèbres sont contenues dans l'extension `Math`.

Voici un inventaire de ces fonctions (qui renvoie toute un nombre).

`Math.abs(x)`	`Math.acos(x)`	`Math.asin(x)`	`Math.atan(x)`
`Math.ceil(x)`	`Math.cos(x)`	`Math.exp(x)`	`Math.floor(x)`
`Math.log(x)`	`Math.max(x,y)`	`Math.min(x,y)`	`Math.pow(x,y)`
`Math.random()`	`Math.round(x)`	`Math.sin(x)`	`Math.sqrt(x)`
`Math.tan(x)`			

Même si ce ne sont pas des fonctions, on trouve aussi les constantes mathématiques.

`Math.E` `Math.PI`

La commande `with (Math) { code }` permet de se passer de taper `Math.` avant chaque commande mathématique.

2.10 Les structures de contrôles

En programmation, on a souvent besoin d'effectuer différentes actions selon l'état en cours de l'exécution du programme. Les structures de contrôles sont là pour résoudre ces difficultés.

2.10.1 La commande if ... else if ... else

Cette commande se traduit par si ... si ... sinon. Seul le premier si et sa conséquence sont nécessaires. Les autres si (else if) ne sont nécessaires que si l'utilisateur en a besoin. Il en va de même pour le sinon (else).

```
if (condition1)
  {
  code qui sera exécuté si la condition1 est vraie
  }

else if (condition2)  //optionnel
  {
  code qui sera exécuté si la condition1 est vraie
  }

else  //optionnel
  {
  code qui sera exécuté si toutes
  les conditions ci-dessus sont fausses
  };
```

2.10.2 Une structure de contrôle condensée

Cette commande en une ligne peut s'avérer pratique pour l'utilisateur avancé.

```
variable=(condition)?valeur1:valeur2;
```

Si la condition est vraie, alors la *valeur1* est assignée dans la variable, sinon c'est la *valeur2* qui est assignée.

Un exemple de structure non condensée

Voici une structure de contrôle permettant de saluer l'utilisateur par un texte dynamique (dépendant de l'heure de l'ordinateur).

```
//pour trouver la date et l'heure de l'ordinateur
var d    = new Date();

//pour extraire l'heure (10h33 -> 10)
var time = d.getHours();

if (time >= 6 && time < 18)
  {
  document.write("<b>Bonjour</b>");
  }
else if (time >=18 && time < 22)
  {
  document.write("<b>Bonsoir</b>");
  }
else
  {
  document.write("<b>Bonne fin de soirée</b>");
  };
```

Un exemple de structure condensée utilisée dans une fonction

Voici un exemple de structure de contrôle condensée qui se trouve à l'intérieur d'une fonction qui permet de saluer une personne :

```
function salutations(visiteur,nom_visiteur)
  {
  texte=(visiteur=="prof")
    ? "Cher Monsieur " + nom_visiteur + ","
    : "Cher " + nom_visiteur + ",";
  return texte;
  }

document.write(salutations("prof","Perret"));
document.write("<br><br>");   //sauts à la ligne (langage HTML)
document.write(salutations("élève","Lambda"));
```

2.10.3 La commande switch

Cette commande, aussi appelée *case* dans certains langages s'utilise si on veut faire du cas à cas.

```
switch (expression)
{
case label1:
  code à exécuter si expression = label1
  break;
case label2:
  code à exécuter si expression = label1
  break;
default:
  code à exécuter si expression est différente
  de tous les labels rencontrés
}
```

Voici une structure de contrôle permettant faire une constatation à propos du jour de la semaine.

```
var d  = new Date();     //pour la date et l'heure actuelles
theDay = d.getDay();     //pour extraire le jour
                         //Dimanche = 0, lundi = 1, samedi = 6
switch(theDay)
{
case 5:
  document.write("c'est vendredi");
  break;
case 6:
  document.write("c'est un chouette samedi");
  break;
case 0:
  document.write("c'est un dimanche tranquille");
  break;
default:
  document.write("vivement ce week-end !");
};
```

2.10.4 La commande break

Cette commande doit être utilisée dans une structure de contrôle `switch`. Elle peut aussi être utilisée pour sortir d'une boucle (voir plus loin).

2.11 Les boucles

Les boucles servent à exécuter plusieurs instructions un certains nombres de fois, prédéfini ou pas.

