

JAVASCRIPT



JavaScript

Eric WILLEMS – Luc Granier
2022-2023

Sommaire

| | |
|--|----|
| I Introduction au JavaScript | 3 |
| II Les variables et les opérateurs..... | 6 |
| III Les structures de contrôle..... | 16 |
| L'instruction if (SI) | 17 |
| L'instruction if ... else (SI ALORS ... SINON) | 19 |
| L'instruction for (POUR) | 22 |
| L'instruction while (TANT QUE)..... | 25 |
| L'instruction switch (SELON) | 28 |
| L'instruction do ... while (Répéter tant que) | 29 |
| IV Les Fonctions..... | 31 |
| V Les événements | 38 |
| VI Le DOM (Document Object Model) | 41 |
| VII Comment manipuler les tableaux ? | 45 |
| Exemples de tableaux associatifs :..... | 51 |
| Les tableaux multidimensionnels..... | 52 |
| VIII Comment manipuler les dates ? | 55 |
| IX Comment manipuler les chaînes de caractères ? | 60 |
| X Regex pour valider des formulaires HTML..... | 65 |

I Introduction au JavaScript

Dynamisez votre site Web avec Javascript

Le dynamisme coté client consiste à effectuer des traitements en local sans solliciter le serveur pour notamment :

- affiner l'ergonomie de l'interface utilisateur et augmenter la réactivité
- diminuer la charge de travail du serveur au profit de celle du client
- limiter la communication entre le client et le serveur
- préparer les informations transmises vers le serveur

JavaScript permet de gérer l'interactivité

Exemple : gérer un événement provoqué par la souris

```
<!-- Avec_Javascript.html -->
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8"/>
    <title>Hello World!</title>
  </head>

  <body>

    <script >
      alert("Bonjour!"); // Commentaire ou /* */
      var date = new Date();
      document.write("Nous sommes le " + date);
    </script>

  </body>
</html>
```

Le langage JavaScript est adopté par l'ensemble des navigateurs. Ce langage orienté objet est exécuté (interprété) côté client.

Un script est une portion de code qu'on insère dans une page HTML entre les balises <script> et </script>

alert('Hello world!'); produit le même effet que window.alert('Hello world!');

alert() est une méthode de l'objet window.

Toutes les fonctions ne font pas nécessairement partie de l'objet window. Ainsi, les fonctions comme isNaN(), parseInt() ou encore parseFloat() ne dépendent pas d'un objet.

Ce sont des *fonctions globales*. Ces dernières sont, cependant, extrêmement rares.

Il est possible d'écrire du code en langage JavaScript dans un fichier .js (*méthode la plus recommandée*) ;

Il suffit de rajouter dans le fichier .html : `<script src="js/module.js"> </script>`
en ayant pris soin, dans ce cas, de mettre le fichier module.js dans le dossier js

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8"/>
    <title>Hello World!</title>
    <script src="js/module.js"> </script>
  </head>

  <body>

  </body>
</html>
```

```
// module.js

alert('Bonjour!');
var date=new Date();
document.write ("Nous sommes le " + date);
```

Pour lire une information au clavier, il suffit d'utiliser la fonction `prompt ()`:

Exemple : `var x;`
 `x= prompt ("Saisir une valeur", 0);`
 `x = parseInt (x); // pour obtenir une valeur numérique`

Outils de débogage

Avec Google Chrome : Menu/Plus d'outils/Outils de développement/Sources

Avec Firefox : Menu/Développement Web/ Débogueur

Gestion d'un événement : Ouverture d'une boîte de dialogue lors d'un clic sur un lien.

```
<!-- evenement.html -->

<html lang="fr">
<head>
  <title>Ouverture d'une boîte de dialogue lors d'un clic</title>
</head>

<body>
  <a href = "" onClick="window.alert('Message d\'alerte') ;">Cliquez ici!</a>
</body>
</html>
```



Gestion d'un événement : Appel d'une fonction lors d'un clic sur un bouton

```
<!-- evenement_bouton.html -->

<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>Appel d'une fonction lors d'un click sur un bouton </title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <script src="exo.js"> </script>
  </head>
  <body>
    <input type="button" value="exo1" onClick = "exercice1();"><br>
    <input type="button" value="exo2" onClick = "exercice2();"><br>
    <input type="button" value="bouton" onClick="alert('Evènement onclick déclenché');"/><br>
  </body>
</html>
```

```
// exo.js
function exercice1()
{
  alert("Bonjour!");
  var date=new Date();
  document.write ("Nous sommes le
",date);
}
function exercice2()
{
  alert('Deuxième exercice!');
}
```



II Les variables et les opérateurs

Objectifs

- Manipuler les divers opérateurs du langage JavaScript

Outils et supports nécessaires

- Eclipse ou Netbeans
- Firefox ou Google Chrome

Notions sur les variables

Déclaration de 2 variables : valeur1 et valeur2 :

```
var valeur1;  
var valeur 2;
```

Initialisation à une valeur spécifique

```
valeur 1 = 23;  
alert (typeof(valeur1)); // affiche number  
valeur 1 = "Bonjour"; // équivalent à valeur 1 = 'Bonjour';  
alert (typeof(valeur1)); // affiche string  
valeur 1 = true;  
alert (typeof(valeur1)); // affiche boolean  
  
alert (typeof(valeur2)); // affiche undefined  
  
alert ("\n"); // saut à la ligne suivante
```

Le signe "+" a 2 rôles distincts : addition ou concaténation

```
var x;  
var y;  
x="123";  
y="456";  
document.write "<br>"); // saut à la ligne suivante  
document.write ("x+y = " + x+y + "<br>"); // affiche x+y = 123456  
document.write ("x+y = " , x , y + "<br>"); // affiche x+y = 123456  
x=parseInt (x); // parseFloat() converti en valeur flottante (réelle)  
y=parseInt (y);  
document.write ("x+y = " , x+y + "<br>"); // affiche x+y = 579
```

En résumé, JavaScript gère 4 types de variables :

| | |
|----------------------|---|
| entier | 123 (base 10) 0123 (base 8) 0x12A (base 16) |
| flottant | 0.123 -0.23 63E-17 |
| booléen | true false |
| chaîne de caractères | "texte" ou 'texte' |

Portée des variables :

locale (uniquement dans le script ou la fonction)
var x;

globale
y=4;

Les boîtes de dialogue

Fonction pour lire une valeur saisie au clavier :

`prompt ()` : retourne une chaîne de caractères (string) acquise au clavier.

Exemple : `var x ;`
 `x = prompt ("Saisir un nombre : " , 0); // 0 est la valeur par défaut`
 `x=parseInt (x); // pour obtenir une valeur numérique`

Fonctions pour écrire ou afficher :

`alert ()` : permet d'écrire un message dans une fenêtre.

```
var resultat = 10 ;  
alert (" Le résultat vaut : " + resultat + "\n");
```

`confirm ()` permet de confirmer un choix entre **OK** et **Annuler**.

Cette méthode retourne la valeur **true** si l'utilisateur appuie sur OK, et retourne **false** dans le cas contraire.

Fonctions pour écrire ou afficher dans une page HTML :

`document.write ()` : permet d'écrire un message dans une page HTML

```
var message = " Bonjour " + " &nbsp; " + " à tous " + "<br>" ;  
document.write (message);
```


Les opérateurs de calcul

| Opérateur | Dénomination | Effet | Exemple | Résultat (avec x valant 7) |
|-----------|-----------------------------|--|----------|------------------------------------|
| + | Opérateur d'addition | Ajoute deux valeurs | $x+3$ | 10 |
| - | Opérateur de soustraction | Soustrait deux valeurs | $x-3$ | 4 |
| * | Opérateur de multiplication | Multiplie deux valeurs | $x*3$ | 21 |
| / | Opérateur de division | Divise deux valeurs | $x/3$ | 2.3333333 |
| = | Opérateur d'affectation | Affecte une valeur à une variable | $x = 3$ | Met la valeur 3 dans la variable x |
| % | Opérateur modulo | Retourne le reste de la division entière de l'opérande de gauche par celui de droite | $x \% 2$ | 1 |

Deux manières d'obtenir le même résultat :

```
<!-- TP Operateurs EXO1.html -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Page d'exercices en JavaScript</title>
  </head>
  <body>    <!-- Corps du document -->
    <script>
      var n = 12;
      var p = 15;
      var resultat;
      resultat = n + p;
      window.alert("Le resultat vaut : " + resultat + "\n");
      document.write("Le resultat vaut : " + resultat + "<br>");
    </script>
  </body>
</html>
```

```
<!-- TP Operateurs EXO2.html -->
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Page d'exercices en JavaScript</title>
  </head>
  <body>    <!-- Corps du document -->
    <script src="/js/exo.js"></script> <!-- Si le fichier se trouve dans le dossier js -->
    <script src="../js/exo.js"></script> <!-- permet de remonter dans l'arborescence -->
  </body>
</html>
```

```
// exo.js

var n = 12;
var p = 15;
var resultat;
resultat = n + p;
window.alert("Le resultat vaut : "+resultat);
```



Les opérateurs de comparaison

Pour tous ces exemples, supposez que w vaut 9, x 10, y 10 et z 20.

| Opérateur | Opération | Exemple de code | Résultat |
|-----------|-------------------------------------|-----------------|----------|
| < | Inférieur à | w < z | true |
| > | Supérieur à | w > z | false |
| <= | Inférieur ou égal | w <= z | true |
| >= | Supérieur ou égal à | z >= x | true |
| == | Egalité | x == y | true |
| != | Inégalité (non égal) | x != y | false |
| === | Contenu et type égal à | x === y | true |
| !== | Contenu ou type différent de | x !== y | false |

Les opérateurs logiques (&& ; || ; !)

| Opérateur | Opération | Exemple de code | Résultat bool3 |
|-----------|------------------------|-------------------------|----------------|
| && | AND logique | bool3 = bool1 && bool2; | false |
| | OR logique | bool3 = bool1 bool2; | true |
| ! | Négation logique (not) | bool3 = !bool1; | false |

Pour tous les exemples, supposez la valeur **true** pour bool1 et la valeur **false** pour bool2.

Les opérateurs bit-à-bit (& ; | ; ^).

Les opérateurs suivants effectuent des opérations bit-à-bit, c'est-à-dire avec des bits de même poids.

| Opérateur | Opération | Exemple de code | Résultat |
|-----------|------------------------|----------------------|-----------|
| & | ET bit à bit | 9 & 12 (1001 & 1100) | 8 (1000) |
| | OU bit à bit | 9 12 (1001 1100) | 13 (1101) |
| ^ | OU exclusive bit à bit | 9 ^ 12 (1001 ^ 1100) | 5 (0101) |

Les opérateurs d'affectation

Ces opérateurs permettent de simplifier des opérations telles que *ajouter une valeur dans une variable et stocker le résultat dans la variable*. Une telle opération s'écrirait habituellement de la façon suivante par exemple: $x = x + 2$

Avec les opérateurs d'assignation il est possible d'écrire cette opération sous la forme suivante: $x += 2$

Ainsi, si la valeur de x était 7 avant opération, elle sera de 9 après...

| Opérateur | Opération | Exemple de code | Résultat |
|-----------|--|-----------------|----------|
| += | ajoute l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche | | |
| -= | soustrait l'opérande de droite à l'opérande de gauche et stocke le résultat dans l'opérande de gauche | | |
| *= | multiplie l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche | | |
| /= | divise l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche | | |
| %= | calcule le reste de la division entière de l'opérande de gauche par l'opérande de droite et stocke le résultat dans l'opérande de gauche | | |

Les opérateurs d'incrément

Ce type d'opérateur permet de facilement augmenter ou diminuer d'une unité une variable. Ces opérateurs sont très utiles pour des structures telles que des boucles, qui ont besoin d'un compteur (variable qui augmente de un en un).

Un opérateur de type $x++$ permet de remplacer des notations lourdes telles que $x = x + 1$ ou bien $x += 1$

| Opérateur | Opération | Exemple de code | Résultat (avec x valant 7) |
|-----------|--|-----------------|----------------------------|
| ++ | Incrément (Augmente d'une unité la variable) | $x++$; | 8 |
| -- | Décrément (Diminue d'une unité la variable) | | |

Les opérateurs de rotation de bit

Ce type d'opérateur traite ses opérandes comme des données binaires d'une longueur de 32 bits, plutôt que des données décimales, hexadécimales ou octales. Ces opérateurs traitent ces données selon leur représentation binaire mais retournent des valeurs numériques standards dans leur format d'origine.

Les opérateurs suivants effectuent des rotations sur les bits, c'est-à-dire qu'il décale chacun des bits d'un nombre de bits vers la gauche ou vers la droite. Le premier opérande désigne la donnée sur laquelle on va faire le décalage, la seconde désigne le nombre de bits duquel elle va être décalée.

| Opérateur | Opération | Exemple de code | Résultat (avec x valant 7) |
|-----------|---|----------------------|----------------------------------|
| << | Rotation à gauche. Décale les bits vers la gauche (multiplie par 2 à chaque décalage). Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite | 6 << 1 (0110 << 1) | 12 (1100) |
| >> | Rotation à droite avec conservation du signe. Décale les bits vers la droite (divise par 2 à chaque décalage). Les bits qui sortent à droite sont perdus, tandis que le bit non-nul de poids plus fort est recopié à gauche. Le signe est conservé. | 6 >> 1 (0110 >> 1) | 3 (0011) |
| >>> | Rotation à droite avec remplissage de zéros Décale les bits vers la droite (divise par 2 à chaque décalage). Les bits qui sortent à droite sont perdus, tandis que des zéros sont insérés à gauche. Perte du signe négatif. | 6 >>> 1 (0110 >>> 1) | 3 (0011) |

Les opérateurs de manipulation de chaînes de caractères

L'opérateur '+' lorsqu'il est utilisé avec des chaînes de caractères permet de les concaténer, c'est-à-dire de joindre bout-à-bout les deux chaînes de caractères :

Ainsi `var='a'+'b'` est équivalent à `var='ab'`.

