République Tunisienne
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique

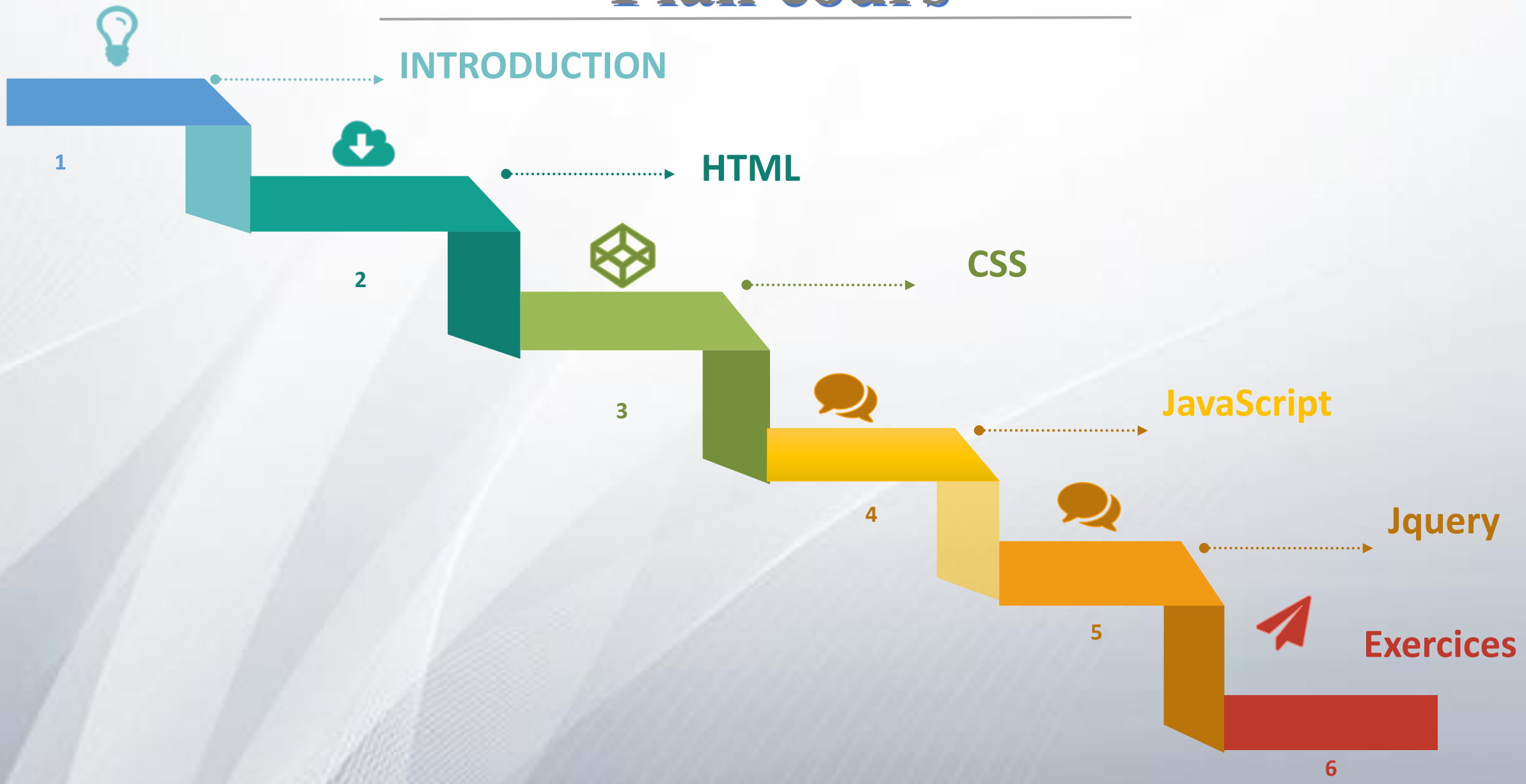
Technologies et programmation web
JavaScript

Njeh Maissa

Audiences: DLSI-ADBD

Année universitaire : 2023/2024

Plan cours


INTRODUCTION
1
HTML
2
CSS
3
JavaScript
4
Jquery
5
Exercices
6

JavaScript

3 Principe

- JavaScript permet **d'améliorer la présentation et l'interactivité** des pages web.
- JavaScript est un langage de **programmation (pseudo) orienté objet** mais ce langage est seulement interprété et ne s'utilise que incorporé a un document HTML. L'exécution du script, comme dans le cas des applets java , s'effectue sur **le poste client.**
- **L'expression pseudo-orienté objet désigne que,** comme pour java, tout objet dans JavaScript posséde des propriétés et des méthodes , mais la similitude s'arréte pratiquement là.

Principe

- Langage de script objet
- Syntaxe style C / C++ / Java
- Sensible à la casse
- N'est PAS du Java
- Exécuté par le client Web
- ...côté client seulement !
- Nombreux objets pour la manipulation HTML
- Gestion des événements HTML
- Rendre les pages interactives (HTML+CSS+JS)

4 Que peut-on faire en Javascript ?

- Puissance d'un **langage de programmation complet**, notamment structures conditionnelles (if/then/else), et boucles pour les répétitions (for, while).
- Possibilité de **changer le contenu de la page** (modifier le HTML), les styles (modifier le CSS), d'afficher des boîtes de dialogue (messages d'alerte), de récupérer les événements clavier et souris (et réagir en conséquence), de dessiner…

Quelques exemples courants d'utilisation de JavaScript :

- Afficher **des boîtes de dialogue** pour confirmer une action (un peu passé de mode)
- **Vérifier en temps réel** (sans soumettre) que les informations sont valides lorsqu'on remplit les champs d'un formulaire (le mot de passe a le bon nombre de caractères, téléphone = que des chiffres…)
- Objets dynamiques un peu complexes qui **répondent aux mouvements de souris** (ou frappes clavier) : un menu déroulant dynamique, un graphique qui montre l'évolution de quelque chose en temps réel (cours de la bourse par exemple), etc.

5 JavaScript n'est pas de java

- Souvent java et JavaScript sont comparés comme si ils étaient équivalent. Ce n'est pas le cas

JavaScript	Java
Code intégrer **dans** la page html	Module(applet) **distinct** de la page HTML
Code interprété par le navigateur au **moment** de l'exécution	Code source compilé **avant** son exécution
Codes de programmation **simples** mais pour des applications limitées	Langage de programmation **beaucoup** plus complexe mais plus performant
Permet d'accéder aux objets du navigateur	**N'accède pas** aux objets du navigateur
Confidentialité des codes nulles (code source visible)	Sécurité (code source compilé)

HTML et JavaScript

- la page HTML devra TOUJOURS contenir les deux balises spécifiques et indispensables:

<SCRIPT **language="JavaScript"**> **contenu du script** </SCRIPT>

OU bien :

<SCRIPT **type="text/JavaScript"**> **contenu du script** </SCRIPT>

Utilisation et emplacement

- **<SCRIPT>…</SCRIPT>** : autant de fois que nécessaire et n'importe où dans le document HTML.
- On traite la page HTML de haut en bas.
- De façon générale, les fonctions sont placées dans la section entête **(<HEAD> …</HEAD>)** du document.
  - Ainsi on interprète les scripts, si nécessaire avant que le reste du document ne soit téléchargé.

7 INTRODUTION

Utilisation et emplacement

- Utilisation de l'attribut «**SRC**» pour faire référence à l'URL, le fichier qui contient le script.