2.11.1 La commande while

While signifie tant que. Ainsi la commande suivante exécute le code tant que la condition est vraie. Si au départ, la condition est fausse, alors le code n'est pas exécuté.

```
while (condition)
{
  code qui sera exécuté à chaque passage
}
```

Voici un bête exemple.

```
k=1;
while (k<=10)
  {document.write("hahaha ");
     k++; // équivalent à k=k+1
  };
```

2.11.2 La commande do...while

Do...while signifie fait...tant que. Ainsi la commande suivante exécute le code une fois, puis la refait tant que la tant que la condition est vraie. Donc le code va s'exécuter au moins une fois.

```
do
{
  code qui sera exécuté à chaque passage
}
while (condition);
```

2.11.3 La commande for

For signifie pour. Cette commande dépend d'une variable que l'on initialise au début de la commande. Ensuite, tant qu'une certaine condition est vraie, on exécute le code, puis la variable subit une incrémentation.

```
for (initialisation; condition; incrémentation)
{
  code qui sera exécuté à chaque passage
}
```

Dans l'exemple suivant, on prend la variable k qu'on initialise à 1, puis tant que k ¡ 10, on affiche la valeur de k, puis on incrémente la variable k de deux unités.

```
for (var k=1; k<10; k=k+2)
  {
  document.write(k);
  document.write("<br>");
  };
```

2.12 Les différents types de données

Les données que l'on peut stocker dans des variables sont de types multiples.

2.12.1 Les booleans

Un boolean est une variable contenant une valeur de vérité, il n'y a que deux valeurs de vérité : vrai ou faux. Les booleans sont principalement utilisés dans les structures de contrôles. On peut changer la valeur de vérité en utilisant l'opérateur `!` (voir page 5).

```
var b1 = new Boolean(true);

document.write("valeur de b1 : ",b1);
document.write("<br>");
document.write("contraire de b1 : ",!b1);
```

2.12.2 Les tableaux

Un tableau est un type de donnée qui peut contenir d'autres types de données. Lors de la création, la taille du tableau doit être indiquée. En JavaScript, les tableaux sont en fait des listes (tableaux à une ligne), par contre les types des éléments des tableaux peuvent être mélangés (chaînes de caractères, nombres, etc.)

Voici un tableau de taille 3. Si rien n'est indiqué, les 3 éléments sont vides (initialisation automatique).

```
var prenoms = new Array(3);    // attention à la majuscule
```

Voici un tableau de taille 3 dont les éléments sont connus.

```
var prenoms = new Array("Steve","Cindy","John");
```

Pour faire appel à un des éléments du tableau, par exemple le deuxième, on utilise des crochets. Attention, pour un tableau à 3 éléments, le premier élément porte l'indice 0 et le dernier 2.

```
document.write(prenoms[1]);   // prenoms[0] correspond à Steve
```

Pour trouver la longueur d'un tableau, on accole la commande `.length` au nom du tableau.

```
prenoms.length;
```

On peut afficher d'un coup le contenu d'un tableau. Par défaut, JavaScript sépare ses éléments par des virgules. Si on désire avoir un autre séparateur, on peut accoler la commande `.join(séparateur)`.

```
document.write(prenoms.join("; "));
```

Deux méthodes pour trier automatiquement des tableaux

On peut trier les tableaux à l'aide de la commande `.sort`. Deux méthodes de tri sont possible : le tri lexicographique et le tri numérique. Le tri lexicographique trie dans l'ordre alphabétique en commençant par le premier caractère, tandis que le tri numérique (qui ne trie que les nombres) ordonne les nombres du plus petit au plus grand. La commande `.sort` effectue le trie lexicographique par défaut.

Comme cette commande est très générale, elle permet bien d'autres possibilités, mais il faut programmer ces méthodes. Voici un programme qui permet de faire un tri numérique.

```
// Cette fonction sera utilisée pour le tri numérique
function compareNum(a,b) { return a-b; };

// tableau de nombres
tableau = new Array("1","2","11","70","9","800");

// tri lexicographique
document.write("Tri lexicographique : " + tableau.sort());
document.write("<br>");

// tri numérique (a l'aide de la fonction compareNum)
// remarquer qu'en argument de la commande sort on indique
// le nom de la fonction
document.write("Tri numérique : " + tableau.sort(compareNum));
```

Ajouts et suppressions d'éléments d'un tableau

On peut ajouter un ou plusieurs éléments à la fin d'un tableau en accolant la commande `.push`. Attention, pour cette commande on ne doit pas réassigner le tableau (`prenoms=prenoms.push("Toto")` ne fonctionne pas !). En effet, la commande `.push` est une fonction qui renvoie le nouvelle longueur du tableau.

```
prenoms.push("Toto");
document.write(prenoms.join());
```