```
var1='a'  
var=var1+'b' -> var='ab'
```

Les priorités des opérateurs

Lorsque l'on associe plusieurs opérateurs, il faut que le navigateur sache dans quel ordre les traiter, voici donc dans l'ordre décroissant les priorités de tous les opérateurs :

() []

-- ++ ! ~ -

* / %

+ -

< <= >= >

== !=

&

^

|

&& ||

' :

= += -= *= /= %= <<= >>= >>>= &= ^= |=

,

Exercice 1 : Ecrire un programme en Javascript qui permet de saisir 3 nombres au clavier, de calculer la somme, le produit et la moyenne, puis d'afficher les résultats dans un page HTML.

Dans un premier temps, votre code javascript sera placé entre les balises `<script>` et `</script>` dans un fichier .html

Puis dans un second temps, vous créerez un fichier .js à l'extérieur de votre fichier .html

Exercice 2 : Complétez les tableaux précédents en réalisant divers exemples pour chacun des opérateurs du langage JavaScript

III Les structures de contrôle

Objectifs

- Concevoir et développer une application
- Passer de l'algorithme à la programmation

Outils et supports nécessaires

- Eclipse ou Netbeans
- Firefox ou Google Chrome

Outils de débogage

Avec Google Chrome : Menu/Plus d'outils/Outils de développement/Sources

Avec Firefox : Menu/Développement Web/ Débogueur

Trois structures de contrôle :

Séquentielle

Conditionnelle (trois variantes)

if ... else ...
if ...
switch ... case ...

Iterative (trois variantes)

while ...
do ... while
for

Les opérateurs de comparaison

| Opérateur | Opération | Exemple de code | Résultat |
|-----------|----------------------|-----------------|----------|
| < | Inférieur à | w < z | true |
| > | Supérieur à | w > z | false |
| <= | Inférieur ou égal | w <= z | true |
| >= | Supérieur ou égal à | z >= x | true |
| == | Egalité | x == y | true |
| != | Inégalité (non égal) | x != y | false |

Pour tous ces exemples, supposez que w vaut 9, x 10, y 10 et z 20.

Les opérateurs logiques

| Opérateur | Opération | Exemple de code | Résultat bool3 |
|-----------|------------------------|-------------------------|----------------|
| && | AND logique | bool3 = bool1 && bool2; | false |
| | OR logique | bool3 = bool1 bool2; | true |
| ^ | OR exclusif | bool3 = bool1 ^ bool2; | true |
| ! | Négation logique (not) | bool3 = !bool1; | false |

Pour tous les exemples, supposez la valeur **true** pour bool1 et la valeur **false** pour bool2.

Les blocs de code

Pour rester simple, disons qu'un bloc de code peut être :

- Une ligne de code.
- Un ensemble de lignes de code placé entre accolades ({ }). Pour l'interpréteur JavaScript, tout le code entre accolades est équivalent à une ligne de code. Il peut même n'y avoir aucun code entre deux accolades.

Voici un exemple de bloc de code :

```
x = x + y; //ligne de code
```

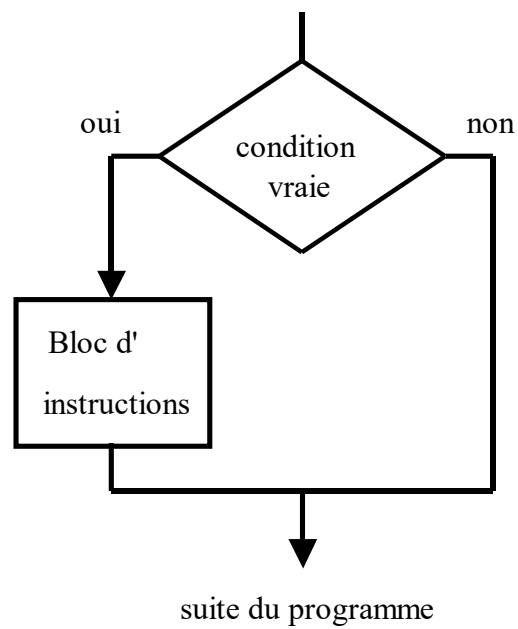
Et en voici un autre :

```
{ // ensemble de ligne entre accolades
  y = 10;
  x = 20;
  window.alert("Bonjour !");
} // fin de bloc
```

Les instructions if et if . . . else (SI et SI ALORS ... SINON)

L'instruction if (SI)

```
if (expression conditionnelle vraie)
{
    < instruction>;
    < instruction>;
}
```



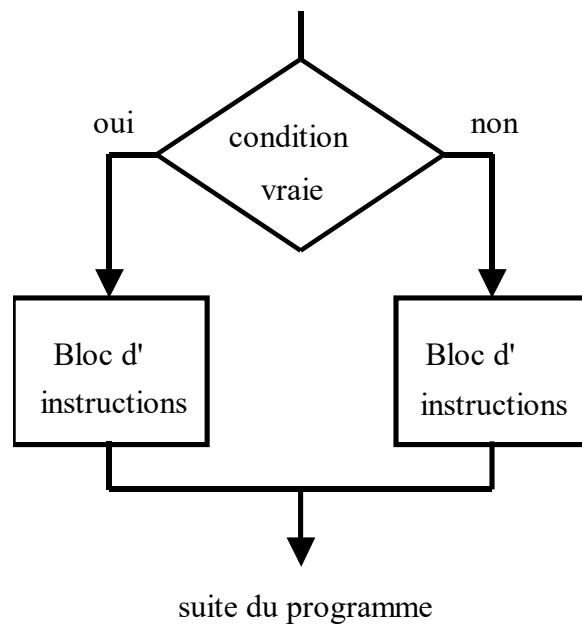
Exemple :

```
var pluie = true ;  
if (pluie==true) // identique à if (pluie)  
{  
    window.alert("Prends le parapluie !") ;  
}
```

Attention : *if (pluie==true) ; // le point virgule empêche l'exécution du if*

L'instruction if ... else (SI ALORS ... SINON)

```
if (expression conditionnelle vraie)
{
    < instruction>;< instruction>;
}
else
{
    < instruction>;< instruction>;
}
```



Exemple :

```
var pluie = true;
if (pluie == true) // identique à if (pluie)
{
    window.alert("Prends le parapluie !");
} else
{
    window.alert("Ne prends pas le parapluie !");
}
```

Exercice 1 :

Ecrire un programme qui affiche dans une boîte d'alerte la **somme** ou le **produit** de 2 nombres. Ce programme doit permettre de saisir les deux nombres ainsi que la **lettre** représentant l'**opération** à effectuer (la saisie des **2 nombres** et de l'**opération** se fait au clavier (grâce la méthode **window.prompt()**).

Exemple : var nb; nb=window.prompt ("Saisir un nombre",0); // 0 par défaut

Les valeurs saisies au clavier sont des chaines de caractères. Il suffit d'utiliser parseInt pour obtenir des entiers : Exemple :
nb = parseInt (nb);

Si l'opération vaut « **s** » (comme somme), il calcule et affiche la somme, et si l'opération vaut « **p** » (ou tout autre caractère), le programme doit calculer et afficher le produit.

Exercice 2 :

Ecrire un programme qui permet d'afficher un message de bienvenue (« Bonjour Madame ! ») ou (« Bonjour Monsieur ! »), en fonction du **genre** (« **F** » pour féminin, « **M** » pour masculin). Le **genre** sera saisi avec la méthode **window.prompt()**.

Exercice 3 :

Ecrire un programme qui permet la saisie d'un nombre et affiche si ce nombre est compris dans les intervalles 5-10 ou 15-20. (Condition composée)

Exercice 4 :

Ecrire un programme qui affiche le montant à payer d'une facture.

On dispose d'un montant déjà calculé et on veut effectuer une remise de 1% si le montant de la facture dépasse strictement 150 €

Exercice 5 :

Ecrire un programme qui permet le calcul de l'âge approximatif d'un individu à partir de l'année de naissance saisie au clavier.

Exercice 6 :

Quelle différence faites-vous entre l'opérateur = et l'opérateur == ? Ecrivez un script permettant à l'utilisateur de faire cette différence.

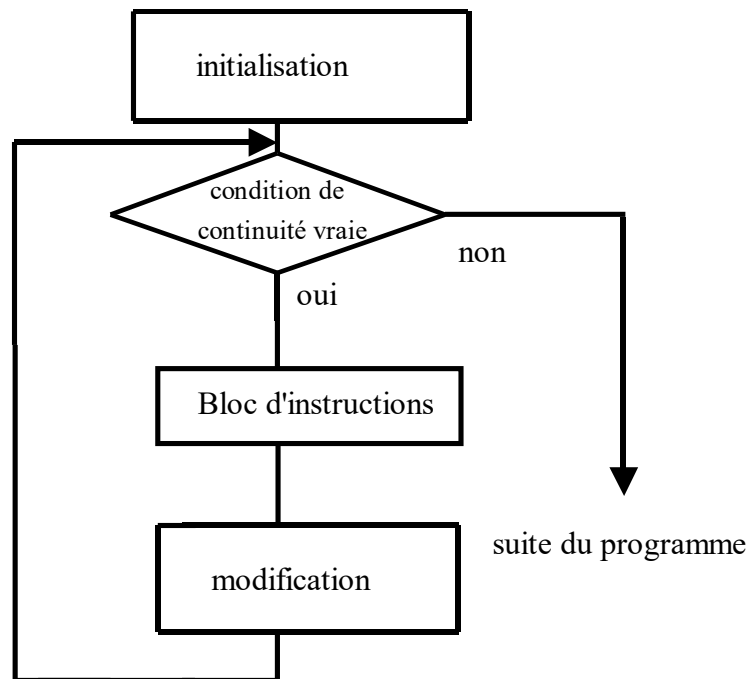
Remarque : pour afficher " il suffit d'écrire \"

Exemple pour afficher : vous avez saisi " == ",

il suffit d'écrire : document.write (" vous avez saisi \" == \" ") ;

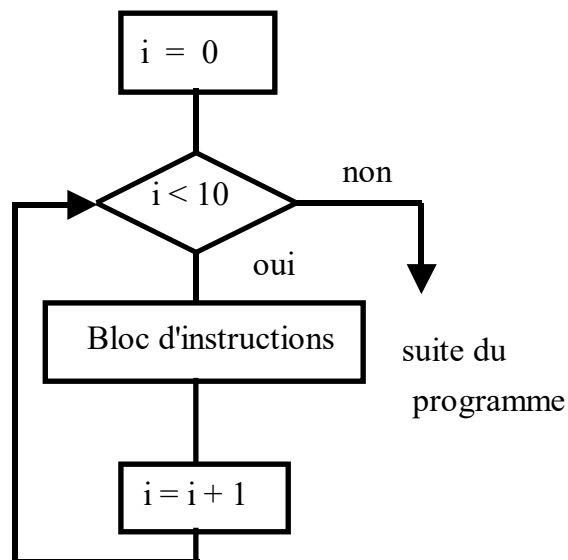
L'instruction for (POUR)

```
for (initialisation ; condition de continuité ; modification)  
{  
    < instruction>;          /* bloc d'instructions */  
}
```



Les 3 instructions du **for** ne portent pas forcément sur la même variable.

La boucle **for**($i = 0 ; i < 10 ; i++$) correspond au logigramme suivant :



Attention : **for**($i = 0 ; i < 10 ; i++$) ; // le point virgule empêche l'exécution de la boucle

Exemple :

```
for (var i=0 ; i<10 ; i++) //attention i++ est identique à i = i + 1
{
    window.alert("Itération n° " + i+1) ; // + permet de concaténer 2 chaînes
}
```

Exercice 7 :

Ecrire un programme qui calcule le **produit** de **3 nombres** saisis (méthode **window.prompt()**), en utilisant la structure **for** pour **répéter** la saisie des 3 nombres.

Exercice 8 :

Ecrire un programme qui affiche dans une page HTML (**document.write()**) la table de multiplication de **3**. Utiliser la structure **for** pour créer cette table de multiplication.

Exemple d’affichage ;

1 fois 3 = 3
2 fois 3 = 6
.....
10 fois 3 = 30

Exercice 9 :

On reprend **l'exercice 7** de calcul du produit des 3 nombres saisis, écrire le programme de telle sorte qu'il affiche également la **moyenne** de ces 3 nombres, toujours en **utilisant la structure for**.

Exercice 10 :

Reprendre l'exercice 7 de calcul du **produit** des 3 **nombres** saisis et écrire le programme de telle sorte qu'il affiche également le nombre de nombres **négatifs saisis**, en utilisant la structure **for** et la structure **if**. On utilisera une variable de **comptage** des nombres négatifs dans la structure conditionnelle.

Exercice 11 :

Reprendre l'exercice 7 de calcul du produit des 3 nombres saisis, écrire le programme de telle sorte qu'il affiche également le nombre de **nombres pairs** saisis, en utilisant la structure **for** et la structure **if** ainsi que l'opérateur arithmétique **modulo** (%). On utilisera une variable de **comptage** des nombres pairs dans la structure conditionnelle.

Exercice 12 :

Ecrire un programme qui combine l'exercice 10 et 11, en utilisant la structure **for** et 2 structure **if** ainsi que l'opérateur arithmétique **modulo** (%). On utilisera **2** variables de **comptage**, une pour structure conditionnelle des nombres négatifs et une pour la structure conditionnelle des nombres pairs.

Exercice 13 :

Ecrire le programme qui permet de calculer un nombre à un exposant donné. Le nombre **x** et l'exposant **n** seront saisis. Le calcul de x^n devra alors être effectué avec la méthode **Math.pow()**.

Remarque : utilisez le site **www.toutjavascript.com**

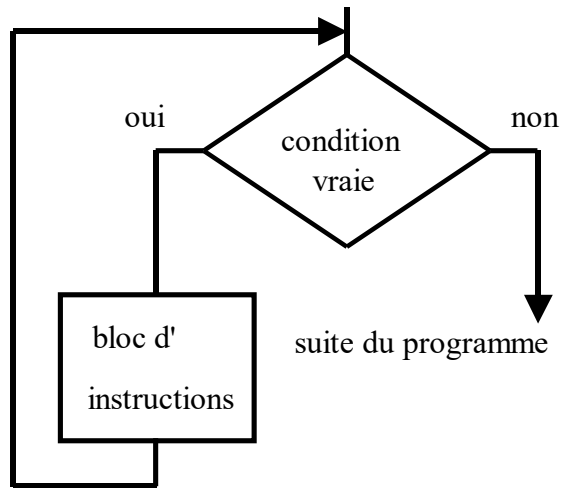
Exercice 14 :

Imaginons que la méthode **Math.pow()** n'existe pas en JavaScript. Ecrire le programme qui permet de calculer un nombre à un exposant donné. Le nombre **x** et l'exposant **n** seront saisis. Le calcul de x^n devra alors être effectué de façon itérative (**for**).