<SCRIPT language="JavaScript" **SRC="fichier.js"**> </SCRIPT>

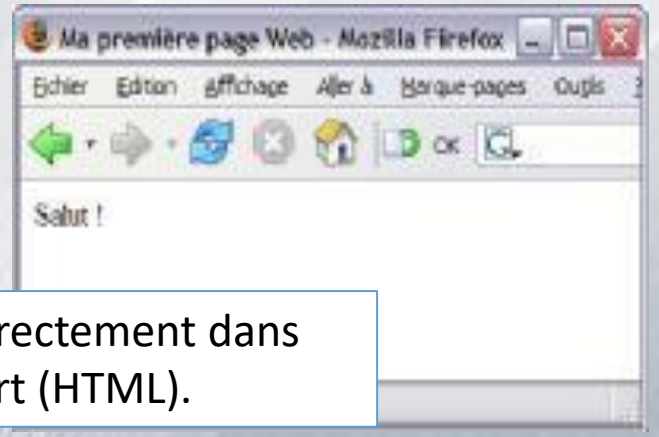
- NB.
  - **js** est l'extension du fichier javascript

Syntaxe

- La casse est importante: les majuscules et minuscules ne doivent jamais être interchangées entre elles. **BON: alert(); MAUVAIS: Alert();**
- Le caractère point-virgule est utilisé comme **séparateur** de fin de ligne mais sa présence n'est pas obligatoire si la ligne ne comporte qu'une seule instruction.
- Les **commentaires** simples(sur une ligne) sont indiqués par une double barre oblique **(//)**.
- Les **commentaires** qui s'étendent sur plusieurs lignes utilisent **/* et */**

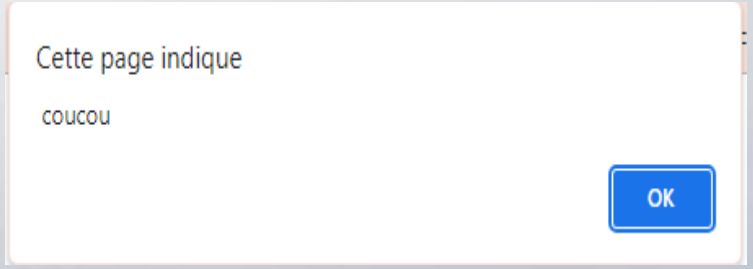
Exemple

```
<html>
 <head>
  <title>Ma première page Web</title>
 </head>
 <body>
 <script type="text/javascript"
 language="JavaScript">
  <!--
   document.writeln("Salut !") ;
  // -->
 </script>
</body>
</html>
```



writeln(): méthode écrit directement dans un flux de documents ouvert (HTML).

```
<html>
 <head>
  <title>Ma deuxieme page Web</title>
 </head>
 <body>
 <script type="text/javascript" language="JavaScript">
  <!--
   alert(« coucou") ;
  // -->
 </script>
</body>
</html>
```



Alert() : sert à afficher à l'utilisateur des informations simples de type **texte**. Une fois que ce dernier a lu le message, il doit cliquer sur **OK** pour faire disparaître la **boîte**

9 Variable

- Déclaration de variables facultative
- Variables non typées à la déclaration

var *nom_variable* ;

Const *nom_variable* ; si la valeur ne doit pas être modifiée

Let *nom_variable* ; déclare des variables locales réaffectables de portée bloc, en les initialisant éventuellement à une valeur.

- Typage dynamique à l'affectation
- Types gérés:
  - Nombres (**10, 3.14**)
  - Booléens (**true**, **false**)
  - Chaînes ("Salut !", 'bonjour !')
  - **null**
  - **undefined**

10 Structures conditionnelles

```
if (condition)
{
  instructions ;
}
[ else
{
  instructions ;
} ]
```

Exemple

```
<script language="JavaScript">
var code;
code= window.prompt("Entrez le code de 3 chiffres
    du produit", "");
if (code.length == 3)
{
document.write ("Le code est bon!");
}
 else
{
document.write("SVP, recommencez car le code \""
    +code+ "\" n'est pas valide.");
}
</script>
```

```
switch (expression)
{
  case étiquette :
    instructions ;
    break ;
  case étiquette :
    instructions ;
    break ;
  default :
    instructions ;
}
```

Exemple

```
<script language="JavaScript">
var choix;
choix=window.prompt( "Choisissez un code de produit:\n"
    +"1 (oranges), 2 (pommes)", "");
switch (choix) {
case "1":
document.write("Les oranges sont à 1,900 dinars le
    kilo.<BR>");
break;
case "2":
document.write("Les pommes sont à 4,500 dinars le
    panier.<BR>");
break;
default:
document.write("Vous avez tapé un mauvais code<BR>");
}
document.write("Avez vous besoin d'autres choses
    aujourd\'hui?<BR>");
</script>
```

La **prompt()méthode affiche une boîte de dialogue qui invite l'utilisateur à effectuer une saisie.**
La **prompt()**méthode renvoie la valeur d'entrée si l'utilisateur clique sur "OK", sinon elle renvoie null.

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <script>
            a=100;
            if (a<100)
                console.log("Inférieur");
            else{
                if (a==100)
                    console.log("Egal");
                else
                    console.log("Supérieur");
            }
        </script>
    </head>
    <body>

    </body>
</html>
```

```
...<html> == $0
  ▶ <head>…</head>
    <body>
html
Styles   Computed   Event Listeners   »
Filter                :hov  .cls  +
element.style {
}
html {                user agent stylesheet
  display: block;
}
Console   What's New                  ✕
top
Egal                  index.html:11
>
```

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <script>
            a=10;
            switch (a){
                case 0: {console.log("Zéro"); break;}
                case 1: {console.log("Un"); break;}
                default: console.log("Autre");
            }
        </script>
    </head>
    <body>

    </body>
</html>
```

```
  ▶ <head>…</head>
    <body>
html
Styles   Computed   Event Listeners   »
Filter                :hov  .cls  +
element.style {
}
html {                user agent stylesheet
  display: block;
}
Console   What's New                  ✕
top
Autre                 index.html:10
>
```

11 Structures itératives

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <script>
            i=1;
            while(i<=5){
                console.log("Bonjour "+i+"  fois");
                i++;
            }

        </script>
    </head>
    <body>

    </body>
</html>
```

Console What's New

top

Bonjour 1	fois	index.html:8
Bonjour 2	fois	index.html:8
Bonjour 3	fois	index.html:8
Bonjour 4	fois	index.html:8
Bonjour 5	fois	index.html:8

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <script>
            for (i=1;i<=10;i++)
                console.log("Bonjour "+i+"  fois");

        </script>
    </head>
    <body>

    </body>
</html>
```

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <script>
            a=10;
            do{
                console.log("Bonjour");
                a++;
            }
            while(a<5);

        </script>
    </head>
    <body>

    </body>
</html>
```

Console What's New

top

Bonjour	index.html:8

>

- 3 types de boites de messages
  - **Méthode alert()**
    - sert à afficher à l'utilisateur des informations simples de type texte. Une fois que ce dernier a lu le message, il doit cliquer sur OK pour faire disparaître la boîte
  - **Méthode confirm()**
    - permet à l'utilisateur de choisir entre les boutons OK et Annuler.
  - **Méthode prompt()**
    - La méthode prompt() permet à l'utilisateur de taper son propre message en réponse à la question posée
- La méthode **document.write** permet d 'écrire du code HTML dans la page WEB

13 FONCTION

- Valeur de retour non typée
- Arguments non typés

```
// Déclaration
```

```
function ma_fonction(arguments)
{
 instructions ;
 return quelque_chose; // ou pas…
}
ma_fonction(12) ; // Appel
```