La commande `.splice` est très puissante, elle permet d'ajouter autant d'entrées que l'on veut, là où le veut. On peut simultanément supprimer des entrées au même endroit. Cette fonction renvoie un tableau de taille 0 si on n'a rien supprimé ou un tableau contenant les éléments enlevés (la taille de ce tableau est égale au nombre d'éléments enlevés).

```
nomTableau.splice(position de l'insertion/suppression,
                  nombre d'élément supprimes,
                  liste d'éléments a ajouter);
```

Voici un exemple complet utilisant la commande `.splice` :

```
// creation du tableau
var prenoms = new Array("Steve","Cindy","John");

// on affiche la liste prenoms avant de la modifier
document.write("liste avant modification : ",prenoms,"<br>");

// insertion des prenoms "Jack" et "Britney" apres "Steve"
// (position 1), sans suppressions de prénoms
prenomsEnleves1 = prenoms.splice(1,0,"Jack","Britney");

// on affiche la liste prenoms pour verifier
document.write("nouvelle liste : ",prenoms,"<br>");

// on affiche la liste des éléments supprimés (ici aucun)
document.write("liste enlevée : ",prenomsEnleves1,"<br>");

// insertion du prénom "Christina" après "Jack"
// (et avant "Britney") (position 2), avec
// suppression de "Britney" (1 entree)
prenomsEnleves2 = prenoms.splice(2,1,"Christina");

// on affiche la liste prénoms pour vérifier
document.write("nouvelle liste : ",prenoms,"<br>");

// on affiche la liste des éléments supprimés (ici "Britney")
document.write("liste enlevée : ",prenomsEnleves2,"<br>");
```

Il y a bien d'autres commandes pour gérer les tableaux, mais nous ne les verrons pas !

2.12.3 Les dates

On peut définir une date grâce à la commande `Date`. Attention, tous les paramètres sont des nombres. le nombre 0 correspond au mois de janvier ou au dimanche. Ce qui est entre crochet ci-dessous est optionnel.

```
jour = new Date(annee,mois,jour[,heures,min,sec,millisec]);
```

On peut très facilement extraire la date de l'horloge interne de l'ordinateur avec la commande suivante.

```
maintenant = new Date();
```

Voici un programme astucieux qui permet d'afficher la date de manière conventionnelle en séparant les informations composant la date.

```
var d=new Date()
var weekday=new Array("dimanche","lundi","mardi","mercredi",
                      "jeudi","vendredi","samedi");
var monthname=new Array("jan","feb","mars","avril","mai",
                        "juin","juillet","août","sep",
                        "oct","nov","dec");
document.write("Nous sommes le ");
document.write(weekday[d.getDay()] + " ");
document.write(d.getDate() + " ");
document.write(monthname[d.getMonth()] + " ");
document.write(d.getFullYear() + " à ");
document.write(d.getHours() + "h");
document.write(d.getMinutes() + " et ");
document.write(d.getSeconds() + " secondes !");
```

2.12.4 Les chaînes de caractères

Une chaîne de caractères commence toujours par " et se termine par ". Les caractères spéciaux (tels que les guillemets) s'ajoute grâce à la commande \. On peut concaténer deux chaînes de texte à l'aide de la commande +.

Il y plein d'autres opérations que l'on peut faire subir à une chaîne de caractères.

On peut obtenir sa longueur.

```
chaine="Youpie";
chaine.length;
```

On peut convertir une chaîne de caractère en minuscules.

```
chaine="Youpie";

//convertion en majuscules avec toUpperCase()
document.write(chaine.toLowerCase());
```

La commande ci-dessous permet de tester si une chaîne de caractères contient un morceau de texte spécifique et retourne la position du premier caractère du morceau cherché. On obtient 0 si le morceau commence en première position. Si le morceau n'apparaît nul part, alors cette commande renvoie -1.

```
chaine="Youpie";

document.write(chaine.indexOf("pie"));
// renvoie 3 (car p est en quatrième position)
```

On peut fracturer une chaîne de caractère en un tableau de caractères ce qui permet de faire des effets. En argument de la commande `split` on met la chaîne de caractère faisant office de séparateur. Une chaîne vide `""` permet de séparé chaque caractère.

```
chaine="de plus en plus grand";
tableau=chaine.split(" ");
for (var k=0;k<tableau.length;k++)
 {
  document.write(tableau[k].fontsize(k+3)," ");
 };
```

On peut afficher du texte en couleur.

```
chaine="Youpie";
document.write(chaine.fontcolor("red"));
```

On peut afficher du texte dans la taille voulue (`taille` va de 1 à 7).

```
chaine="Youpie";
document.write(chaine.fontsize(taille));
```

Il y a plein d'autres options pour l'affichage du texte, mais la plupart sont déjà disponibles avec du code HTML.