```
// r = 1 ;  
// x = 6 ;  
// r = r * x ; r alors correspond à 61 = 6  
// r = r * x ; r alors correspond à 62 = 36
```


L'instruction while (TANT QUE)

```
while (expression)  
{  
    < instruction>;          /* bloc d'instructions */  
}
```



La condition est testée au moins une fois. Le bloc d'instructions peut, dès lors, s'exécuter 0 ou n fois.

Remarque : `while (expression);` - sans la présence de bloc d'instructions - signifie : "**tant que l'expression est vraie attendre**".

Attention : `while(i<10);` // le point virgule empêche l'exécution du `while`

Exemple :

```
var i = 0; //Attention à bien initialiser la variable i  
while (i < 10) // tant que cette condition est vraie => 10 itérations  
{  
    i++; //la variable i est incrémentée de 1  
    window.alert("Itération n° " + i);  
}
```

Exercice 15 :

Ecrire un programme qui calcule le **produit** de 3 **nombres** saisis au clavier (méthode `window.prompt()`), en utilisant la structure **while** pour saisir ces 3 nombres.

Exercice 16 :

Ecrire un programme qui calcule et affiche le produit de n nombres saisis (méthode `window.prompt()`), en utilisant la structure **while**. Le nombre n sera saisi au clavier par l'utilisateur.

Exercice 17 :

Ecrire un programme qui calcule et affiche le produit d'un nombre indéterminé de nombres saisis par l'utilisateur (méthode **window.prompt()**), en utilisant la structure **while**. Le programme sera interrompu si le caractère saisi au clavier est égal à 0 (zéro).

Exercice 18 :

Ecrire un programme qui permet de trouver et d'afficher le plus petit parmi une série de 5 nombres saisis par l'utilisateur, en utilisant la structure **while** et la structure **if**. Il faudra passer par une variable intermédiaire.

Exercice 19 :

Ecrire le programme qui permet de calculer un nombre à un exposant donné. Le nombre **x** et l'exposant **n** seront saisis. Le calcul de x^n devra alors être effectué de façon itérative (**while**).

```
// r = 1 ;  
// x = 6 ;  
// r = r * x ; r alors correspond à  $6^1 = 6$   
// r = r * x ; r alors correspond à  $6^2 = 36$ 
```

Les instructions if et if . . . else (SI et SI ALORS ... SINON) imbriquées

Exemple :

```
var prix = 99;
if (prix == 49) {
    window.alert("Pas cher !");
} else // non égal à 49
if (prix == 99) { //première imbrication
    window.alert("Cher !");
} else //non égal à 99
if (prix == 149) { //deuxième imbrication
    window.alert("Très cher !");
} else { //non égal à 149
    window.alert(" Cher ou pas cher ?");
} // fin de l'instruction if
```

Exercice 20 :

Ecrire un programme qui affiche dans une boîte d’alerte la saison en fonction du mois saisi (1, 4, 7, 10 seront les valeurs acceptables).

Exercice 21 :

Ecrire un programme qui permet d’afficher un message de bienvenue comme par exemple (« Bonjour Madame ! »), en fonction de l’état-civil (M pour Monsieur, F pour Madame, L pour Mademoiselle). L’état-civil sera saisi avec la méthode `window.prompt()`.

Exercice 22 :

Ecrire un programme qui permet la saisie d’un **nombre** et affiche l’intervalle dans lequel il se trouve (traiter les intervalles 5-10, 10-15, 15-20 et « hors intervalle » pour les autres cas).

Exercice 23 :

Ecrire un programme qui affiche le **montant à payer** d’une facture.

On dispose d’un **montant déjà calculé** et on veut effectuer une **remise** de 1% si le montant de la facture dépasse 150 €, 2% si le montant de la facture dépasse 250 €, 3% si le montant de la facture dépasse 350 €.

L'instruction switch (SELON)

L'instruction **switch** permet des choix multiples sur des variables.

```
switch(variable) //      au cas où la variable vaudrait :
{
    case valeur1: <instruction1> ; // si variable == valeur1
    break ; //permet de sortir du bloc formé par le switch
    case valeur2: <instruction2>;      // si variable == valeur2
    break;
    default: <instruction3> ; // ni valeur2, ni valeur1
}
```

Exemple :

```
var prix = 99;
switch (prix) {
    case 49:
    {
        window.alert("Pas cher !");
        break; //break permet d'interrompre le switch si prix vaut 49
    }
    case 99:
    {
        window.alert("Cher !");
        break;
    }
    case 149:
    {
        window.alert("Très cher !");
        break;
    }
    default:
    { //pour tous les autres cas
        window.alert(" Cher ou pas cher ?");
    }
} // fin de l'instruction switch
```

Exercice 24 :

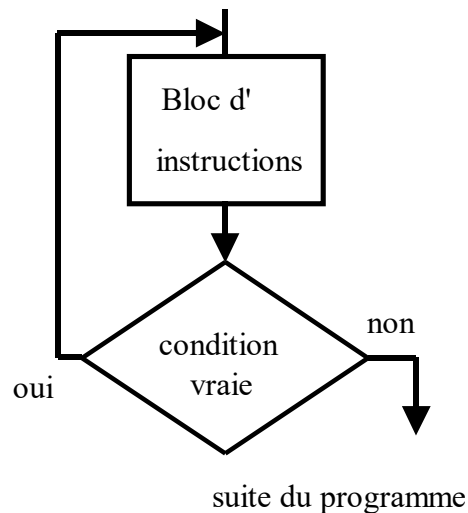
Ecrire un programme qui affiche dans une boîte d'alerte la saison en fonction du mois saisi (1, 4, 7, 10), en utilisant un **switch**.

Exercice 25 :

Ecrire un programme qui permet d'afficher un message de bienvenue comme par exemple (« Bonjour Madame ! »), en fonction de l'état-civil (M pour Monsieur, F pour Madame, L pour Mademoiselle). L'état-civil sera saisi avec la méthode **window.prompt()**. Vous devez utiliser le **switch**.

L'instruction do ... while (Répéter tant que)

```
do
{
    < instruction>;          /* bloc d'instructions */
    < instruction>;
    < instruction>;
}
while (expression);
```



Le bloc d'instruction est exécuté au moins une fois. La condition est examinée après la première exécution du bloc d'instructions.

Exemple :

```
var etoile = "*";
var n = 1;
do {
    document.write(ettoile + "<br>");
    n++;
    etoile = etoile + "*";
} while (n < 3);
```

Exercice 26 :

Ecrire un programme qui calcule et affiche le **produit** d'un nombre **défini** de nombres **saisis** (méthode **window.prompt()**), en utilisant la structure **do ... while**. Le nombre **défini** de nombres saisis sera entré au clavier par l'utilisateur.

Exercice 27 :

Ecrire un programme qui calcule et affiche le **produit** d'un nombre **quelconque** de nombres saisis (méthode **window.prompt()**), en utilisant la structure **do ... while**. Le programme sera interrompu si le caractère saisi au clavier est égal à 0 (zéro).

Exercice 28 :

Ecrire un programme qui permet de trouver et d'afficher le plus **petit** parmi une liste 5 nombres saisis par l'utilisateur, en utilisant la structure **do ... while** et la structure **if**. Il faudra passer par une variable **intermédiaire**.

Exercice 29 :

Ecrire un programme qui permet de tracer et d'afficher un **rectangle** (avec des étoiles) dont la **longueur** et la **largeur** sont déterminées par l'utilisateur, en utilisant les structures de votre choix.

Exercice 30 :

Ecrire un programme qui permet de tracer et d'afficher un **triangle isocèle** (avec des étoiles) dont la **hauteur** est saisie par l'utilisateur, (la largeur sera égale à la hauteur multipliée par 2) en utilisant les structures de votre choix.

Remarque : Utilisez " " pour insérer un espace.

Isocèle : au moins 2 cotés égaux.

Exercice 31 :

Ecrire un programme qui permet de tracer et d'afficher un **losange** (avec des étoiles) dont la **hauteur** et la **largeur(ou hauteur *2)**, sont saisies par l'utilisateur, en utilisant les structures de votre choix.

Exercice 32 :

Ecrire un programme qui permet de tracer et d'afficher deux **triangles** isocèles opposés par leur sommet (avec des étoiles) dont la **hauteur** et la **largeur(ou hauteur *2)**, de chaque triangle sont saisies par l'utilisateur, en utilisant les structures de votre choix.

Exercice 33 :

Ecrire un programme qui demande à l'utilisateur de saisir les valeurs en centimètres des **3 côtés** d'un **triangle** et affiche si le triangle est rectangle, isocèle, isocèle rectangle, équilatéral ou quelconque.

Exercice 34 :

Ecrire un programme qui demande à l'utilisateur de saisir **l'âge** d'un enfant et qui l'informe de sa **catégorie** :

- Poussin de 6 à 7 ans
- Pupille de 8 à 9 ans
- Minime de 10 à 11 ans
- Cadet de 12 à 14 ans

Exercice 35 :

Ecrire un programme qui demande à un utilisateur de saisir au clavier son genre (H ou F), sa taille en mètre et sa masse en kg. Le programme devra ensuite calculer l'indice de masse corporelle : $\text{Indice} = \text{Masse} / (\text{Taille} * \text{Taille})$

- Si indice ≥ 25 pour les hommes : Message = « Attention !! »
- Si indice ≥ 23 pour les femmes : Message = « Attention !! »
- Si indice ≤ 19 pour les hommes : Message = « Attention !! il faut grossir »
- Si indice ≤ 18 pour les femmes : Message = « Attention !! il faut grossir »
- Autres valeurs : Poids de forme

IV Les Fonctions

Objectifs

- Concevoir et développer des sous programmes, des procédures et des fonctions

En JavaScript, le mot clé **function** est utilisé indifféremment pour les procédures et pour les fonctions. La particularité est qu'une procédure ne renvoie pas de valeur en retour.

Syntaxe de la procédure :

```
function Ma_Procédure (parametre1, parametre2, ...)  
{  
  
}
```

Appel de la procédure : Ma_Procédure (param1, param2,...);

Syntaxe de la fonction :

```
function Ma_Fonction (parametre1, parametre2, ...)  
{  
  
    return (valeur_retour);  
}
```

Appel de la fonction : Ma_variable = Ma_Fonction (param1, param2,...);

Une variable précédée du mot clé "var" sera locale.

Une variable déclarée implicitement dans la fonction (non précédée du mot clé "var") sera globale en tout point du document.

Exemple :

```
<!DOCTYPE html>
<!-- Fonctions.html -->
<html>
<head>
<title>Vos résultats</title>

<script>
    // Variables locales
    //var a,b;
    // Saisies
    a = parseInt(prompt("Paramètre a : "));
    b = parseInt(prompt("Paramètre b : "));

    document.write("valeur de a : " + a + "<br />");
    document.write("valeur de b : " + b + "<br />");

    // Appel de la procédure
    doublement(a, b);
    document.write("valeur de a : " + a + "<br />");
    document.write("valeur de b : " + b + "<br />");

    // Appel de la fonction
    valeur_retour = multipli(a, b);
    document.write("valeur de a*b : " + valeur_retour + "<br />");

function doublement(x, y)
{
    x = x * 2;
    y = y * 2;
    document.write("Les valeurs doubles sont : (" + x + "," + y +
    ")<br />");
}

function multipli(p1, p2)
{
    var result;
    result = p1 * p2;
    return result;
}
</script>
</head>
</html>
/*
    Après exécution :
    valeur de a : 4
    valeur de b : 2
    Les valeurs doubles sont : (8,4)
    valeur de a : 4
    valeur de b : 2
    valeur de a*b : 8
    */
```


Exemple d'utilisation d'une fonction JavaScript avec saisies via un formulaire.

```
<!DOCTYPE html>
<!-- Calcul imc simple.html -->
<html>
<head>
<title>Votre IMC</title>

<script>

function Valider(ideal)
{
    var taille = ideal.taille.value
    var poids = ideal.poids.value
    if ((taille == "Entrez votre taille (en cm) ici.") || (taille == "") || (poids == "Entrez
votre poids ici.") || (poids == ""))
    {
        alert("Si vous voulez que je calcule votre indice de masse corporel il faut
entrer vos mensuration");
    }
    else
    {
        //Fonction calcul de l'imc
        var nombre1 = (taille*0.01) //conversion de la taille des centimètres en
mètres
        var nombre2 = (nombre1*nombre1) //taille au carré
        var nombre3 = (poids/nombre2) //poids divisé par la taille
        alert("Votre indice de masse corporel est de: " + nombre3 + "!");
        // Pour afficher le résultat dans la page HTML
        ideal.imc.value=nombre3;
    }
}
</script>
</head>
<body>
<form action="" method="post" name="ideal" id="ideal"><br />
<input type="text" value="Entrez votre taille (en cm) ici." name="taille" size="30"><br
/><br />
<input type="text" value="Entrez votre poids ici." name="poids" size="30"><br /><br />
<input type="button" value="Calculer" name="calculez" onClick="Valider(this.form)"><br />
<input type="reset" value="Remettre à zero" name="reset"><br />
<input type="text" value="IMC" name="imc" size="10">
</form>
</body>
</html>
```



Votre IMC

Entrez votre taille (en cm) ici.

Entrez votre poids ici.

Calculer

Remettre à zero

IMC

Exemple d'utilisation d'une fonction en JavaScript pour contrôler une valeur trop grande.