```
function myFunction(a, b)
{
//Function returns the product of a & b
  return a * b;
}
let x = myFunction(4, 3); // Appel
```

Une **fonction JavaScript** est un **bloc de code** conçu pour effectuer une tâche particulière.
Une **fonction JavaScript** est exécutée lorsque « quelque chose » l'invoque (**l'appelle**).

13 FONCTION

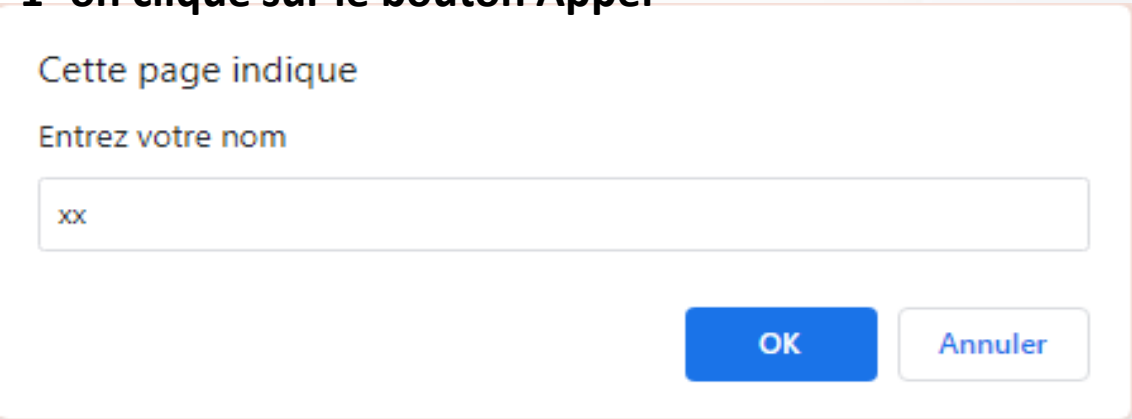
```
<!DOCTYPE html>
<html>
<body>
<p>Cliquez sur le bouton pour appeler la fonction !</p>
<button onclick="maFonction()">Appel</button>
<p id="demo"></p>
<script>
function maFonction()
{
   var personne;
   personne = prompt("Entrez votre nom", "Toto");

   if (personne != null)
   {
      document.getElementById("demo").innerH
         TML =  "Bonjour, " + personne + " !";
   }
}
</script>
</body></html>
```

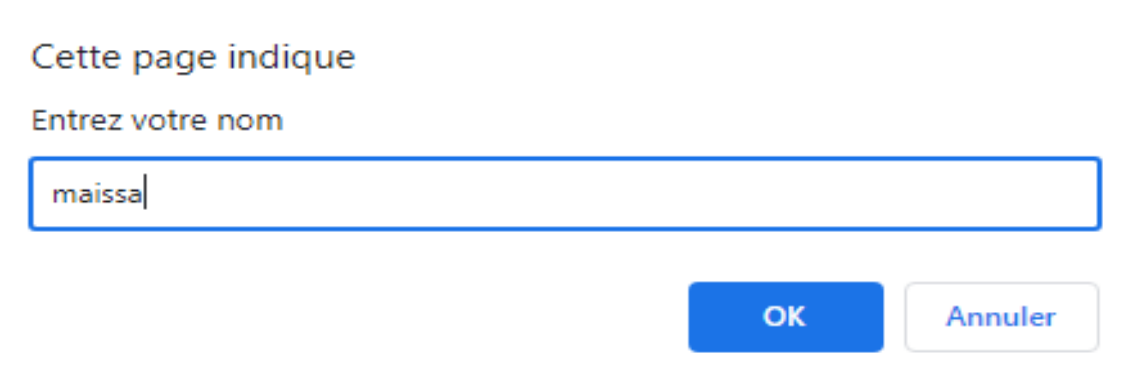
Cliquez sur le bouton pour appeler la fonction !

Appel

1- on clique sur le bouton Appel



2- Une boite de dialogue s'affiche



3- Saisir votre nom et cliquer sur le bouton ok

4- le nom s'affiche

Cliquez sur le bouton pour appeler la fonction !

Appel

Bonjour, maissa !

13 FONCTION

```
a=10;
var b=20;
function f(){
    c=30;
    var d=40;
    alert(a); // Affiche 10
    alert(b); // Affiche 20
}
f();
alert(c); // Affiche 30
alert(d); // Erreur
```

Quand on appelle la fonction, celle ci affichera les valeurs des variables globales **a** et **b** (même si **b** est précédée par **var** car elle est initialisée en dehors de la fonction). En dehors de la fonction, quand on affiche **c**, il n'y a aucun problème car même si elle est déclarée dans la fonction elle n'est pas précédée par **var**, ce qui la rend globale. Par contre, quand on essaie d'afficher **d**, une erreur survient, car **d** est initialisées à l'intérieur de la fonction avec le mot clé **var**, ce qui la rend locale à la fonction f().

Les Objets

- Les objets de JavaScript, sont :

soit **des entités pré définies** du langage

soit **créés par le programmeur**

  - Par exemple, le navigateur est un objet qui s'appelle "**navigator**".
  - La fenêtre du navigateur se nomme "**window**"
  - La page HTML est un autre objet, que l'on appelle "**document**".
  - Un formulaire à l'intérieur d'un "**document**", est aussi un **objet**.
  - **Un lien hypertexte** dans une page HTML, est encore un autre **objet**. Il s'appelle "**link**". etc…
- Les objets **javascript** peuvent réagir à des "**Evénements**".
- Tous les navigateurs ne supportent pas les mêmes objets
- Accès aux **propriétés** d'un **objet**

voiture.couleur.value

voiture.couleur.value = verte

15 Les objets

- **L'opérateur New**
  - L'opérateur *new* est utilisé pour **créer une nouvelle instance** ou un **nouveau type d'objet** défini par l'utilisateur ou de l'un des types d'objets prédéfinis, *Array, Boolean, Date, Function, Image, Number, Object*, ou *String*.
    - **nouvel_objet = new type_objet(parametres)**

texte = **new** String("Une chaîne de caractère");

- **L'opérateur Typeof**
  - L'opérateur *typeof* **renvoie une chaîne de caractères** indiquant quel **est le type de l'opérande**.

```
var i = 1;
typeof i; //retourne number
var titre="Les raisins de la colère";
typeof titre; //retourne string
var jour = new Date();      typeof jour; //retourne object
var choix = true;           typeof choix; //retourne boolean
var cas = null;             typeof cas; //retourne object
typeof parseFloat; //retourne function
typeof Math; //retourne object (IE 5.*, NS 6.*, NS 4.78, Opera 6.*,Opera 5.*
typeof Math; //retourne function NS 3.*, Opera 3.*
```

16 L'objet String

- **Propriété :**
  - ***length*** : retourne la longueur de la chaîne de caractères;
- **Méthodes :**
  - ***anchor( )*** : formate la chaîne avec la balise <A> nommée;
  - ***b( )*** : formate la chaîne avec la balise <B>;
  - ***big( )*** : formate la chaîne avec la balise <BIG>;
  - ***charAt( )*** : renvoie le caractère se trouvant à une certaine position;
  - ***charCodeAt( )*** : renvoie le code du caractère se trouvant à une certaine position;
  - ***concat( )*** : permet de concaténer 2 chaînes de caractères;
  - ***fromCharCode( )*** : renvoie le caractère associé au code;
  - ***indexOf( )*** : permet de trouver l'indice d'occurrence d'un caractère dans une chaîne;

16 L'objet String

- ***italics( )*** : formate la chaîne avec la balise <I>;
- ***lastIndexOf( )*** **:** permet de trouver le dernier indice d'occurrence d'un caractère;
- ***link( )*** **:** formate la chaîne avec la balise <A> pour permettre de faire un lien;
- ***slice( )*** **:** retourne une portion de la chaîne;
- ***substr( )*** **:** retourne une portion de la chaîne;
- ***substring( )*** **:** retourne une portion de la chaîne;
- ***toLowerCase( )*** : permet de passer toute la chaîne en minuscule;
- ***toUpperCase( )*** **:** permet de passer toute la chaîne en majuscules;

Chaînes : Exemples