3 Interactivité en JavaScript

3.1 Les boutons

Ce programme HTML permet d'assigner une fonction JavaScript à un bouton.

```
<html>
<head>
<script language="javascript">
function reaction()
 { alert("vous venez de presser le bouton"); };
</script>
</head>
<body>
<input type="button"
       value="Presse-moi !"
       onMouseOver="window.status='explications';return true"
       onMouseOut="window.status='';return true"
       onclick="reaction()">
</body>
</html>
```

Même si les liens HTML fonctionnent, le bouton de la page web ci-dessous permet d'ouvrir une page Internet. Par ailleurs, la fonction `open_win()` pourrait faire bien d'autres choses !

```
<html>
<head>
<script language="javascript">
function open_win()
 {
 window.open("http://www.jura.ch/lcp/");
 };
</script>
</head>

<body>
<input type=button value="Site du lycée" onclick="open_win()">
</body>
</html>
```

3.2 Les effets sur les images

On peut changer d'image lorsque la souris passe sur une image où lorsqu'on clique sur une image. Pour cela JavaScript utilise la variable `document.images` qui n'est pas implémentée sur les vieux navigateurs (voir annexes page 22).

Voici l'entête du programme (on gère un tableau qui contient des images) :

```
<html>
<head>
<script language="javascript">

// mémorise les images affichées si la souris est dessus
var onImgArray = new Array()
onImgArray["image1"] = new Image
onImgArray["image1"].src = "gif05.gif"
onImgArray["image2"] = new Image
onImgArray["image2"].src = "gif06.gif"

// mémorise les images affichées si la souris n'est plus dessus
var offImgArray = new Array()
offImgArray["image1"] = new Image
offImgArray["image1"].src = "gif03.gif"
offImgArray["image2"] = new Image
offImgArray["image2"].src = "gif04.gif"

// fonctions qui change les images
function imageOn(imgName)
{ document.images[imgName].src = onImgArray[imgName].src }
function imageOff(imgName)
{ document.images[imgName].src = offImgArray[imgName].src }

// fonction pour régler le message dans la barre de statut
function setMsg(msg) { window.status = msg; return true; }

// fonctions activées en cliquant sur les images
function action1()
 { document.images["image2"].height = 500
   document.images["image2"].src = "gif09.gif" }
function action2() {  }

</script>
</head>
```

...Ce script continue sur la page suivante...

Voici le corps du programme (on y trouve l'utilisation de code `HTML` et de la variable `document.images`). La commande `name` dans la balise `<img> ... </img>` est capitale !

```
<body>
<center>
<A HREF="javascript:action1()"
   onMouseOver="imageOn('image1'); return setMsg('on image1')"
   onMouseOut="imageOff('image1'); return setMsg('off image1')">
  <img src="gif01.gif" name="image1">
</A>

<A HREF="javascript:action2()"
   onMouseOver="imageOn('image2'); return setMsg('on image2')"
   onMouseOut="imageOff('image2'); return setMsg('off image2')">
  <img src="gif02.gif" name="image2" height=202 width=202>
</A>
</center>
</body>
</html>
```

On peut remarquer que si la taille de l'image (`height` (pour la hauteur) et `width` (pour la largeur)) n'est pas précisée, alors c'est la taille de l'image qui l'emporte, ce qui va changer la mise en page de la page web : cela peut être un effet désiré par le créateur de la page.

On remarque l'emploi des balises `HTML` qui permettent de créer des hyperliens (balise `<A> ... </A>`) et d'insérer des images (balise `<IMG> ... </IMG>`). Il est important que le nom de l'image (`name="imageX"`) corresponde à l'indice passé en paramètre du tableau `document.images` (on remarque par ailleurs qu'il est possible que les indices ne soient pas des nombres, mais des chaînes de caractères).

Les commandes `onMouseOver` et `onMouseOut` servent à décrire l'action à effectuer lorsque le curseur de la souris se trouve respectivement *sur* ou *en dehors* de l'hyperlien (ou de l'image).