```
<!DOCTYPE html>

<html>
  <head>
    <title>Formulaire de contrôle</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="css/styles.css">
  </head>
  <body>
    <script>
      function VerifValeur(val)
      {
        val = parseInt(val);
        if (val <100 ) alert ("Valeur correcte : "+val);
        else alert ("Valeur trop grande : "+val);
      }
    </script>
    <h1>FICHE DE CONTROLE</h1>
    <form action="#" method="post" >

      VALEUR : <input type="text" name="nom" size="8" value="">
      <INPUT type=button value="Vérifier la valeur"
        onClick="VerifValeur(this.form.nom.value)">

    </form>
  </body>
</html>
```

```

<!DOCTYPE html><!-- FORMULAIRE -->
<html>
  <head>
    <title>Formulaire de commande</title> <meta charset="UTF-8">
  </head>
  <body>
    <script>
      function VerifArticle(article1)
      {
        article=article1+2;
        alert ("Référence article :"+article);
        article1 = parseInt(article1);
        article=article1+2;
        alert ("Référence article :"+article);
      }
      function VerifArticle2(article2)
      {
        article=article2.value;
        alert ("Référence article :"+article);
      }
      function VerifArticle3(article3)
      {
        article=article3.value;
        alert ("Référence article :"+article);
      }
      function VerifArticle4(article4)
      {
        article=article4.value;
        alert ("Référence article :"+article);
      }
      function VerifArticle5(article5)
      {
        article=article5.value;
        alert ("Référence article :"+article);
      }
    </script>
    <form action="#" method="post" name="maForme" enctype="text/plain">
      Article1 : <input type="text" name="article1" size="8" value="">
                 <INPUT type="button" value="Vérifier l'article"
                       onClick="VerifArticle(this.form.article1.value)"> <br>
      Article2 : <input type="text" name="article2" size="8" value="">
                 <INPUT type="button" value="Vérifier l'article"
                       onClick="VerifArticle2(this.form.article2)"> <br>
      Article3 : <input type="text" name="article3" size="8" value="">
                 <INPUT type="button" value="Vérifier l'article"
                       onClick="VerifArticle3(this.form)"> <br>
      Article4 : <input type="text" name="article4" size="8" value="">
                 <INPUT type="button" value="Vérifier l'article"
                       onClick="VerifArticle4(maForme)"> <br>
      Article5 : <input type="text" name="article5" size="8" value="">
                 <INPUT type="button" value="Vérifier l'article"
                       onClick="VerifArticle5(article5)"> <br>
    </form>
  </body>
</html>

```

Exemple d'un calculateur.

| Simulateur de calcul de coût de cotisation annuel | | |
|---|--------------------------------|----------------------------------|
| | Nombre | Coût |
| Nb de personnes au total | <input type="text" value="0"/> | <input type="text"/> |
| Dont adultes | <input type="text" value="2"/> | <input type="text" value="100"/> |
| Dont enfants de moins de 12 ans | <input type="text" value="2"/> | <input type="text" value="30"/> |
| Dont enfants de 12 à 18 ans | <input type="text" value="2"/> | <input type="text" value="20"/> |
| Total | | <input type="text" value="150"/> |

```

<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Calculateur</title>
</head>
<body>

<table>
<tr>
  <th colspan=3> Simulateur de calcul de coût de cotisation annuel </th>
</tr>

<tr>
  <th></th>
  <th> Nombre </th>
  <th> Coût </th>
</tr>
<tr>
  <th> Nb de personnes au total </th>
  <td> <input type="number" value="0" onfocus ="this.value=' ' "></td>
  <td> <input type="text" value="" readonly></td>
</tr>
<tr>
  <th> Dont adultes </th>
  <td> <input onfocus ="this.value=' ' " type="number" id="field1" value="0"
onchange="maFonction()" "></td>
  <td> <input type="number" id="field11" value="" readonly> </td>
</tr>
<tr>
  <th> Dont enfants de moins de 12 ans </th>
  <td> <input onfocus ="this.value=' ' " type="number" id="field2" value="0"
onchange="maFonction()" "></td>
  <td> <input type="number" id="field22" readonly value=""></td>
</tr>
<tr>
  <th> Dont enfants de 12 à 18 ans </th>
  <td> <input onfocus ="this.value=' ' " type="number" id="field3" value="0"
onchange="maFonction()" "></td>
  <td> <input type="number" readonly id="field33" "></td>
</tr>
<tr>
  <th></th>
  <th> Total </th>
  <td> <input type="number" id="fieldsomme" readonly "></td>
</tr>
</table>

<script>
function maFonction() {
  document.getElementById("field11").value = parseInt(document.getElementById("field1").value)*50;
  document.getElementById("field22").value =
parseInt(document.getElementById("field2").value)*50*3/10;
  document.getElementById("field33").value =
parseInt(document.getElementById("field3").value)*50*2/10;
  document.getElementById("fieldsomme").value=parseInt(document.getElementById("field11").value)+
parseInt(document.getElementById("field22").value)+parseInt(document.getElementById("field33").val
ue);
}

</script>

```

Exercice 1 : Réalisez deux fonctions en JavaScript, l'une permettant de calculer la surface d'un cercle et l'autre permettant de calculer le périmètre en passant le rayon en paramètre :

surface (rayon)
périmètre (rayon)

Rappels : Surface = $\pi * r * r$ Périmètre = $2 * \pi * r$

En Javascript, la constante PI s'écrit : Math.PI

Exercice 2 : Refaire le même programme en utilisant un formulaire (balise form) pour réaliser la saisie des informations et pour afficher le résultat.

Exercice 3 : Afficher les résultats d'une course.
La saisie du temps en minute et de la distance parcourue sera faite via un formulaire.
Le résultat affiché sera donné en km/h.

V Les événements

Les événements sont des actions de l'utilisateur, qui vont pouvoir donner lieu à une interactivité. L'événement par excellence est le clic de souris. Grâce à JavaScript, il est possible d'associer des fonctions, des méthodes à des événements tels que le passage de la souris au-dessus d'une zone, le changement d'une valeur, ...

Ce sont les gestionnaires d'événements qui permettent d'associer une action à un événement. La syntaxe d'un gestionnaire d'événement est la suivante :

```
onEvenement= "Action JavaScript ou Fonction ( ) ;"
```

Utilisation de l'événement onclick.

L'événement onclick se produit lorsque l'utilisateur clique sur l'élément associé à l'événement (un lien hypertexte ou un élément de formulaire).

L'événement onclick est associé au clic du bouton gauche de la souris. Il exécute le code javascript contenu dans l'attribut "onclick" de l'élément HTML sur lequel il a été appliqué.

Exemple de click sur un lien :

```
<a href="" onClick="alert('Vous avez cliqué sur un lien') ;">Cliquez ici!</a>
```

Exemple de click sur un BOUTON :

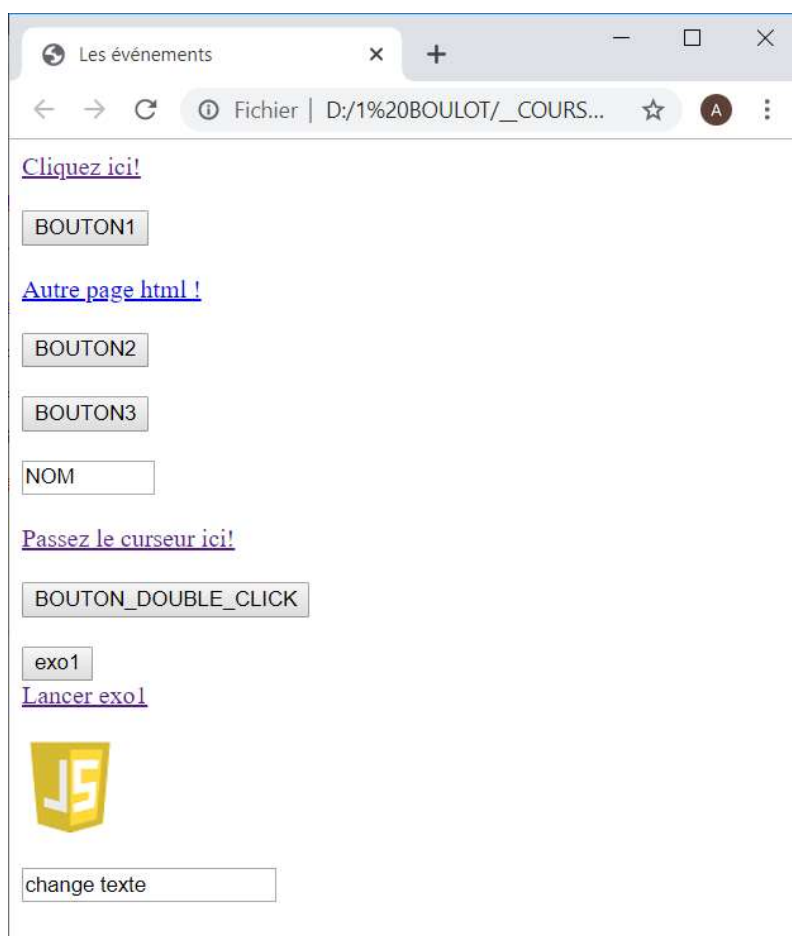
```
<input type="button" value="BOUTON" onClick="alert('Vous avez cliqué sur un bouton') ;">
```

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">    <title>Les événements</title>
  </head>
  <body>
    <a href="" onClick="alert('Vous avez cliqué sur un lien') ;">Cliquez ici!</a>
    <br><br>
    <input type="button" value="BOUTON1" onClick="alert('Vous avez cliqué sur un bouton');">    <br><br>

    <a href="page.html" >Autre page html !</a>    <br><br>
    <button onclick="window.location.href='page.html'">BOUTON2</button>    <br><br>
    <form method="get" action="page.html">
      <button type="submit">BOUTON3</button>
    </form>    <br>
    <input type="text" value="NOM " size="8"
      onClick="alert('Vous avez cliqué sur un element de formulaire') ;">
    <br><br>
    <a href="" onMouseOver="alert('Curseur passé sur un lien');">Passez le curseur ici!</a>
    <br><br>
    <input type="button" value="BOUTON_DOUBLE_CLICK"
      onDblClick="alert('Vous avez double cliqué sur un bouton') ;">    <br><br>
    <input type="button" value="exo1" onclick = exercicel();>    <br>
    <a href="" onclick = exercicel();> Lancer exo1</a>    <br><br>
    
    <br><br>
    <input type="text" value="change texte" onchange="alert('changement');"/>
  </body>
</html>

```



Evènements liés à la souris

onclick
ondblclick
onmousedown
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel
onscroll

Evènements liés au clavier

onkeydown
onkeypress
onkeyup

Evènements liés au multimédia

onabort
oncanplay
oncanplaythrough
ondurationchange
.....

Evènements liés au formulaire

onblur
onchange
oncontextmenu
onfocus
onformchange
onforminput
oninput
oninvalid
onselect
onsubmit

Evènements liés au drag and drop

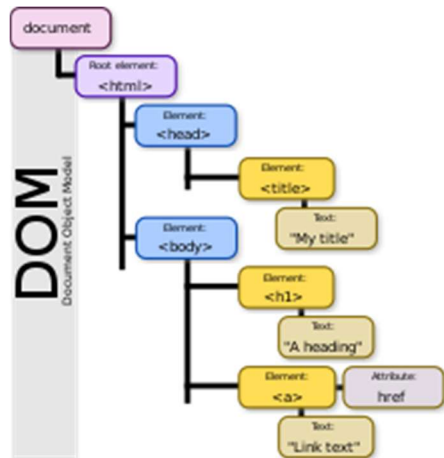
ondrag
ondragend
ondragenter
ondragleave
ondragover
ondragstart
ondrop

Evènements associés à la fenêtre

onafterprint
onbeforeprint
onbeforeunload
onblur
onerror
onfocus
onhaschange
onload
onmessage
onoffline
ononline
onpagehide
onpageshow
onpopstate
onredo
onresize
onstorage
onundo
onload

VI Le DOM (Document Object Model)

Le Document Object Model (**DOM**) est une interface de programmation normalisée par le W3C, qui permet à des scripts d'examiner et de modifier le contenu du navigateur web.



La méthode **getElementById()** de [Document](#) renvoie un objet [Element](#) représentant l'élément dont la propriété **id** correspond à la chaîne de caractères spécifiée. Étant donné que les ID d'élément doivent être uniques, s'ils sont spécifiés, ils constituent un moyen utile d'accéder rapidement à un élément spécifique.

Syntaxe : `var element = document.getElementById(id);`

Paramètres : **id** L'ID (*identifiant*) de l'élément à localiser. Il est une chaîne de caractères sensible à la casse qui est unique ; un seul élément peut avoir un ID donné.

Valeur de retour : Un objet [Element](#) décrivant l'objet élément du DOM correspondant à l'ID spécifié ou null si aucun n'a été trouvé dans le document.

Exemple

Contenu HTML

```
<html>
  <head>
    <title>getElementById example</title>
  </head>
  <body>
    <p id="para">Some text here</p>
    <button onclick="changeColor('blue');">blue</button>
    <button onclick="changeColor('red');">red</button>
  </body>
</html>
```

Contenu JavaScript

```
function changeColor(newColor) {
  var elem = document.getElementById('para');
  elem.style.color = newColor;
}
```

https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_ev_onclick

ou https://www.w3schools.com/tags/ev_onclick.asp

```
<!DOCTYPE html>
<html>
<body>
  <button onclick="myFunction()">Click me</button>

  <p id="demo"></p>
  <p> Une fonction est déclenchée lorsque l'utilisateur clique sur le bouton. La
  fonction génère du texte dans l'élément p avec id = "demo".</p>

  <script>
    function myFunction() {
      document.getElementById("demo").innerHTML = "Hello World";
    }
  </script>
</body>
</html>
```

Click me

Hello World

```
<!DOCTYPE html>
<html>
<body>
  <p id="demo" onclick="myFunction()">Cliquez sur moi pour changer la couleur
  de mon texte.</p>

  <p>Une fonction est déclenchée lorsque l'utilisateur clique sur l'élément p. La
  fonction définit la couleur de l'élément p en rouge.</p>

  <script>
    function myFunction() {
      document.getElementById("demo").style.color = "red";
    }
  </script>
</body>
</html>
```

Click me to change my text color.

https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_ev_onclick3

ou https://www.w3schools.com/tags/ev_onclick.asp

```

<!DOCTYPE html>
<html>
<body>
    Field1: <input type="text" id="field1" value="Hello World!"><br>
    Field2: <input type="text" id="field2"><br><br>

    <button onclick="myFunction()">Copy Text</button>

    <p>Une fonction est déclenchée lorsque l'utilisateur clique sur le bouton.<br>
    La fonction copie le texte de Field1 dans Field2.</p><br><br>

    <script>
        function myFunction() {
            document.getElementById("field2").value =
            document.getElementById("field1").value;
        }
    </script>

```

Field1: Hello World!

Field2: Hello World!