```
var s = "Bon anniversaire Benjamin" ;
```

```
document.write(s.charAt(2)) ;
```

charAt() : renvoie le caractère se trouvant à une certaine position;

→ **n**

```
→ document.write(s.charCodeAt( 2)) ;
```

charCodeAt() : renvoie le code du caractère se trouvant à une certaine position;

→ **110**

concat( ) : permet de concaténer 2 chaînes de caractères;

```
document.write(s.concat(" du groupe C12")) ;
```

→ **Bon anniversaire Benjamin du groupe C12**

```
→ document.write(String.fromCharCode( 49, 50)) ;
```

String.fromCharCode() méthode convertit les valeurs Unicode en caractères. UTF-16 est utilisé dans tous les principaux systèmes d'exploitation comme Windows, IOS et Unix.

→ **12**

```
document.write(s.indexOf("Benjamin")) ;
```

fromCharCode() : renvoie le caractère associé au code;

→ **17**

indexOf() : permet de trouver l'indice d'occurrence d'un caractère dans une chaîne;

```
document.write(s.lastIndexOf("a")) ;
```

lastIndexOf() : permet de trouver le dernier indice d'occurrence d'un caractère;

→ **21**

```
document.write(s.match(/Benjamin$/)) ;
```

→ **Benjamin (*null si non trouvé*)**

match(): Une recherche de "`Benjamin`" à l'aide d'une chaîne **s**

Chaînes : Exemples

```
var s = "Bon anniversaire Benjamin" ;
```

```
document.write(s.slice( 17)) ;
```

slice( ) : retourne une portion de la chaîne;

➔ **Benjamin**

```
document.write(s.split( " ")) ;
```

split()méthode divise une chaîne en un tableau de sous-chaînes.

➔ **Bon,anniversaire,Benjamin**

```
document.write(s.substr( 4, 12)) ;
```

substr( ) : retourne une portion de la chaîne;

➔ **anniversaire**

```
document.write(s.substring( 4, 16)) ;
```

substring( ) : retourne une portion de la chaîne;

➔ **anniversaire**

```
document.write(s.toUpperCase()+s.toLowerCase()) ;
```

➔ **BON ANNIVERSAIRE BENJAMINbon anniversaire benjamin**

L'objet Array

- **Propriété :**
- ***length*** : retourne le nombre d'éléments du tableau;
- **Méthodes :**
  - ***concat( )*** : permet de concaténer 2 tableaux;
  - ***join( )*** : converti un tableau en chaîne de caractères;
  - ***reverse( )*** : inverse le classement des éléménts du tableau;
  - ***slice( )*** : retourne une section du tableau;
  - ***sort( )*** : permet le classement des éléments du tableau;

Remarque : Les index des tableaux commencent par 0. [0] est le premier élément. [1] est le deuxième élément.

L'objet Array

- **Nom_du_tab = new array(x);** Ou x le nombre d 'element du tableau.
- **Nom_du_tab[i] =' element';** Ou i est un nombre compris entre 0 et x-1

Un tableau à une dimension

Exemple : un carnet d'adresse avec 3 personnes

Construction du tableau:

carnet = new array(3);

Ajout des données :

carnet [0]='ali';
carnet [1]='asma';
carnet [2]='aya';

Pour accéder a un élément :

document.write(carnet[2]);

Pour un tableau à 2 dimensions

Exemple : un tableau de 3 sur 3

tarif	t.small	t.medium	t.large
chemises	32	35	37
polos	27	28	30
T-shirts	20	22	23

nom = new array(3);

nom [0]=new array(3);
nom [1]=new array(3);
nom [2]=new array(3);

nom [0][0]='1200'; nom [0][1]='1250';nom [0][2]='1300';
nom [1][0]='800';nom [1][1]='850'; nom [1][2]='900';
nom [2][0]='500'; nom [2][1]='520';nom [2][2]='540';

21 L'objet Math

- **Méthodes** :
  - ***abs( ), exp( ), log(), sin( ), cos( ), tan( ), asin( ), acos( ), atan( ), max( ), min( ), sqrt( )*** sont les opérations mathématiques habituelles;
  - ***atan2( )*** : retourne la valeur radian de l'angle entre **l'axe des abscisses et un point**;
  - ***ceil( )*** : retourne le plus **petit** entier supérieur à un nombre;
  - ***floor( )*** : retourne le plus **grand** entier inférieur à un nombre;
  - ***pow( )*** : retourne le **résultat** d'un nombre mis à une certaine puissance;
  - ***random( )*** : retourne un nombre **aléatoire** entre 0 et 1;
  - ***round( )*** : **arrondi** un nombre à l'entier le **plus proche**.

- **Propriétés :**
  - ***E*** : renvoie la valeur de la constante d'Euler (~2.718);
  - ***LN2*** : renvoie le logarithme népérien de 2 (~0.693);
  - ***LN10*** : renvoie le logarithme népérien de 10 (~2.302);
  - ***LOG2E*** : renvoie le logarithme en base 2 de e (~1.442);
  - ***LOG10E*** : renvoie le logarithme en base 10 de e (~0.434);
  - ***PI*** : renvoie la valeur du nombre pi (~3.14159);
  - ***SQRT1_2*** : renvoie 1 sur racine carrée de 2 (~0.707);
  - ***SQRT2*** : renvoie la racine carrée de 2 (~1.414);

L'objet Math

```
document.write(115.04+15) ;
➔ 130.04000000000002
document.write(Math.PI) ;
➔ 3.141592653589793
document.write(Math.abs( -12.34)) ;
➔ 12.34
document.write(Math.floor( 12.54)) ;
➔ 12
 document.write(Math.round( 12.54)) ;
➔ 13
document.write(Math.ceil( 12.54)) ;
➔ 13
document.write(Math.random()) ;
➔ 0.394555831655689
```

L'objet Date

Méthodes :

- ***getFullYear( ), getYear( ), getMonth( ), getDay( ), getDate(), getHours( ), getMinutes( ), getSeconds( ), getMilliseconds( ):*** retournent respectivement l'année complète, l'année (2chiffres), le mois, le jour de la semaine, le jour du mois, l'heure, les minutes, les secondes et les millisecondes stockés dans l'objet *Date*;
- ***getFullYear( ), getYear( ),*** *...* retournent respectivement l'année complète, l'année (2chiffres), ... stockés dans l'objet *Date* en temps universel;
- ***setFullYear( ), setYear( ),*** *...* remplacent respectivement l'année complète, l'année (2chiffres), ... dans l'objet *Date*;
- **setUTCFullYear( ), setUTCYear( ), ...** remplacent l'année complète, l'année (2chiffres), ... dans l'objet Date en temps universel;
- **getTime( ) :** retourne le temps stocké dans l'objet Date;
- **getTimezoneOffset(**) : retourne la différence entre l'heure du client et le temps universel;
- **toGMTString( ), toLocaleString( ), toUTCString( )** : convertissent la date en chaîne de caractère selon la convention GMT, selon la convention locale ou en temps universel;

L'objet Date

```
new Date(year,month,day,hours,minutes,seconds,ms)
```

- var today = new Date()

document.write(today) ; **Tue Nov 02 2011 00:11:36 GMT+0100**

- var birthday = new Date("December 17, 1995 03:24:00")

document.write(birthday) ; **Sun Dec 17 1995 03:24:00 GMT+0100**

- birthday = new Date(95,11,17)

document.write(birthday) ; **Sun Dec 17 1995 00:00:00 GMT+0100**

- birthday = new Date(95,11,17,3,24,0)

document.write(birthday) ; **Sun Dec 17 1995 03:24:00 GMT+0100**

JavaScript compte les mois de **0** à **11**
Janvier = 0 /Décembre = 11 .