On peut aussi changer la taille d'une image (pourrez-vous expliquer l'effet étrange ?).

```
<html>
<head>
<script language="javascript">
function setHeight(ImgName,a,b)
{ document.images[ImgName].height=a
  document.images[ImgName].width=b  }
</script>
</head>

<body>
<img src="gif01.gif" name="image1" width="202" height="202"
     onMouseOver="setHeight('image1',202,101)"
     onMouseOut ="setHeight('image1',101,202)">
</body>
</html>
```

3.3 Les formulaires

Créer un formulaire demande une certaine connaissance du langage HTML. Cela sort du cadre de ce cours. Toutefois, le lecteur intéressé peut se rendre sur le site de Didier Müller.

`http://www.apprendre-en-ligne.net/javascript/index.html`

Les leçons 5 et 8 concernent la gestion de formulaire. La leçon 7 explique comment ouvrir une nouvelle fenêtre à l'aide de JavaScript.

Le formulaire qui se trouve à la leçon 8 possède une partie de code en JavaScript,

`http://www.apprendre-en-ligne.net/javascript/lecon8/autoeval2.html`

mais aussi une partie de code en HTML.

`http://www.apprendre-en-ligne.net/javascript/lecon8/autoeval3.html`

3.3.1 Un exemple de formulaire sans JavaScript

Même si la page web n'est pas hébergée sur un serveur, on peut recevoir les réponses sur son email (sans utiliser JavaScript).

Au verso de cette page se trouve un prototype de formulaire n'utilisant pas JavaScript, mais permettant d'obtenir un feed-back de la part d'un utilisateur.

Sur la page suivante, on trouve un tel exemple de code HTML pur (c'est-à-dire sans JavaScript).

```
<HTML>
<HEAD>
<TITLE>Évaluation du cours de Mathématiques</TITLE>
</HEAD>

<BODY bgcolor="#DDAAFF">

<CENTER>
<H1>Évaluation du cours de Mathématiques</H1>

Ce questionnaire permettra vous permettre de signaler vos besoins en cochant les cases
correspondantes. Profiter de l'espace consacré aux remarques et commentaires pour
donner votre opinion personnelle.

</CENTER>

<form action="mailto:professeur@ecole.ju?subject=Formulaire" method="POST" enctype="text/plain">

<HR>

<H3>Concernant les objectifs du cours de Mathématiques</H3>

<TABLE cols=2 cellpadding=5>

<TR> <TD width=250> <H3>
1. Les principes de l'algèbre
</H3> </TD> </TR>

<TR> <TD> <H4>
1.1 Notions de base
</H4> </TD> </TR>

<TR>
<TD>
- Les ensembles de nombres
</TD>
<TD>
<TABLE border cols=2 align=center>
<TR align=center>
<TD width=200> Besoin d'explications </TD>
<TD width=200> Besoin d'exercices    </TD>
</TR>
<TR align=center>
<TD> <input name="Les ensembles de nombres" type=Checkbox value=" besoin d'explications"> </TD>
<TD> <input name="Les ensembles de nombres" type=Checkbox value=" besoin d'exercices"> </TD>
</TR>
</TABLE>
</TD>
</TR>

</TABLE>

<HR>

<H3> Remarques et commentaires </H3>

<TEXTAREA name="Commentaires" rows=10 cols=74 wrap=Physical></TEXTAREA>
<BR> <BR>

<HR>

<BR>
<CENTER>
<INPUT type=Submit value="Envoi du formulaire">
<INPUT type=Reset value="Tout recommencer">
</CENTER>

</FORM>

</BODY>
</HTML>
```

4 Annexes et références

Problèmes possibles avec les vieux navigateurs internet

Si le code JavaScript s'affiche à l'écran

Pour les plus vieux navigateurs, il a un risque que le code JavaScript soit affiché à l'écran, il faut donc indiquer au langage HTML qu'il s'agit d'un commentaire.

```
<script language="javascript">
<!--
  lignes de code
//-->
</script>
```

Remarquons au passage que les `//` signalent un commentaire pour JavaScript. Malgré tout la première ligne ne s'écrit pas `//<!--`, c'est bizarre, mais c'est comme ça.

Si la commande document.images n'existe pas

Dans ce cas, il faut utiliser la distinction suivante :

```
<script language="javascript">

if (document.images)
{
 code qui sera exécuté si tout va bien
}
else
{
 code qui sera exécuté si document.images n'est pas reconnu
}

</script>
```

Si `document.images` n'existe pas, la condition `(document.images)` renvoie *undefined* ce qui est traité par un `if` comme *false*.

Références

1. Livre «JavaScript : le guide du développeur» de chez Osman Eyrolles Multimedia.
2. Cours web très complet (en anglais) qui se trouve sur le lien suivant :
   `http://www.w3schools.com/js/js_intro.asp`
3. Livre «Guide de l'utilisateur HTML» de chez Micro Application.