Copy Text

```

Field3: <input onchange="myFunction2()" type="text" id="field3" value="123"><br>
Field4: <input onchange="myFunction2()" type="text" id="field4" value="456"><br>
Field5: <input type="text" id="field5"><br><br>
<!-- <button onclick="myFunction2()">Somme</button> -->

<p>Une fonction est déclenchée lorsque l'utilisateur modifie le contenu dans Field3 ou
4.<br>
La fonction copie le résultat (somme) dans Field5.</p><br><br>

<script>
    function myFunction2() {
        document.getElementById("field5").value =
        parseInt(document.getElementById("field3").value)+parseInt(document.getEle
mentById("field4").value);
    }
</script>

```

```

</body>
</html>

```

Field3: 4

Field4: 55

Field5: 59

```
<!DOCTYPE html>
<head>
  <title> Exemple_DOM </title>
</head>

<body>
<div id="maDiv">
  <p> Un peu de texte </p>
</div>

<input id="entree" type="number" value=3 >

</body>
<script>

// https://developer.mozilla.org/fr/docs/Web/API/HTMLInputElement
// https://developer.mozilla.org/fr/docs/Web/API/Element/innerHTML

  var val=document.getElementById("maDiv");
  alert(val); // object HTMLDivElement

  alert(val.value); // undefined
  alert(val.innerHTML); // <p> Un peu de texte </p>
  alert(val.textContent); // Un peu de texte

  window.document.getElementById('maDiv').innerHTML = "<p> Autre texte </p>";
  // affiche dans la page : Autre texte (à la place de un peu de texte)

  var val3=document.getElementById("maDiv").innerHTML;
  alert(val3); // Autre texte

  var nb=document.getElementById("entree");
  alert(nb); // object HTMLInputElement

  alert(nb.value); // 3

  document.getElementById("entree").value=8;
  nb2=document.getElementById("entree");
  alert(nb2.value); // 8

</script>
</html>
```

VII Comment manipuler les tableaux ?

Objectifs

- Savoir utiliser l'objet **Array**
- Savoir manipuler et trier les tableaux
- Utiliser le site <http://toutjavascript.com/reference/index.html>

L'objet Array

Lorsqu'il s'agit de manipuler une grande quantité d'informations, le recours aux seules variables n'est parfois pas suffisant. L'usage de tableaux afin de stocker ces informations est, alors, d'un grand secours. Les **tableaux** JavaScript n'ont rien à voir avec les tableaux HTML, leur rôle n'est pas de présenter l'information mais de la stocker et de la mettre à disposition des programmes JavaScript pour la manipuler (ajouter, supprimer, trier, ...). Un tableau ne peut pas recevoir plus de 256 valeurs, mais les valeurs stockées dans celui-ci peuvent être de type différent (texte, numérique, ...). L'objet **Array** représente donc à lui seul, une variable à l'intérieur de laquelle sont stockées des informations identifiées par un numéro d'indice débutant par **zéro** (comme bien souvent en JavaScript).

L'objet **Array** est un objet **natif** (c'est-à-dire appartenant au **core**) de JavaScript, qui dispose de plusieurs méthodes. Pour utiliser un tableau, il convient tout d'abord de le déclarer. Il existe plusieurs syntaxes pour la déclaration d'un tableau.

Il est, par exemple, possible de déclarer un tableau et le nombre d'éléments qu'il contient par l'instruction suivante :

```
var tableau = new Array(5) ; // contient 5 éléments
```

Mais l'indication du nombre d'éléments du tableau n'est pas obligatoire, en effet, **JavaScript** gère les tableaux de manière dynamique et s'adapte donc, au nombre d'éléments qui lui est fourni. Ainsi, la syntaxe suivante convient également :

```
var tableau = new Array() ;
```

Attention à ne pas oublier les parenthèses ouvrantes et fermantes après **Array**. Puis, vous affectez les valeurs dans le tableau par l'instruction suivante :

```
//tableau= ["valeur1", "valeur2", "valeur3", "valeur4"] ;
```

Tout comme avec les variables, il est possible de réaliser la déclaration et l'affectation simultanément. Donc, la syntaxe suivante est aussi valide et préférable : (syntaxe dite "POO Programmation Orientée Objets")

```
var tableau = new Array("lundi", "mardi", "mercredi", "jeudi", "vendredi",  
"samedi", "dimanche") ;
```

Cette syntaxe est également valable. (syntaxe littérale)

```
var tableau= ["valeur1", "valeur2", "valeur3", "valeur4"] ;
```

L'accès aux valeurs stockées dans le tableau se fait par leur numéro d'indice :

Ainsi, **document.write(tableau[4]) ; //affichera vendredi**

Évidemment, le numéro d'indice peut provenir d'un compteur de boucle, ce qui permet de passer en revue l'ensemble des éléments du tableau.

Exercice 1 : afficher le nombre d'éléments d'un tableau comportant les jours de la semaine (semainier).

Après la définition des éléments du tableau, le calcul du nombre d'éléments le composant est facilité par la propriété **length**. (Voir sur *toutjavascript*)

Exercice 2 : afficher dans un document HTML les jours de la semaine, contenus dans un tableau.

| |
|---|
| lundi mardi mercredi jeudi vendredi samedi dimanche |
|---|

Le script utilise un compteur et une boucle pour afficher le contenu d'un tableau dont vous connaissez le nombre d'éléments.

Dans le cas où vous ne connaissez pas le nombre d'éléments d'un tableau, il est possible d'utiliser la propriété **length**, correspondant au nombre de valeurs ou d'éléments du tableau.

En vous aidant de l'exemple ci-après, réalisez un programme qui permet d'afficher le tableau verticalement.

Les jours de la semaine :

| Indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|--------|-------|-------|----------|-------|----------|--------|----------|
| Valeur | lundi | mardi | mercredi | jeudi | vendredi | samedi | dimanche |

```

<!-- tab horizontal-->
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <script>

      var tableau = new Array("lundi", "mardi", "mercredi", "jeudi",
                              "vendredi", "samedi", "dimanche");

      document.write( " <br> Les jours de la semaine : <br> <br>" );

      document.write( " <table Border=5> <tr><th> Indice </th>" );
      for (var i = 0; i< tableau.length; i++)
      {
        document.write("<td Align=CENTER>"+i+"</td>");
      }
      document.write("</tr><tr><th> Valeur </th>");
      for (var i = 0; i< tableau.length; i++)
      {
        document.write("<td> "+tableau[i]+" </td> ");
      }
      document.write("</tr></table>");
    </script>
  </body>
</html>

```

Vérifiez que : **tableau.pop()**; supprime la dernière case du tableau

a. Les propriétés

Des trois propriétés, seule **length** est très souvent rencontrée dans les scripts. Elle permet ainsi de déterminer le nombre d'itérations d'une boucle, correspondant au nombre d'éléments du tableau.

| Propriété | Résultat | Exemple |
|--------------------|---|---------------------------------------|
| constructor | Indique comment a été créé un objet référencé. | var tab=new Array(1,2,7) ; |
| length | Correspond au nombre d'éléments du tableau. | var longueur = tab.length; //3 |
| prototype | Permet d'ajouter des propriétés personnalisées à l'objet. | |

b. Les méthodes

Les méthodes applicables à l'objet **Array** sont très utiles et disposent d'une syntaxe relativement simple à retenir :
nomdutableau.méthode(arguments) ;

| Méthode | Action | Retour |
|-------------------|---|--|
| concat() | Concatène deux tableaux en un. | Retourne le tableau concaténé |
| indexOf() | Cherche un élément dans le tableau | Retourne la position de l'élément |
| join() | Crée une chaîne de caractères à partir des éléments du tableau. | Retourne le tableau sous forme de chaîne |
| pop() | Supprime le dernier élément d'un tableau. | Retourne l'élément supprimé |
| push() | Ajoute des éléments en fin de tableau. | Retourne la nouvelle longueur |
| reverse() | Inverse l'ordre des éléments d'un tableau. | |
| shift() | Supprime le premier élément d'un tableau. | Retourne l'élément supprimé |
| slice() | Extrait une partie d'un tableau. | Retourne la partie extraite |
| sort() | Trie les éléments d'un tableau. | |
| splice() | Supprime une partie des éléments d'un tableau. | |
| toString() | Convertit un tableau en une chaîne de caractères. | Retourne une chaîne |
| unshift() | Ajoute un élément au début d'un tableau. | Retourne la nouvelle longueur |
| valueOf() | Retourne un élément précis d'un tableau. | |

Exercice 3 : afficher dans un document HTML, le contenu d'un tableau composé des jours de la semaine suivi chacun de la chaîne de caractères « puis ».

lundi puis mardi puis mercredi puis jeudi puis vendredi puis samedi puis dimanche

L'utilisation de la méthode **join()** permet d'adjoindre une chaîne de caractères (ici, le mot puis), entre les éléments du tableau. Mais, cet argument est facultatif. La chaîne de caractères correspond, alors, aux seuls éléments du tableau séparés par des virgules et il est possible de la traiter comme toute autre chaîne de caractères. Ex :
tab.join("puis") ;

Exercice 4 : afficher le jour de la semaine de la date courante. (Voir l'objet **Date** pour le jour de la semaine)

```
var date = new Date() ;  
var jour = date.getDate() ;
```

Vous devez utiliser le semainier créé auparavant.

Afficher le mois en toutes lettres de la date courante. Vous devez créer un tableau des mois en toutes lettres (exemple : « janvier », « février », ...)

Exercice 5 : afficher en rouge les éléments d'un tableau après les avoir convertis en chaîne de caractères. (Voir l'objet **String** pour la coloration en rouge). **Facultatif**

```
Voici la semaine en débutant par lundi : lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche  
lundi mardi mercredi jeudi vendredi samedi dimanche
```

Exercice 6 : afficher le contenu d'une **concaténation** de deux tableaux, le premier correspondant aux 3 premiers jours de la semaine, le second aux 4 derniers.

```
lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche
```

Les deux tableaux, comprenant des valeurs différentes, sont facilement concaténés à l'aide de cette méthode.

Exercice 7 : afficher, dans la page HTML, l'ensemble des jours de la semaine puis sans le dimanche et enfin sans le samedi.

```
Voici la semaine complète : lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche  
Voici la semaine sans dimanche : lundi,mardi,mercredi,jeudi,vendredi,samedi  
Puis sans samedi : lundi,mardi,mercredi,jeudi,vendredi
```

Les méthodes qui suivent se fondent sur le principe de la structure de pile. C'est-à-dire que, pour ajouter ou supprimer des éléments dans un tableau, vous ne pouvez le faire que par le haut ou par le bas. Autrement dit, les éléments peuvent être extraits soit dans l'ordre dans lequel ils ont été placés, soit dans l'ordre inverse.

Il faut faire usage ici, de la méthode **pop()** dont la syntaxe d'utilisation est la suivante :

```
nomdutableau.pop(); // supprime le dernier élément du tableau
```

L'usage successif de la méthode **pop()** supprime les éléments du tableau les uns après les autres, en partant du dernier élément.

La méthode **push()** ajoute des éléments en fin de tableau

Exercice 8 : afficher, dans la page HTML, les jours de la semaine dans l'ordre puis dans l'ordre inverse.

Voici les jours de la semaine dans l'ordre : lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche
Voici les jours de la semaine dans le désordre : dimanche,samedi,vendredi,jeudi,mercredi,mardi,lundi

La méthode **reverse()** inverse l'ordonnancement des valeurs d'un tableau.

Exercice 9 : afficher, dans une page HTML, l'ensemble des jours de la semaine en partant du lundi, puis en partant du dimanche (la semaine commence le dimanche dans certain pays).

Il suffit de créer une chaîne de caractères et d'utiliser les bons indices

Exercice 10 : afficher, dans une page HTML, l'ensemble des jours de la semaine puis sans le lundi et enfin sans le mardi.

Voici la semaine complète : lundi,mardi,mercredi,jeudi,vendredi,samedi,dimanche
Voici la semaine sans lundi : mardi,mercredi,jeudi,vendredi,samedi,dimanche
Puis sans mardi : mercredi,jeudi,vendredi,samedi,dimanche

La méthode **shift()** fonctionne à l'identique de la méthode **pop()** mais dans l'autre sens (c'est-à-dire en supprimant le premier élément du tableau).

Pour ajouter un élément en début de tableau, il suffit d'utiliser la fonction **unshift()**.

Exercice 11 : afficher, dans une page HTML, uniquement les jours d'une semaine de travail (du lundi au vendredi).

La semaine de travail comprend les jours suivants : lundi,mardi,mercredi,jeudi,vendredi

La méthode **slice()** utilise deux arguments, placés entre parenthèses. Cette méthode retourne une partie du tableau. La syntaxe est donc la suivante :

nouveautableau=tableau.slice(n,m) ;

Où **n** correspond à l'indice inférieur des éléments à extraire et **m** à l'indice supérieur. Si **m** n'est pas renseigné, tous les éléments du tableau sont alors extraits et si **m** est égal à **n**, aucun des éléments du tableau n'est extrait. Le **tableau** d'origine n'est pas modifié. Idem à copier-coller.

Exercice 12 : afficher, dans une page HTML, les jours de la semaine triés par ordre alphabétique.

Les jours de la semaine triés par ordre alphabétique : dimanche,jeudi,lundi,mardi,mercredi,samedi,vendredi
Les jours de la semaine triés en ordre inverse : vendredi,samedi,mercredi,mardi,lundi,jeudi,dimanche

La méthode **sort()** dispose d'une fonction de comparaison permettant d'obtenir un ordre de tri croissant ou décroissant. La syntaxe d'utilisation est la suivante :

nouveautableau=tableau.sort(fonctiondecomparaison) ;

Si cette fonction n'est pas utilisée, c'est l'ordre alphabétique qui est appliqué. C'est le cas dans notre exemple, pour établir la variable nommée **croissant**. Par contre, pour utiliser l'ordre décroissant, il faut passer par une autre fonction renvoyant un résultat à partir d'un test. Ici, **n** n'étant pas supérieur à **m**, c'est la valeur -1 qui est retournée et qui permet de donner l'ordre de tri à la méthode **sort()**.

La méthode **sort()** s'applique, par défaut, sur des éléments de type alphanumérique. Pour obtenir un tri d'éléments numériques, il faut passer une autre fonction comme fonction de comparaison.

Attention : "55" est plus petit que "6".