```
document.write(birthday.getDay())  ;0
document.write(birthday.getDate()) ;17
birthday.setDate(25);
document.write(birthday) ; Mon Dec 25 1995 03:24:00 GMT+0100
```

getDay, getDate retournent respectivement le jour de la semaine, le jour du mois,

Dates : Exemples

JavaScript stocke les dates en millisecondes depuis le 1er janvier 1970.
L'heure zéro est le 1er janvier 1970 00:00:00 UTC .
Un jour (24 heures) équivaut à 86 400 000 millisecondes.
Il est maintenant **1700561646742** millisecondes après le 1er janvier 1970.

Sun Dec 17 1995 03:24:00 GMT+0100

- `document.write(birthday.getMonth())` ; **11**
- `birthday.setMonth(10)` ;

`document.write(birthday)` ;**Sat Nov 25 1995 03:24:00 GMT+0100**

`document.write(birthday.getYear())` ; **95** retournent l'année (2chiffres),

- `birthday.setYear(96)` ; remplacent l'année (2chiffres),

`document.write(birthday)` ; **Mon Nov 25 1996 03:24:00 GMT+0100**

`document.write(birthday.getFullYear())` ; **1996**

- `birthday.setFullYear(2010)` ;

`document.write(birthday)` ;**Thu Nov 25 2010 03:24:00 GMT+0100**

Images : Propriétés & Méthodes

- **Propriétés**
  - `complete`
  - `width`
  - `height`
  - `name`
  - `src`
  - …
- **Méthodes**
  - `constructeur`
    - **`Image()`**
    - **`Image(`*`largeur, hauteur`*`)`**

Images: Exemples

```
<script type="text/javascript" >
// Une image
rouge = new Image(100, 100) ;
rouge.src = 'rouge.gif' ; // Préchargement

// Une autre image
vert = new Image(100, 100) ;
vert.src = 'vert.gif' ; // Préchargement

// Modification de la 13e image de la page Web
document.images[ 12].src = rouge.src ;

// Modification de la 15e image de la page Web
document.images[ 14].src = rouge.src ;
</script>
```

27 Les objets du navigateur

- L'objet le plus haut dans la hiérarchie est *window* qui correspond à la fenêtre même du navigateur.
- L'objet *document* fait référence au **contenu de la fenêtre**.
- *document* regroupe au sein de propriétés **l'ensemble des éléments** HTML présents sur la page.
- Pour atteindre ces différents éléments, nous utiliserons :
  - **soit des méthodes propres à l'objet document**, comme la méthode ***getElementById( )***, qui permet de trouver l'élément en fonction de son **identifiant (ID)**;
  - **soit des collections d'objets** qui regroupent sous forme de **tableaux** Javascript tous les éléments de type déterminé.

L'objet window

- ***closed*** : indique que la fenêtre a **été fermée**;
  - ***defaultStatus*** : indique le message par défaut dans la barre de status;
  - ***document*** : retourne l'objet *document* de la fenêtre;
  - ***frames*** : retourne la collection de cadres dans la fenêtre;
  - ***history*** : retourne l'historique de la session de navigation;
  - ***location*** : retourne l'adresse actuellement visitée;
  - ***name****:* indique le nom de la fenêtre;
  - ***navigator*** : retourne le navigateur utilisé;
- ***opener*** : retourne l'objet *window* qui a créé la fenêtre en cours;
- ***parent*** : retourne l'objet *window* immédiatemment supérieur dans la hiérarchie;
- ***self*** *:* retourne l'objet window correspondant à la fenêtre en cours;
- ***status*** : indique le message affiché dans la barre de status;
- ***top*** : retourne l'objet *window* le plus haut dans la hiérarchie

L'objet window

- ***blur( )*** : enlève le focus de la fenêtre;
- ***close( )*** : ferme la fenêtre;
- ***focus( )*** : place le focus sur la fenêtre;
- ***moveBy( )*** : déplace d'une **distance**;
- ***moveTo( )*** : déplace la **fenêtre vers un point spécifié;**
- ***open ()*** : **ouvre** une nouvelle fenêtre;
- ***print( )*** : **imprime** le contenu de la fenêtre;
- ***resizeBy( )*** : **redimensionne** d'un certain rapport;
- ***resizeTo( )*** : redimensionne la fenêtre;
- ***setTimeout( )*** : évalue une chaîne de caractère après un certain laps de temps.

L'objet document

- **Propriétés** :
  - ***applets*** : retourne la collection d'applets java présente dans le document;
  - ***cookie*** : permet de stocker un cookie;
  - ***domain*** : indique le nom de domaine du serveur ayant apporté le document;
  - ***forms*** : retourne la collection de formulaires présents dans le document;
  - ***images*** : retourne la collection d'images présentes dans le document;
  - ***links*** : retourne la collection de liens présents dans le document;
  - ***referrer*** : indique l'adresse de la page précédente;
  - ***title*** : indique le titre du document.
- **Méthodes :**
  - ***close( )*** : **ferme** le document en écriture;
  - ***open( )*** : **ouvre** le document en écriture;
  - ***write( )*** : écrit dans le document;
  - ***writeln( )*** : écrit dans le document et effectue un retour à la ligne

31 L'objet navigator

- **Propriétés**
  - ***appName*** : application (Netscape, Internet Explorer)
  - ***appVersion*** : numero de version.
  - ***platform*** : système d'exploitation (Win32)
  - ***plugins***
  - ***language***
  - ***mimeTypes***
  - ***JavaEnabled()***

53 Les évènements

Javascript **est un langage événementiel**.

Un **événement** est une action de l'utilisateur qui peut se produire sur le navigateur. Le plus célèbre est sans doute le **clic** qui peut survenir sur un objet quelconque (hyperlien, image, bouton...).

Le navigateur écoute les événements qui se produisent sur la page Web puis les transmet au script par le biais de ce que l'on appelle **le gestionnaire d' événements**.

Le **gestionnaire d'événements** permet d'associer un traitement à un événement. Sa syntaxe respecte la forme suivante:

onEvénement = "code Javascript édité directement ici, ou fonction appelée à cet endroit"

```
<html>
   <head>
      <meta charset="UTF-8" />
      <script language="javascript">
         function f(){
            alert("Vous avez bien cliqué sur le bouton!");
         }
      </script>
   </head>
   <body>
      <input type="button" value="Cliquez ici" onClick="f()" />
   </body>
</html>
```

Le fait de cliquer sur le **bouton** donne naissance à l'événement **click** qui appellera (grâce au gestionnaire d'événements) la fonction **f()** définie dans le **<head>**. On aura donc le message "Vous avez bien cliqué sur le bouton!" qui s'affichera dans une **boite de dialogue alert().**

```html
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8" />
      <script language="javascript">
        function f(){
           alert("bonjour a tous");
        }
        function l(){
           alert("on load");
        }
        function g(){
           alert("evenement detecte ");
        }
     </script>
   </head>
   <body onload="l()">
    <input type="button" value="bouton1" onclick="alert('Bonjour1')" />
    <input type="button" value="bouton2" onmouseover="alert('Bonjour2')" />
    <input type="button" value="bouton3" onmouseout="f()" />
    <input type="button" value="bouton4" ondblclick="f()" />
    <input type="text" name="zonedetexte" onkeyup="g()" />
    <input type="text" onblur="alert(‘ceci est un Blur!')" />
   </body>
</html>
```

OnClick = lorsque l’utilisateur clique sur un bouton
Onmouseover = lorsque l’utilisateur place le pointeur de la souris sur un lien ou tout autre élément
Onmouseout = lorsque le pointeur de la souris quitte le bouton
Ondblclick = lorsque on clique sur un objet deux fois de suite (d
Onload = lorsque la page est chargée par le navigateur
Onkeyup = lorsque une touche du clavier est relâchée
OnBlur = lorsqu’un élément de formulaire perd le focus c’est a dire que l’utilisateur clique hors du champs et que la zone d’entrée n’est plus active

Les évènements

click (onClick) Cet événement est capturé sur un objet quand on clique dessus. Idéal pour les boutons, images, hyperliens, vidéos...

dblClick (onDblClick)Quand on clique sur un objet deux fois de suite (double clic).

mouseOver (onMouseOver)Quand on survole un objet avec le curseur de la sourie.

mouseOut (onMouseOut)Quand on quitte l'objet avec la sourie après l'avoir survolé.

focus (onFocus) Quand on active un élément (quand on place le curseur dans un champ de formulaire, par click ou par tabulation, pour commencer la saisie par exemple).

blur (onBlur)Quand un élément perd le focus (quitter un champ de formulaire après être activé par exemple).

keyDown (onKeyDown)Quand une touche du clavier est enfoncée.

keyUp (onKeyUp)Quand une touche du clavier est relâchée.

keyPress (onKeyPress)Quand une touche du clavier est maintenue enfoncée

load (onLoad) Quand un élément est chargé par le navigateur. Elle peut être appliquée à la page entière (balise **<body>**), dans ce cas l'événement se produira quand tous les éléments de la page seront chargés.

unLoad (onUnload)Quand la page en cours est fermée ou quittée.

55 Les évènements

resize (onResize)Quand l'internaute redimensionne la taille de la fenêtre du navigateur.

select (onSelect)Quand l'internaute essaie de sélectionner le texte contenu dans l'objet accueillant l'événement.

change (onChange)Quand l'internaute change le contenu d'un élément (liste de sélection ou zone de texte par exemple).

submit (onSubmit)Quand l'internaute clique sur n'importe quel bouton de type **submit** présent dans la page (ou dans le formulaire).

mouseDown (onMouseDown)Quand l'internaute appuie sur n'importe quel bouton de la souris.

mouseUp (onMouseUP)Quand le bouton de la souris et relâché.

mouseMove (onMouseMove)Quand l'internaute fait bouger le curseur de la souris dans la zone accueillant l'événement.

dragStart (onDragStart)

Se produit quand l'internaute commence le déplacement d'un élément par "glisser-déposer" (Drag & Drop).

dragEnter (onDragEnter) Se produit quand l'internaute entre dans la zone où sera déposé l'élément glissé.

dragOver (onDragOver) Se produit quand l'internaute survole la zone où sera déposé l'élément glissé (même si le concept est proche de celui de

onDragEnter, ils sont des événements légèrement différents).

drop (onDrop)Se produit quand l'internaute dépose l'élément glissé dans la zone prévue à cet effet.

Les objets et leur hiérarchie

- L'objet le plus élevé est la fenêtre du navigateur, il est identifié par le nom **window**. Cet objet peut être manipulé directement à travers des fonctions qui lui sont prédéfinies.
- On appelle ces fonctions des **méthodes**. A titre d'exemple **alert()** qui affiche les messages des exemples précédents est une méthode de l'objet **window**, on aurait pu donc l'écrire comme ceci: **window.alert()**.
- **Le point (.)** permet de passer d'un objet à une de ces méthodes ou d'un objet parent à un autre objet enfant, ou encore d'un objet à une variable qui lui est propre. On appelle cette variable un **attribut**.

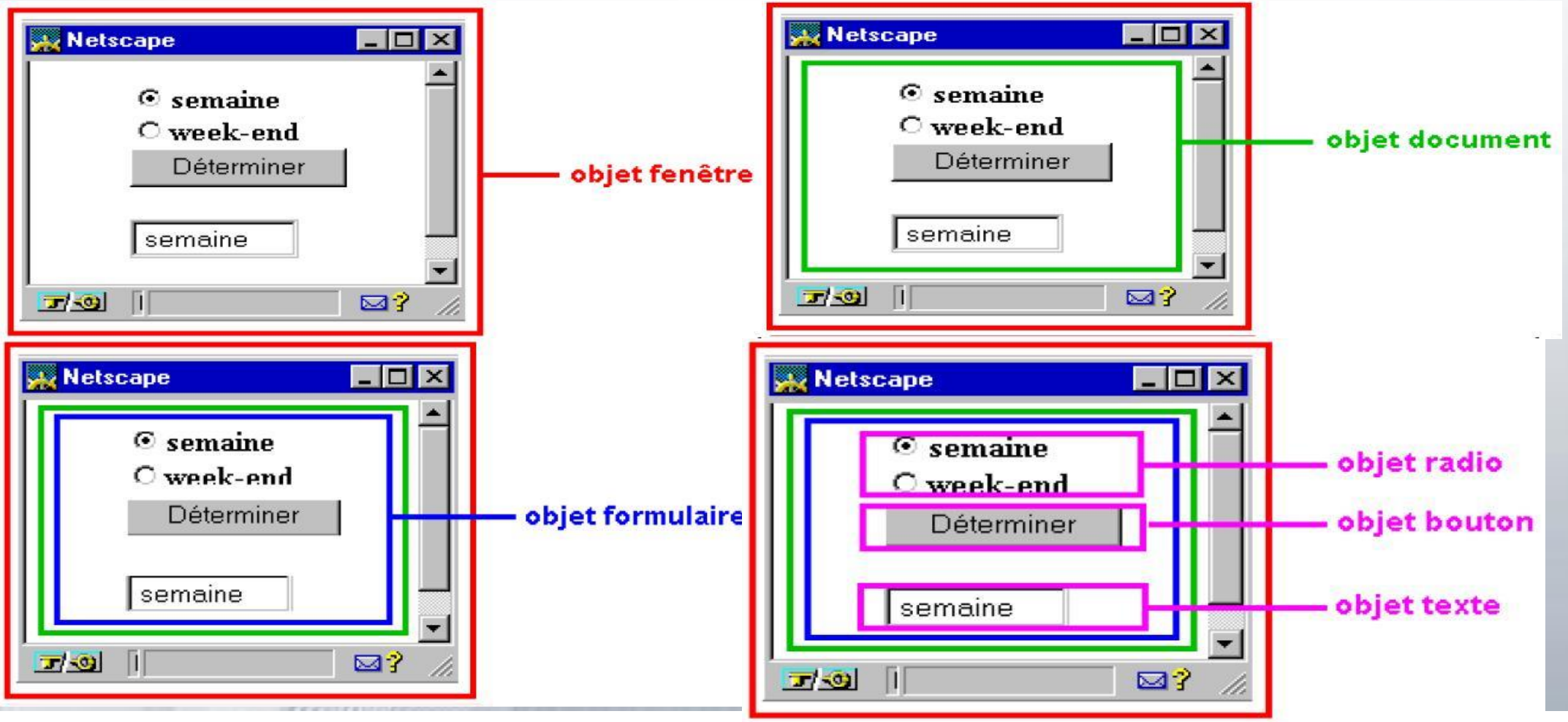
- Un **objet** peut représenter un élément HTML (image, texte, formulaire...) ou un élément propre au Javascript qui sert à exécuter des traitements spéciaux.
- Une **méthode** est une **fonction** prédéfinie appartenant à l'objet et qui permet de faire les traitement sur celui-ci (on peut dire qu'elle décrit le comportement de l'objet).
- Un **attribut** est une **variable** qui appartient à l'objet . Elle représente une propriétés ou caractéristique de celui-ci. On peut le manipuler grâce aux **méthodes** ou directement.

Les objets et leur hiérarchie

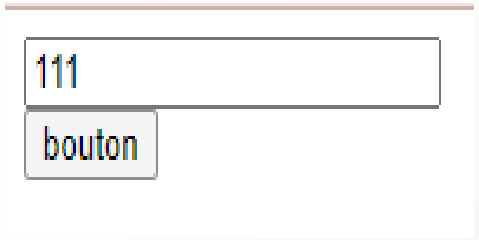
On voit apparaitre la notion de la hiérarchie des objets JavaScript :

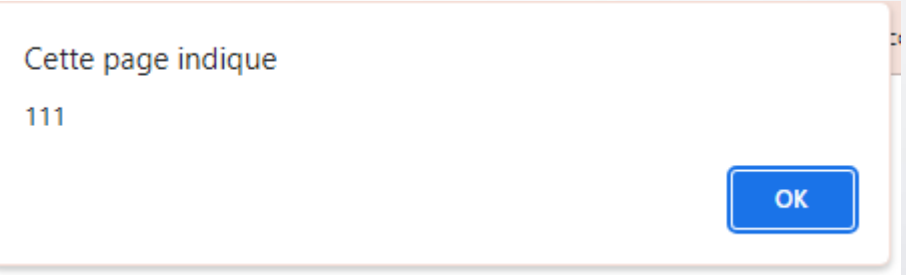
l'objet **fenêtre** contient l'objet **document** qui contient l'objet **formulaire** qui contient a son tour l'objet **radio, texte..**

Pour accéder a un objet , il faudra donner le chemin complet de l'objet en allant du contenant le plus extérieur a l'objet a l'objet référence window.document.form.radio

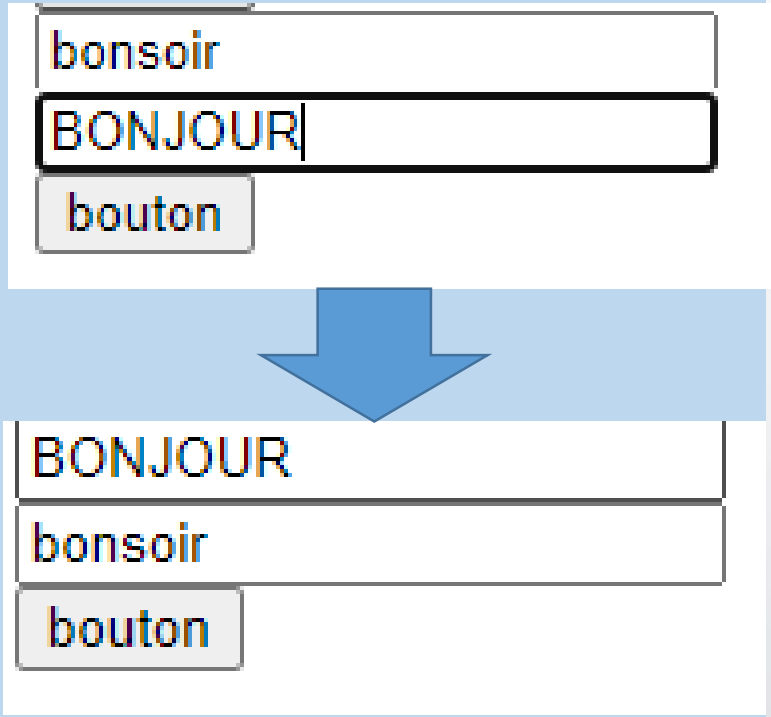