Exercice 13 : afficher, dans un document HTML, tout d'abord les deux premiers jours d'une semaine de **travail** (lundi et mardi), et sur une ligne suivante les deux derniers jours (jeudi et vendredi).

Les deux premiers jours de la semaine de travail : lundi,mardi
Les deux derniers jours de la semaine de travail : jeudi,vendredi

La méthode **splice()** utilise plusieurs arguments afin d'extraire les éléments d'un tableau. Attention, la partie extraite est supprimée. La syntaxe est la suivante :

nouveautableau=tableau.splice(i,j) ;

Où ***i*** correspond à l'indice du premier élément concerné par l'application de la méthode et ***j*** correspond au nombre d'éléments à extraire. Le tableau d'origine est modifié.

La suppression des deux premiers éléments du tableau réaffecte les numéros indices (Mercredi reprend l'indice zéro après la première suppression (tranche 1)). Idem couper-coller

Exercice 14 : Créez un tableau de nombres entiers. Le remplir de plusieurs valeurs (lors de la création).

Recherchez et affichez l'indice du plus petit nombre et du plus grand nombre.

Recherchez et affichez l'indice d'un nombre saisi au clavier. Si le nombre n'existe pas affichez un message correspondant.

Affichez le nombre de valeurs égales à zéro.

Calculez et affichez la somme des éléments du tableau.

Refaire le même travail avec un tableau de nombres réels.

Exemples de tableaux associatifs :

```
<!-- tableaux associatifs-->

<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <script>

      var personne = new Array();
      personne.push({id:"11",nom:"DUPOND",age:"20"});
      personne.push({id:"12",nom:"DURAND",age:"22"});
      indice=0;
      document.write( "NOM : " +personne[indice].nom+
                      " id :"+personne[indice].id+"<br>");

      indice=1;
      document.write( "NOM : " +personne[indice].nom+
                      " id :"+personne[indice].id+"<br>");

      var tab={"un":1,"deux":2,"trois":3};

      document.write( "tab pour un : " +tab["un"]+ "<br>");
      document.write( "tab pour deux : " +tab["deux"]);

      /* Résultat :
         NOM : DUPOND id :11
         NOM : DURAND id :12
         tab pour un : 1
         tab pour deux : 2
      */
    </script>
  </body>
</html>
```

Autre exemple :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tableau associatif</title>
  </head>
  <body>

    <script>
      var tab={"un":1,"deux":2,"trois":3};
      var x=tab["un"];
      document.write( "x vaut : " + x + "<br>"); // Affiche x vaut : 1
      for (var maCle in tab)
      {
        var valeur=tab[maCle];
        document.write( maCle + " = " + valeur + "<br>"); // affiche un=1
                                                         // deux=2
                                                         // trois=3
      }
    </script>
  </body>
</html>
```

Les tableaux multidimensionnels

Lorsque les données à stocker sont nombreuses, il est possible d'utiliser les tableaux à plusieurs entrées, aussi appelés multidimensionnels.

La construction de ce type de tableau nécessite la déclaration des deux tableaux imbriqués. Vous déclarez d'abord, un tableau de façon classique. Puis, vous réalisez l'affectation des valeurs en fonction de leur position (colonne, ligne) dans le tableau.

La syntaxe peut se résumer ainsi pour un tableau de 2 lignes et 3 colonnes :

```
var tableau = new Array(nombre de lignes du tableau);
tableau[0] = new Array(nombre de colonnes du tableau);
tableau[1] = new Array(nombre de colonnes du tableau);
tableau[0][0] = "valeur1";
tableau[0][1] = "valeur2";
tableau[0][2] = "valeur3";
tableau[1][0] = "valeur1";
tableau[1][1] = "valeur2";
tableau[1][2] = "valeur3";
```

Exemple : afficher, dans une page HTML, les réflexions enregistrées du mercredi et du vendredi, toutes ces réflexions ayant été enregistrées préalablement, pour chaque jour de la semaine, dans un tableau à deux dimensions.

Voici le mot du mercredi : Jour des enfants

Voici le mot du vendredi : C'est le week-end

```
<script language="javascript">
  var semainier = new Array(5);
  semainier[0] = new Array(2);
  semainier[1] = new Array(2);
  semainier[2] = new Array(2);
  semainier[3] = new Array(2);
  semainier[4] = new Array(2);
  semainier[0][0] = "lundi";
  semainier[0][1] = "Au boulot";
  semainier[1][0] = "mardi";
  semainier[1][1] = "Ca va fort";
  semainier[2][0] = "mercredi";
  semainier[2][1] = "Jour des enfants";
  semainier[3][0] = "jeudi";
  semainier[3][1] = "Encore un effort";
  semainier[4][0] = "vendredi";
  semainier[4][1] = "C'est le week-end";
  alert("Voici le mot du mercredi : " + semainier[2][1]);
  alert("Voici le mot du vendredi : " + semainier[4][1]);
</script>
```

```

<!-- Exemple_dim2 -->
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
    <script>

      var tableau = new Array("lundi", "mardi", "mercredi", "jeudi",
                              "vendredi", "samedi", "dimanche");

      document.write(" <br> Les jours de la semaine : <br> <br>");
      document.write(" <table Border=5> <tr>\n\
        <th> Indice </th>\n\
        <td Align=CENTER>" + 0 + "</td>" + "<td Align=CENTER>" + 1 + "</td>\n\
        <td Align=CENTER>" + 2 + "</td>" + "<td Align=CENTER>" + 3 + "</td>\n\
        <td Align=CENTER>" + 4 + "</td>" + "<td Align=CENTER>" + 5 + "</td>\n\
        " + "<td Align=CENTER>" + 6 + "</td>\n\
        </tr>\n\
        <th> Valeur </th>\n\
        <td> " + tableau[0] + " </td> " + "<td> " + tableau[1] + " </td>\n\
        <td> " + tableau[2] + " </td> " + "<td> " + tableau[3] + " </td>\n\
        <td> " + tableau[4] + " </td> " + "<td> " + tableau[5] + " </td>\n\
        <td> " + tableau[6] + " </td> \n\
        </tr></table>");
    </script>
  </body>
</html>

```

Les jours de la semaine :

| Indice | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|--------|-------|-------|----------|-------|----------|--------|----------|
| Valeur | lundi | mardi | mercredi | jeudi | vendredi | samedi | dimanche |

Exercices : reprendre le programme ci-dessus en utilisant un indice « i », puis afficher ce même tableau sous forme verticale.

Exercice 15 : Créez un tableau à 2 dimensions de 4 lignes et de 5 colonnes. Le remplir de plusieurs valeurs entières comme dans l'exemple ci dessous.

L'afficher en utilisant des balises html.

Calculez et affichez la somme des éléments du tableau.

Exemple de résultat :

| | | | | |
|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

La somme est : 210

Exercice 16 : Mini tableur.

Réaliser un tableau à 2 dimensions de 4 lignes et de 5 colonnes.

Remplir (via le clavier) les 3 premières lignes et les 4 premières colonnes.

Additionner les colonnes en dernière ligne et les lignes en dernière colonne.

Calculer le total dans la dernière case.

Exemple de résultat :

| | | | | |
|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 10 |
| 5 | 6 | 7 | 8 | 26 |
| 9 | 10 | 11 | 12 | 42 |
| 15 | 18 | 21 | 24 | 78 |

VIII Comment manipuler les dates ?

Objectifs

- Savoir utiliser l'objet *Date*
- Savoir utiliser les systèmes de minuterie
- Utiliser le site : <http://toutjavascript.com>

L'objet Date

L'objet *Date* est un objet inclus de façon native dans JavaScript, c'est-à-dire que vous pouvez l'appeler à tout moment. Il permet de gérer le calcul du temps jusqu'à une précision de la milliseconde. Les dates sont représentées par un nombre compris entre -100 000 000 et +100 000 000 avec comme référence le 1^{er} janvier 1970. L'objet *Date* n'a pas de propriété. Pour manipuler l'objet *Date*, il faut créer une instance de celle-ci dans une variable. Par contre, l'objet *Date* possède trois types de méthode pour lire, écrire ou convertir. Il existe en plus un système de minuterie permettant de gérer les intervalles de temps. À noter toutefois, que le terme d'écriture ne correspond pas au fait de changer l'heure système, qui est tout à fait impossible.

```
var d = new Date();
```

Les méthodes de lecture

| Méthode | Résultat | Exemple |
|----------------------------|--|------------------------------|
| <i>getDate()</i> | Renvoie le numéro du jour du mois entre 1 et 31. | <i>d.getDate()</i> ; |
| <i>getDay()</i> | Renvoie le numéro du jour de la semaine sachant que zéro correspond au dimanche et six au samedi. | <i>d.getDay()</i> ; |
| <i>getFullYear()</i> | Renvoie un nombre de quatre chiffres correspondant à l'année en cours. | <i>d.getFullYear()</i> ; |
| <i>getHours()</i> | Renvoie l'heure courante sachant que zéro correspond à minuit. | <i>d.getHours()</i> ; |
| <i>getMilliseconds()</i> | Renvoie un nombre en millisecondes correspondant à l'heure locale. | <i>d. getMilliseconds()</i> |
| <i>getMinutes()</i> | Renvoie le nombre de minutes de l'heure courante. | <i>d.getMinutes()</i> |
| <i>getMonth()</i> | Renvoie le numéro du mois courant sachant que zéro correspond à janvier et onze à décembre. | <i>d.getMonth()</i> |
| <i>getSeconds()</i> | Renvoie le nombre de secondes de l'heure courante. | <i>d.getSeconds()</i> |
| <i>getTime()</i> | Renvoie le nombre de millisecondes écoulées depuis le 1er janvier 1970. | <i>d.getTime()</i> |
| <i>getTimezoneOffset()</i> | Renvoie le nombre de minutes séparant le lieu de l'exécution du script, du méridien de Greenwich. | <i>d.getTimezoneOffset()</i> |
| <i>getUTCDate()</i> | Renvoie le nombre du jour du mois exprimé en coordonnées UTC (<i>Temps Universel Coordonné</i>). | <i>d.getUTCDate()</i> |

| | | |
|------------------------------------|--|--|
| <i>getUTCDay()</i> | Renvoie le numéro du jour de la semaine exprimé en coordonnées UTC. | |
| <i>getUTCFullYear()</i> | Renvoie un nombre de quatre chiffres correspondant à l'année en cours, en coordonnées UTC. | |
| <i>getUTCHours()</i> | Renvoie l'heure courante exprimée en coordonnées UTC. | |
| <i>getUTCMilliseconds()</i> | Renvoie le nombre de millisecondes de l'heure courante exprimé en coordonnées UTC. | |
| <i>getUTCMonth()</i> | Renvoie le numéro du mois courant exprimé en coordonnées UTC. | |
| <i>getYear()</i> | Renvoie un nombre de deux chiffres pour les années antérieures à l'an deux mille et de quatre pour les années postérieures (exemple 99 pour 1999 et 2008 pour 2008). | |

Exercice 1 : Finissez de remplir la colonne **Exemple** du tableau ci-dessus.

Exemple

```
<script type="text/javascript">
var ladata=new Date()
document.write("Nous sommes le : ");
document.write(ladata.getDate()+"/"+(ladata.getMonth()+1)+"/"+ladata.getFullYear())
</script>
```

Résultat : Nous sommes le : 21/9/2020

Exercice 2 : afficher la date courante dans un document HTML. Sous le format jj/mm/aaaa, comme par exemple *21/09/2020* Puis *21 Septembre 2020* et pour finir *Dimanche 21 Septembre 2020*.

La première étape consiste à créer un objet **Date** permettant de récupérer la date du jour. Ensuite, il est possible de récupérer le jour, le mois et l'année. Une première particularité réside dans la gestion du numéro du jour. En effet pour améliorer l'affichage, il est préférable d'ajouter 0 devant les numéros inférieurs à 10. Une autre particularité se situe dans la gestion des numéros des mois, car étant donné que la numérotation des mois de l'année débute (comme toujours en JavaScript) à zéro, il convient d'ajouter 1 à la variable **mois** pour l'affichage dans le document.

Les mois s'affichent en nombre. Aussi, si vous souhaitez les obtenir en lettres, vous pouvez utiliser une structure de contrôle **switch** pour les déterminer en fonction de la variable **mois**. Le même genre d'instruction peut être utilisé pour déterminer le **jour de la semaine** avec la méthode **getDay()**.

Les méthodes d'écriture

d = new Date();

| Méthode | Résultat | Exemple |
|------------------------------------|--|------------------------------------|
| <i>setDate()</i> | Fixe la valeur du jour du mois. (de 1 à 31) | <i>d.setDate(30);</i> |
| <i>setFullYear()</i> | Fixe la valeur de l'année. (4 chiffres) | <i>d.setFullYear(2012);</i> |
| <i>setHours()</i> | Fixe la valeur de l'heure. | |
| <i>setMilliseconds()</i> | Fixe la valeur de la milliseconde. | |
| <i>setMinutes()</i> | Fixe la valeur des minutes. | |
| <i>setMonth()</i> | Fixe la valeur du numéro du mois. (de 0 à 11) | |
| <i>setSeconds()</i> | Fixe la valeur des secondes. | |
| <i>setTime()</i> | Fixe la date. | |
| <i>setUTCDate()</i> | Fixe la date selon UTC. | |
| <i>setUTCFullYear()</i> | Fixe la valeur de l'année selon UTC. | |
| <i>setUTCHours()</i> | Fixe la valeur entre 0 et 23 de l'heure selon UTC. | |
| <i>setUTCMilliseconds()</i> | Fixe la valeur entre 0 et 999 des millisecondes selon UTC. | |
| <i>setUTCMinutes()</i> | Fixe la valeur entre 0 et 59 des minutes selon UTC. | |
| <i>setUTCMonth()</i> | Fixe la valeur entre 0 et 11 du mois selon UTC. | |
| <i>setUTCSeconds()</i> | Fixe la valeur entre 0 et 59 des secondes selon UTC. | |
| <i>setYear()</i> | Fixe la valeur de l'année. | |

setFullYear() fixe la valeur de l'année, mais peut également fixer l'année, le mois et le jour du mois. Par exemple :

```
var dateNaissance = new Date();
```

```
dateNaissance.setFullYear(1988, 11, 6); // 06/12/1988 car mois + 1
```

setFullMonth() fixe la valeur du mois, mais peut également fixer le mois et le jour du mois. Par exemple :