```
<!DOCTYPE html> <html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function f(){
        alert(document.fo.montexte.value);
        ou bien
        alert(document.forms[0].montexte.value);
      //accede a tous les formulaires 0:premier indice de
formulaire,montexte:sous objet ,value:attribut
        ou bien
        alert(document.forms["fo"].montexte.value);
//tableau associative : le nom de formulaire
              }
        function init(){
        document.forms[0].montexte.focus() ;
        //activer automatiquement le chmaps montexte
    </script> </head>
  <body onload="init()">
  <form name="fo">
   <input type="text" name="montexte"  /><br/>
   <input type="button" value="bouton" onclick="f()" /><br/>
   </form></body></html>
```





```html
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8" />
      <script language="javascript">
        function f(){
           alert(document.fo.montexte.value);
        }
        function permuter(){
           x=document.fo.z1.value;
           document.fo.z1.value=document.fo.z2.value;
 document.fo.z2.value=document.fo.z3.value;
           document.fo.z3.value=x;
        }
       </script>
   </head>
   <body onload="init()">
   <form name="fo">
    <input type="text" name="montexte"  /><br/>
    <input type="button" value="bouton" onclick="f()" /><br/>
    <input type="text" name="z1"  /><br/>
    <input type="text" name="z2"  /><br/>
    <input type="button" value="bouton" onclick="permuter()" /><br/>
```



Les objets et leur hiérarchie

L'objet **window** contient un sous-objet, c'est le document HTML lui même. Cet objet est identifié par le nom **document**. Cet objet peut contenir les éléments HTML conventionnels **comme du texte, les images, les formulaires...**

Tous ces éléments constituent des **sous-objets** de l'objet document. Par exemple l'objet **forms** désigne les formulaires contenus dans le document, et l'objet images englobe toutes les images intégrées...

L'objet formulaire par exemple contient à son tour des sous-objets, ce sont les champs de formulaire etc etc...

Pour commencer, le navigateur est inclus dans sa fenêtre du nom de **window**, c'est l'objet en tête de la hiérarchie. Puis il y a le document HTML du nom de **document** qui vient après. Donc le nom complet de l'objet document est **window.document.**

L'objet **document** contient un formulaire dont l'attribut **name** vaut **fo**. Alors, ce formulaire est identifié en Javascript par son nom **fo** ou plutôt par son nom complet **window.document.fo.**

Le formulaire **fo** renferme deux champs de formulaire. Intéressons nous au premier champs qui est une zone de texte du nom de **login**. Javascript identifiera donc cet objet par son non. Mais pour que l'identifiant soit correct il faut qu'il soit complet, l'objet "zone de texte" sera donc identifié par **window.document.fo.login.**

Cependant, la déclaration de l'objet **window** n'est pas obligatoire. nous avons appelé la méthode **alert**() sans le préfixe **window**. Donc, l'identifiant de la "zone de texte" peut s’exprimer comme ceci: **document.fo.login.**

Les objets et leur hiérarchie

Maintenant que nous avons identifié notre objet on va faire appel à ses attributs et ses méthodes pour pouvoir l'exploiter. Pour faire simple, on va s'intéresser seulement aux **attributs** dans un premier temps

Les attributs de l'objet "**zone de texte**" que nous avons identifié par **document.fo.login** sont les mêmes attributs dont l'élément HTML correspondant dispose.

Par exemple:

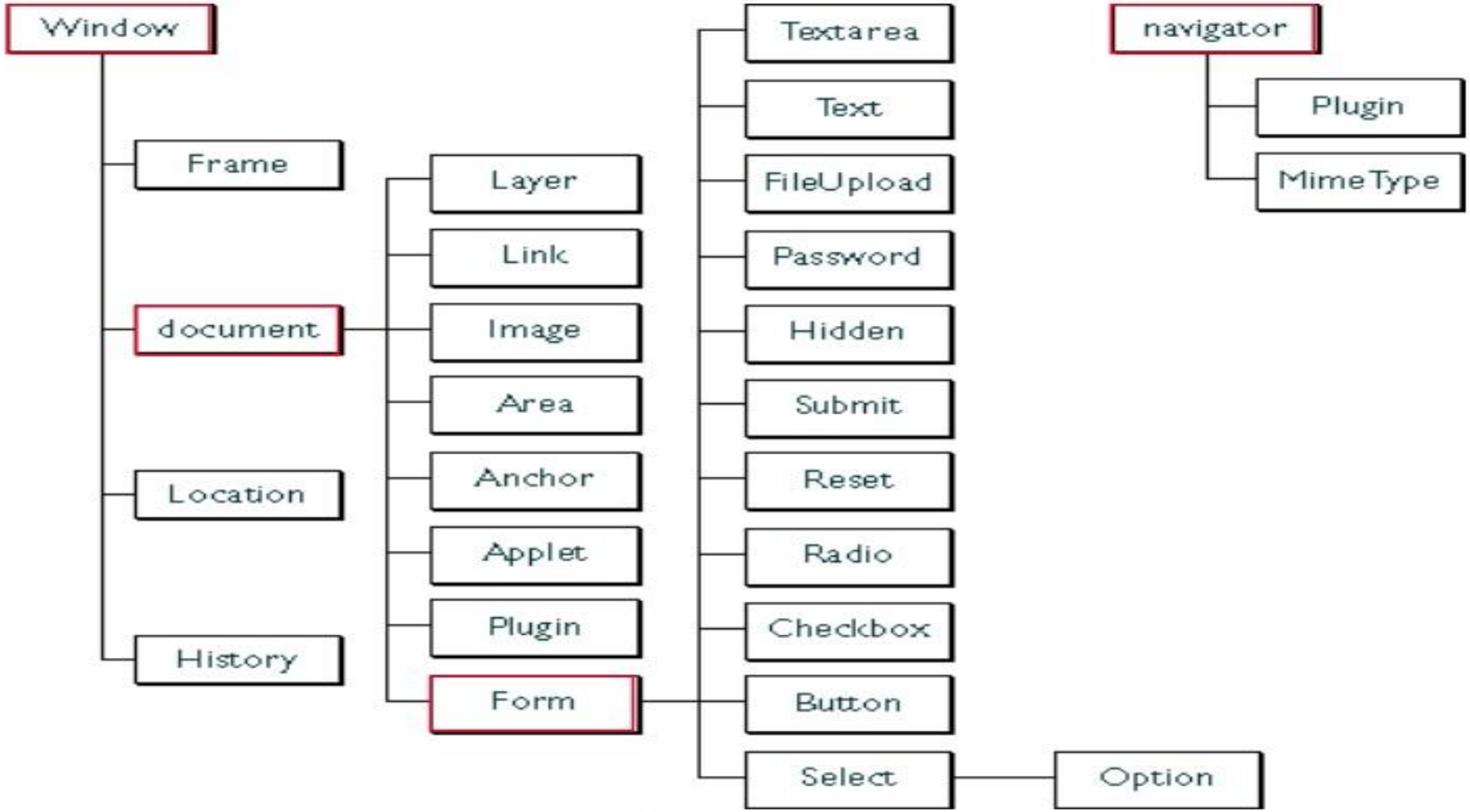
- **value: qui désigne la valeur que contient la zone de texte.**
- **size: qui désigne se taille.**
- **tabindex: qui désigne son ordre de tabulation.**