```
var dateNaissance = new Date();
```

```
dateNaissance.setFullMonth(11, 6); // 06/12 car mois + 1
```

Exercice 3 : Remplissez la colonne **Exemple** du tableau ci-dessus.

Les méthodes de conversion

| Méthode | Résultat | Exemple |
|------------------|---|---------|
| parse() | Convertit un objet Date en nombre de millisecondes depuis le 1er janvier 1970. | |
| toLocaleString() | Convertit l'objet Date en une chaîne de caractères selon le format local. | |
| toString() | Convertit l'objet Date en une chaîne de caractères. | |
| toUTCString() | Convertit l'objet Date en une chaîne de caractères selon le format UTC. | |
| UTC() | Convertit en millisecondes la différence entre une date et le 1er janvier 1970. | |

Exercice 4 : Remplissez la colonne **Exemple** du tableau ci-dessus.

Exercice 5 : À partir de sa date de naissance, calculez et affichez l'âge d'un individu ainsi qu'un message de félicitations si l'anniversaire correspond à la date du jour. Refaites le même programme à partir d'une date de naissance saisie par l'utilisateur grâce à **window.prompt()**.

Les minuteries

Il existe quatre méthodes particulières de l'objet window qui permettent de gérer une minuterie :

| Méthode | Résultat | Exemple |
|------------------------|---|---------|
| <i>setInterval()</i> | Déclenche l'exécution d'une série d'instructions en fonction d'une fréquence en millisecondes passée en argument. | |
| <i>clearInterval()</i> | Stoppe la minuterie lancée par <i>setInterval()</i> . | |
| <i>setTimeout()</i> | Retarde le déclenchement d'une série d'instructions en fonction d'une durée en millisecondes passée en argument. | |
| <i>clearTimeout()</i> | Stoppe la minuterie lancée par <i>setTimeout()</i> . | |

La syntaxe de *setInterval()* est la suivante :

setInterval(action, période, arguments éventuels de l'action) ;

setInterval() est parfois mal reconnu, notamment par les navigateurs dont la version est un peu ancienne. Il est préférable d'utiliser *SetTimeout()* lorsque vous en avez le choix.

La syntaxe de *clearInterval()* est la suivante :

clearInterval(nom de la minuterie) ;

La syntaxe de *setTimeout()* est la suivante :

setTimeout(action,délai, arguments éventuels de l'action) ;

La syntaxe de *clearTimeout()* est la suivante :

clearTimeout(nom de la minuterie) ;

Exercice 6 : créer une horloge simple affichant les heures, les minutes et les secondes dans un tableau :

| | |
|---------------------|---------------------------------------|
| Il est exactement : | <input type="text" value="11:08:37"/> |
|---------------------|---------------------------------------|

Pour réussir à manipuler l'objet *Date*, il faut créer une nouvelle instance de cet objet dans une fonction que vous nommerez *Horloge()*. Puis, il est possible d'en extraire les heures, les minutes et les secondes. Ensuite, il faut traiter l'affichage pour les valeurs inférieures à 10 et leur ajouter un zéro devant si nécessaire, pour obtenir 01 à la place de 1, par exemple. Ensuite, il est possible d'afficher le contenu du champ de formulaire et de demander le rechargement de la fonction *Horloge()* au bout de 1000 millisecondes (soit une seconde). Ainsi, l'horloge se modifiera toutes les secondes.

Exercice complet : Reprendre les Exercices 2, 5 et 6 sur une seule page.

IX Comment manipuler les chaînes de caractères ?

Objectifs

- Savoir utiliser l'objet String
- Extraire une sous-chaîne d'une chaîne
- Rechercher un caractère dans une chaîne
- Utiliser le site : <http://toutjavascript.com>

L'objet String

L'objet **String** est un objet qui permet de manipuler les chaînes de caractères ; c'est pour cela qu'il est très souvent utilisé que ce soit pour la recherche, la concaténation ou l'extraction de sous chaînes de caractères. Il n'est donc pas étonnant de constater que l'objet **String** dispose de nombreuses propriétés et méthodes.

a. Les propriétés

| Propriété | Résultat | Exemple |
|--------------------|---|---|
| constructor | Indique comment a été créé un objet référencé. | <code>var strChaine = new String("Bonjour");</code> |
| length | Correspond à la longueur d'une chaîne caractères. | <code>document.write (strChaine.length) ;</code> |
| prototype | Permet d'ajouter des propriétés à un objet. | |

b. Les méthodes

| Méthode | Résultat | Exemple |
|---------------------|---|---|
| anchor() | Détermine une ancre nommée dans un document HTML. | |
| big() | Augmente d'un niveau la taille de la police utilisée dans un document HTML. | <code>document.write (strChaine.big ());</code> |
| blink() | Permet de faire clignoter le texte dans un document HTML. | <code>document.write (strChaine.blink ());</code> |
| bold() | Affiche une chaîne de caractères en gras. | <code>document.write (strChaine.bold ());</code> |
| charAt() | Permet d'accéder à un seul caractère dans une chaîne complète. | <code>document.write (strChaine.charAt());</code> |
| charCodeAt() | Renvoie la valeur en décimal du caractère indiqué en argument. | <code>document.write (strChaine.charCodeAt ("B"));</code> |

| | | |
|-----------------|--|--|
| concat() | Concatène plusieurs chaînes de caractères. | |
|-----------------|--|--|

| | | |
|-----------------------|--|--|
| eval() | Convertit une chaîne de caractères en code de programme JavaScript. | |
| fixed() | Affiche une chaîne de caractères avec une police à pas fixe. | |
| fontcolor() | Modifie la couleur de la police de caractères d'une chaîne. | |
| fontSize() | Modifie la police de caractères d'une chaîne. | |
| fromCharCode() | Renvoie une chaîne de caractères constituée à partir de représentations isolées d'une suite de caractères Unicode. | |
| indexOf() | Renvoie la position, à partir de la gauche, d'un caractère ou d'une sous-chaîne passé en argument dans une chaîne de caractères. | |
| italics() | Affiche une chaîne de caractères en italique. | |
| lastIndexOf() | Renvoie la position, à partir de la droite, d'un caractère passé en argument dans une chaîne de caractères. | |
| link() | Demande le chargement d'une URL par le navigateur. | |
| match() | Recherche une expression régulière dans une chaîne de caractères. | |
| replace() | Remplace une expression régulière dans une chaîne de caractères. | |
| search() | Effectue une recherche dans une chaîne de caractères. | |
| slice() | Extrait une sous chaîne de caractères, en fonction d'un caractère initial et d'un caractère final. | |

| | | |
|----------------------|--|--|
| small() | Diminue d'un niveau la taille de la police utilisée dans un document HTML. | |
| split() | Découpe une chaîne de caractères en fonction d'un séparateur passé en argument. | |
| strike() | Affiche une chaîne de caractères en mode barré. | |
| sub() | Affiche une chaîne de caractères en indice. | |
| substr() | Renvoie une sous-chaîne de caractères, en fonction d'une position et d'une longueur passées en argument. | |
| substring() | Renvoie une sous-chaîne de caractères, en fonction d'une position passée en argument. | |
| sup() | Affiche une chaîne de caractères en exposant. | |
| toLowerCase() | Convertit en minuscules une chaîne de caractères. | |
| toString() | Tente de convertir un objet en chaîne de caractères. | |
| toUpperCase() | Convertit en majuscules une chaîne de caractères. | |

Exercice 1 : En utilisant le site : <http://toutjavascript.com>, continuez à remplir la colonne **Exemple** avec des exemples personnels.

Exercice 2 : Affichez dans une page la chaîne de caractères « Bonjour » saisie avec la méthode **window.prompt()** selon différents modes d'affichage (gras, italique, clignotant, rouge, caractères barrés et de 7 tailles différentes) :

Voici la chaîne de caractères en gras : **Bonjour**

Voici la chaîne de caractères en italique : *Bonjour*

Voici la chaîne de caractères en caractères clignotants : Bonjour

Voici la chaîne de caractères en rouge : **Bonjour**

Voici la chaîne de caractères barrée : ~~Bonjour~~

Voici la chaîne de caractères en différentes tailles :

Bonjour

Bonjour

Bonjour

Bonjour

Bonjour

Bonjour

Explications : Ici, la plupart des méthodes liées à l'objet **String** sont passées en revue. Ces méthodes doivent être utilisées avec parcimonie, car elles ne respectent pas le standard du W3C. Il vaut mieux leur préférer une mise en forme à l'aide des CSS (Cascade Sheet Style ou feuilles de style en cascade). Néanmoins, le script présente les fonctionnalités disponibles et il débute par une série d'affichage utilisant différentes propriétés de texte, puis se poursuit avec une boucle permettant d'afficher le texte avec une taille de plus en plus grande, à chaque passage

de 1 à 7 correspondant aux tailles HTML définies par le W3C (World Wide Web Consortium : organisme de standardisation des technologies de l'Internet).

Exercice 3 : Afficher la longueur d'une chaîne de caractères saisie avec la méthode **window.prompt()** et convertir en majuscules puis en minuscules le contenu de cette chaîne que vous afficherez dans le document. La même chaîne de caractères sera ensuite convertie avec une majuscule pour débiter (1^{ère} lettre) et le reste en minuscules.

Exercice 4 : Extraire une chaîne de caractères d'une autre chaîne saisie par l'utilisateur en fonction d'un début et d'une fin d'extraction. Le résultat est affiché dans la page et avec des caractères de couleur rouge :



Explications : Le script déclare et initialise trois variables pour débiter :

- La variable **texte** correspond au texte saisi par l'utilisateur.
- La variable **pos1** correspond à la position du premier caractère de la sous-chaîne à extraire.
- La variable **pos2** correspond à la position du dernier caractère de la sous-chaîne à extraire.

Ensuite, il suffit d'appliquer la méthode **substring ()** à la chaîne de caractères pour extraire une sous-chaîne à partir de la position du premier et du dernier caractère. La chaîne est, ensuite, convertie en rouge et affichée dans le document.

Exercice 5 : Effectuer un test sur une extension de fichier saisie avec la méthode **window.prompt()**. Si l'extension n'est pas **gif** ou **jpg**, alors un message d'avertissement s'affiche, sinon un message positif s'affiche.

Explications : Ici, le script s'exécute, l'utilisateur saisit le nom du fichier à transférer. À ce moment, le script récupère le nom du fichier saisi et le stocke dans une variable appelée **fichier**. Ensuite, il calcule la longueur de la chaîne de caractères correspondant au nom du fichier. Étant donné que l'extension comporte trois caractères, vous en déduisez que la chaîne à extraire par la méthode **substr ()** débute au nombre de caractères moins 3. La chaîne de caractères extraite est, ensuite, testée pour savoir si elle correspond à l'une ou l'autre des extensions autorisées. À ce moment-là, il est possible d'afficher l'un ou l'autre des messages dans une boîte de dialogue.

Exercice 6 : Tester la valeur d'une chaîne de caractères saisie avec la méthode **window.prompt()** pour savoir s'il contient une arobase et un point. En cas d'échec, afficher « Adresse invalide », en cas de succès, afficher « Adresse valide ».

Explications : Ce script peut être utilisé pour contrôler si la saisie d'une adresse e-mail a été bien complétée. Malgré tout, il ne permet pas une vérification totale puisque la présence d'une arobase et d'un point ne suffit pas à valider d'une façon certaine une adresse e-mail. Ce script utilise la méthode **IndexOf ()** de l'objet string qui permet de connaître la position d'un caractère dans une chaîne. Sa syntaxe est la suivante :

mail.indexOf(recherche, début) ;

où **mail** représente la chaîne dans laquelle effectuer la recherche, **recherche** correspond à la chaîne à rechercher et **début** à la position du caractère à partir duquel il faut commencer la recherche. L'argument **début** indique le caractère de début de la recherche. S'il est omis, la recherche commence au début de la chaîne.

Si la position retournée par la fonction est inférieure à zéro (étant donné que la position du premier caractère correspond à zéro), c'est que le caractère recherché n'est pas présent dans le champ. Un message d'avertissement sera alors renvoyé, sinon c'est un message de validité qui sera proposé à l'utilisateur.

Attention, si une des positions est retournée inférieure à 0, l'adresse n'est pas valide.

D'autres méthodes peuvent être employées pour vérifier ce type de champ comme les expressions régulières, qui seront expliquées plus tard.

Exercice 7 : Ecrire un script qui permette d'afficher la chaîne de caractères : « **État** » en gras et en rouge. Vous utiliserez la méthode **fromCharCode ()** pour écrire le caractère « **É** »

X Regex pour valider des formulaires HTML

Objectifs

- Savoir écrire un motif ou pattern avec ses options
- Utiliser les expressions régulières pour valider un formulaire HTML

Déclarer un objet RegExp

Il existe 2 manières pour déclarer un objet RegExp. (Les 2 notations sont équivalentes.)

1. La plus lisible est d'utiliser le constructeur RegExp. :

```
var exp = new RegExp(pattern,option) ;
```

Ici, exp devient un objet de type RegExp.

- `pattern` est le motif associé, ce qui permet de faire le traitement. Son format est défini à partir du tableau ci-après.
- `option` est une chaîne de caractères qui peut prendre les valeurs :
 - `""`, vide, aucune option,
 - `"g"`, recherche globale sur toute la chaîne,
 - `"i"`, ignore la casse
 - `"gi"`, association de g et i.

```
Exemple :    var nom ;
              var exp = /^A/ ; // ou var exp = RegExp (^A) ;
              nom = "Alain" ;
              var resultat = exp.text(nom);
              if (resultat === true)
              {
                  document.write (nom);
              }
```

2. Une autre notation, plus compacte, mais moins lisible peut être utilisée :

```
var exp=/pattern/option ;
```

Les motifs et leur signification

Voici un listing des syntaxes de motif pour les expressions régulières. À noter que ces syntaxes sont valables autant pour le Javascript que pour le PHP ou encore les fichiers « .htaccess » (fichier de configuration pour Apache).

| Élément | Signification |
|-----------------|---|
| ^ | Indique le début de ligne ou de chaîne (début du motif) |
| \$ | Indique la fin de motif ou de chaîne (fin du motif) |
| . | Autorise n'importe quel caractère |
| [abc] | Recherche <u>exactement</u> la chaîne entre crochet |
| [a-z] | Ne peut contenir que des caractères compris entre a et z (en minuscule) |
| [A-Z] | Ne peut contenir que des caractères compris entre A et Z (en majuscule) |
| [a-zA-Z] | Ne peut contenir que des caractères compris entre a et z (en minuscule ou majuscule) |
| [0-9] | Ne peut contenir que des caractères numériques compris entre 0 et 9 |
| [^0-9] | Ne peut contenir les caractères numériques compris entre 0 et 9 (donc ici, aucun caractère numérique) |
| (y) | Recherche l'expression parenthésée |
| * | Peut contenir le motif précédemment déclaré de 0 à x fois |
| [a-r]* | Peut contenir des lettres comprises entre a et r de 0 à x fois |
| + | Peut contenir le motif précédemment déclaré de 1 à x fois |
| ? | Peut contenir le motif précédemment déclaré de 0 à 1 fois |
| {n} | Doit contenir le motif précédemment déclaré exactement n fois |
| {n,} | Doit contenir le motif précédemment déclaré au moins n fois |
| {n,m} | Doit contenir le motif précédemment déclaré de n à m fois |
| \ | Caractère d'échappement |
| \\ | Caractère \ |
| \d | Chiffre (équivalent à [0-9]) |
| \D | Aucun chiffre (équivalent à [^0-9]) |
| \b | Frontière / séparateur de mot (espace, alinéa, ...) |
| \s | Caractère d'espacement (espace, tabulation, saut de page, ...) – équivalent à [\f\n\r\t\v] |
| \S | Un seul caractère sauf un espacement |
| \w | N'importe quel caractère alphanumérique dont l'underscore « _ » (équivalent à [a-zA-Z0-9_]) |
| \W | Tout sauf un caractère alphanumérique (équivalent à [^a-zA-Z0-9_]) |

Quelques exemples

Contrôle de login (entre 3 et 8 caractères alphanumériques)

le pattern vaudra :

`^[a-zA-Z0-9]{3,8}$`

La signification est :

- `^`: pour indiquer le début de chaîne,
- `[a-zA-Z0-9]`: pour les caractères alphanumériques de a à z, de A à Z et de 0 à 9,
- `{3,8}`: pour indiquer qu'il faut une longueur de 3 à 8 caractères alphanumériques,
- `$` : pour indiquer la fin de chaîne.

Valider une adresse mail

Le pattern vaudra (sous réserve de mieux) :

`^[A-Za-z0-9\._-]+@([A-Za-z0-9\._-]+)([A-Za-z0-9]){2,4}$`

Exemple accepté : `aaa.aa@ddd_mm.12`

La signification de cette chaîne est :

- `^` pour indiquer le début du mail,
- `[A-Za-z0-9\._-]+` Au moins un caractère alphanumérique en début de mail, point et Under score compris
- `@` Une fois arobase,
- `([A-Za-z0-9\._-]+)` tous les caractères alphanumériques avec - et _ , le tout au moins 1 fois, suivis du caractère «.»
- `([A-Za-z0-9]){2,4}$` tous les caractères alphanumériques au moins 2 fois et au plus 4 fois en fin de chaîne

Exemple :

```
<!DOCTYPE html>
<!-- FORMULAIRE ET REGEX
En utilisant les expressions régulières vous testerez la validité de la saisie
d'un article.
Pour être valide, un article doit avoir les 2 premiers caracteres compris
entre A et Z en majuscule et les 4 caractères suivants compris entre 0 et
9 uniquement.
Au total il doit y avoir 6 caractères ni plus ni moins.
Le bouton "Vérifier l'article" permet de visualiser la validation grâce à
une boîte de dialogue "alert".
-->
<html>
  <head>
    <title>Verifier article avec regex</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <script>
      function VerifArticle(reference)
      {
        var exp=new RegExp("[A-Z]{2}[0-9]{4}$","g");
        if (exp.test(reference)===true)
        {
          alert ("La référence [" +reference +"] est correcte :)");
        }
        else
        {
          alert ("La référence[" +reference +"]n'est pas correcte:)");
        }
      }
    </script>
    <form>
      Article : <input type="text" name="article" size="8" value="">
      <INPUT type=button value="Vérifier l'article"
              onClick="VerifArticle(this.form.article.value)"><br>
    </form>
  </body>
</html>
```

Les différentes actions des RegExp

L'objet RegExp possède plusieurs méthodes :

- La méthode **test()** qui valide une chaîne de caractères en fonction d'un motif , dans les exemples vers fin du document.(retourne true en cas de réussite, sinon retourne false)
- La méthode **exec()** retourne la première sous-chaîne (occurrence) correspondant au motif.

```
<script>
  var reg = new RegExp("[0-9]+", "g"); // motif et option
  var chaine = "Il y a 3600 secondes dans 1 heure";
  document.write(chaine + "</br>"); // affiche : Il y a 3600 secondes dans 1
heure
  document.write(reg.exec(chaine)); // affiche : 3600
</script>
```

Sur l'objet String, il existe également des méthodes qui attendent en paramètre une expression régulière :

- La méthode **search()** permet de trouver des occurrences répondant aux critères du motif.

```
<script type = "text/javascript" >
  var str = "Visit W3Schools!";
  document.write(str.search(/W3SCHOOT/i) + "<BR>");//ne trouvera pas, renvoie -1
  document.write(str.search(/W3SCHOOL/i)); //trouvera, renvoie 6, c'est la
position du W
</script>
```

- La méthode **replace()** permet de trouver et de remplacer des occurrences répondant aux critères du motif.

```
<SCRIPT language=javascript>
  var chaine = "Les Chiens et les chiennes, les chats et les oiseaux";
  var reg = new RegExp("[c|C]", "g");
  document.write("Chaîne d'origine : " + chaine + "<BR>");
  document.write("Chaîne traitée : "+chaine.replace(reg, "Z")+"<BR>");
</SCRIPT>
```

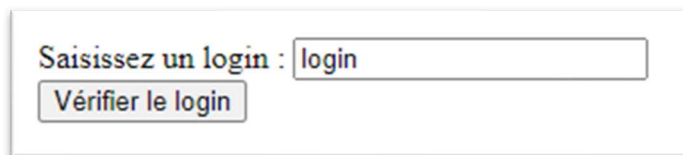
- La méthode **match()** permet de trouver des occurrences répondant aux critères du motif.

```
<SCRIPT language=javascript>
  var chaine = "Le corbeau et le renard";
  var reg1 = new RegExp("[cor]", "g");
  if (chaine.match(reg1)) {
    document.write("Le motif [cor] a été trouvé dans la chaîne");
  }
</SCRIPT>
```

Applications aux formulaires HTML

Exemple 1 : Vérifier un login (Formulaire avec « button » et « onClick »)

```
<script type="text/javascript">
function VerifLogin(saisie)
{
    var exp = new RegExp("[a-zA-Z0-9]{3,8}$", "g");
    if (exp.test(saisie)) // applique le filtre à la variable
    {
        alert("Le login [" + saisie + "] est valide :)");
    }
    else
    {
        alert("Le login [" + saisie + "] n'est pas valide !!!!");
    }
}
</script>
<FORM>
    Saisissez un login : <INPUT type=text name=login value="login"><BR>
    <INPUT type=button value="Vérifier le login" onClick="VerifLogin(this.form.login.value)">
</FORM>
```



Exemple 2 : Vérifier un login (Formulaire avec « submit »)

L'attribut **onSubmit** est surtout utilisé pour contrôler les saisies via le langage JavaScript avant d'envoyer ou pas les données au serveur.

```
<script type="text/javascript">
function VerifLogin(saisie) {
    var exp = new RegExp("[a-zA-Z0-9]{3,8}$", "g");
    if (exp.test(saisie))
    {
        alert("Le login [" + saisie + "] est valide :)");
        return true;
    } else
    {
        alert("Le login [" + saisie + "] n'est pas valide !!!!");
        return false;
    }
}
</script>

<FORM onSubmit="return VerifLogin(this.login.value);">
    Saisissez un login : <INPUT type=text name=login value="login">
    <INPUT type=submit value="Envoyer le login" >
</FORM>
```

Exemple 3 : Vérifier un mot de Passe

Le mot de passe doit être robuste, il doit comprendre au moins 7 caractères.

Parmi ces caractères, il doit y avoir un chiffre au moins, ainsi qu'une majuscule au moins et un caractère de ponctuation

```
<script type="text/javascript">
    function VerifPassword(saisie) {
        // le mot de passe doit comprendre au moins 7 caractères.
        // Parmi ces caractères, il doit y avoir un chiffre au moins,
        // ainsi qu'une majuscule au moins et un caractères de ponctuation
        var exp1 = new RegExp("^[a-zA-Z0-9\? , ; ! : ] {7,} $", "g");
        var exp2 = new RegExp("[0-9]+");
        var exp3 = new RegExp("[A-Z]+");
        var exp4 = new RegExp("[\? , ; ! \ $ : ]+");
        if (exp1.test(saisie)
            && exp2.test(saisie)
            && exp3.test(saisie)
            && exp4.test(saisie)) {
            alert("Le password [" + saisie + "] est valide !!!!");
            return true;
        } else {
            alert("Le password [" + saisie + "] n'est pas valide
            !!!!");
            return false;
        }
    }
</script>

<FORM onSubmit="return VerifPassword (this.password.value);">
Saisissez votre mot de passe : <INPUT type="password" name="password">
<INPUT type=submit value="Tester le mot de passe" >
</FORM>
```

Exemple 4 : Vérifier un format de date

```
<html>
<head>
  <title>Test de validité d'une date</title>
  <script type="text/javascript">
    function testDate(date) { //format 20/10/2010
      var model = /^(^([0-9]{2})\/([0-9]{2})\/([0-9]{4}))$/;
      // $ signifie : pas de caractères après le 8ème chiffre

      if (model.test(date)) {
        alert("ok");
        return true;
      } else {
        alert("erreur");
        return false;
      }
    } // fin function
  </script>
</head>
<body>

  <form method="POST" action="" onsubmit="return testDate(this.date.value)">
    Date de naissance : <input type="text" name="date" size="50"><br>
    <input type="submit" value="Envoyer" >
  </form>
</body>
</html>
```

Amélioration : `/^([0-2][0-9]|(3)[0-1])(\V)((((0)[0-9])|((1)[0-2]))(\V)\d{4})$/ ;`

Exemple 5 : validation d'un champ devant commencer par une majuscule

```
<html>
<head>
  <title>Test validité d'un nom</title>
  <script type="text/javascript">
    function testNom(nom) {
      //format doit commencer par une majuscule : Hector
      var model = /^[A-Z]/;
      if (model.test(nom)) {
        alert("ok");
        return true;
      } else {
        alert("erreur");
        return false;
      }
    }
  </script>
</head>
<body>
  <form method="POST" action="" onsubmit="return testNom(this.nom.value)">
    Nom : <input type="text" name="nom" size="50"><br>
    <input type="submit" value="Envoyer" >
  </form>
</body>
</html>
```

Exemple 6 : validation d'un numéro de téléphone

```
<html>
<head>
  <title>Test de validité d'un numéro de téléphone en France</title>
  <meta charset="UTF-8">
  <script type="text/javascript">
    function testPhone(phone) {
      //format français 33 4 38 88 88 88 sans les espaces
      var model = /^(^33) ([1-9]{1}) ([0-9]{8})$/;
      if (model.test(phone)) {
        alert("ok");
        return true;
      } else {
        alert("erreur");
        return false;
      }
    }
  </script>
</head>
<body>
  <form method="POST" action="" onsubmit="return testPhone(this.phone.value)">
    Téléphone : <input type="text" name="phone" size="50"><br>
    <input type="submit" value="Envoyer" >
  </form>
</body>
</html>
```

Exemple de formulaire complet

A compléter : voir exercice 5 de la page suivante

```
<html>
<head>
  <title>Test de validité d'un formulaire complet</title>
  <script type="text/javascript">
    function testFormulaire(f) {
      //on passera en paramètre le formulaire
      // Ajouter ici les différents modèles (ou motifs) relatifs aux
      // champs à valider
      var testPhone = /^(^33)([1-9]{1})([0-9]{8})$/;
      if (!testPhone.test(f.phone.value)) {
        return false;
      }
      if (!testMail.test(f.mail.value)) {
        return false;
      }
      if (!testNom.test(f.nom.value)) {
        return false;
      }
      if (!testDate.test(f.date.value)) {
        return false;
      }
      return true;
    }
  </script>
</head>
<body>
  <form method="POST" action="" onsubmit="return testFormulaire(this)">
    Nom : <input type="text" name="nom" size="50"><br>
    Date : <input type="text" name="date" size="50"><br>
    Telephone : <input type="text" name="phone" size="50"><br>
    E-mail : <input type="text" name="mail" size="50"><br>
    <input type="submit" value="Envoyer" >
  </form>
</body>
</html>
```

Exercices

1. Vous devez créer un tableau des villes de la région (Carcassonne, Mende, Montpellier, Nîmes et Perpignan). Ensuite vous devrez parcourir le tableau et afficher les villes commençant par « M ». Pour cela, utilisez une expression régulière.
2. Vous devez créer un formulaire avec une zone de saisie pour le login et une fonction pour effectuer sa validation. Le login doit commencer par une lettre et ne contenir ensuite que des caractères alphanumériques. Sa longueur minimale sera de 7 caractères.
3. Vous devez créer un formulaire avec une zone de saisie pour le mot de passe et une fonction pour effectuer sa validation. Le mot de passe doit être robuste et donc contenir obligatoirement des majuscules, des minuscules, des chiffres et des caractères non alphanumériques. Sa longueur devra être comprise entre 7 caractères et 15 caractères.
4. Vous devez créer un formulaire avec une seule zone de saisie, dans laquelle vous saisirez «Dupont Jean ». Vous y ajouterez un bouton (<input> de type « button »). Lors de l'événement « onclick » du bouton, vous appellerez une fonction qui découpera le contenu de la zone de saisie et l'affichera dans une alerte sous la forme de « Jean Dupont ». (Voir méthode split). L'expression régulière sera du type : /(^.+)\s(.+)/
5. Compléter l'**exemple de formulaire complet** de la page précédente en écrivant les motifs (ou modèles).
6. En s'appuyant sur les exercices 2 et 3, créez un formulaire qui permet la saisie du login et du mot de passe, avec validation unique. Il faudra créer une fonction qui valide tout le formulaire.