En général c'est l'attribut value qui est souvent utilisé. Pour accéder à la valeur de la zone de texte on fait **document.fo.login.value**.

Pour mettre en pratique ce que nous avons vu. Imaginons que nous voulons améliorer le code HTML de tout à l'heure en y intégrant du code Javascript qui permet d'afficher dans une boite alert() le texte que l'internaute a saisi dans la zone de texte après avoir cliqué sur le bouton.

Le fait de cliquer sur le bouton appelle la **fonction f() (onClick="f()").**

La fonction f() déclarée plus haut, affiche le contenu de la zone de texte login dans une boite de dialogue **alert() (alert(document.fo.login.value);).**


Window
Frame
document
Location
History
Layer
Link
Image
Area
Anchor
Applet
Plugin
Form
Textarea
Text
FileUpload
Password
Hidden
Submit
Reset
Radio
Checkbox
Button
Select
Option
navigator
Plugin
MimeType

Les méthodes de l'objet window

Objet window

Méthodes de gestion des boites de dialogue

- Alert()
- Confirm()
- Promt()

Méthodes d'ouverture des fenêtres

- open()
- close()
- focus()
- opener

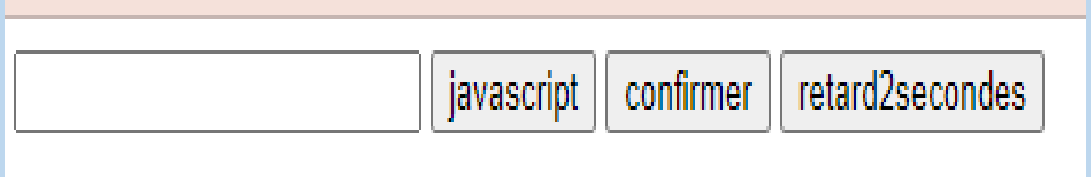
Méthodes de gestion des minuteries

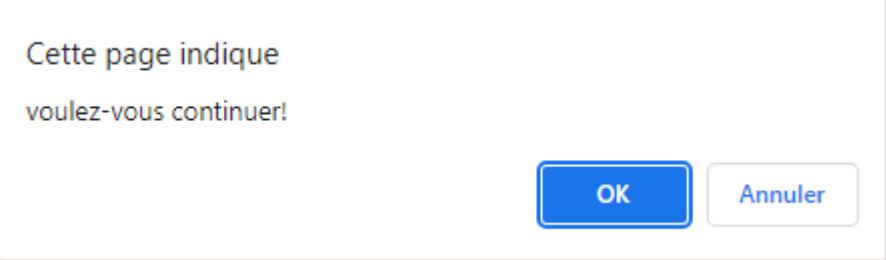
- setTimeout()
- setInterval()
- clearTimeout()

Méthodes de gestion des fenêtres

- moveTo() et moveBy
- resizeTo() et resizeBy
- screenLeft() ou screenX
- screenTop ou screenY
- scroolHeight
- scroolWidth ou outerWidth
- clientHeight ou innerHeight
- clientWidth ou innerWidth
- scroolTop ou pageXOffset
- scroolLeft ou pageYOffset
- scroolBy()
- scroolTo()

```html
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8" />
      <script language="javascript">
         function f(){
            alert("Bonsoir");
         }
         function confirmer(){
            confirm("voulez-vous continuer!");
         }
         function g(){
            // appeler f avec une retard 2000milessecondes=2secondes
            setTimeout("f()",2000);
         }
      </script>
   </head>
   <body>
      <form name="fo">
         <input type="text" name="login" />
         <input type="button" value="javascript" onClick="f()" />
         <input type="button" value="confirmer" onClick="confirmer()" />
         <input type="button" value="retard2secondes" onClick="g()" />
      </form>   </body> </html>
```






```html
<input type=‘ radio" name=‘choix’ value=’1’>Fem
<input type=‘ checkbox" name=‘check1’ value=’1’>choix
```

Nommage des objets-éléments

- Pour pouvoir **manipuler** un objet en javaScript, il doit posséder **un nom**
- Pour pouvoir **distinguer** les différents **objets-éléments** d'une page web, il suffit de leur donner **un nom à travers de l'attribut NAME**
  - **<Table Name=« tableau1 »>…**
  - **<Table Name=« tableau2 »>…**
  - **<Form Name = « formulaire1 »>…**
  - **<Form Name =« formulaire2 »>…**
  - **<Textarea Name =« texte1»>…**
- Dans le cas où l'objet serait **unique** alors pas besoin de nom pour désigner cet objet
  - Exemple :
  - **le cas de BODY(une seul BODY par document),**
  - **DOCUMENT (un seul DOCUMENT par fenêtre)**

Manipulation des objets

- Pour **adresser un objet**, il ne suffit pas de **donner son nom** : il faut aussi préciser son « **chemin d'accès** » dans l'arborescence de la structure

```
<HTML>
<BODY
onLoad="window.document.formulaire.zone.value='Bonjour';">
<FORM name="formulaire">
<INPUT NAME="zone" TYPE="text">
</FORM>
</BODY>
</HTML>
```

- Si le nom de la fenêtre est omis, le navigateur utilisera par défaut **la fenêtre courante**
- Dans le cas de cadres (**frames**), il est pertinent de donner le nom de la fenêtre
- Il est possible aussi d'omettre **window.document** : l'adressage réussi puisqu'il n'y a qu'un seul objet « **document**» dans la fenêtre

Contrôle de formulaires

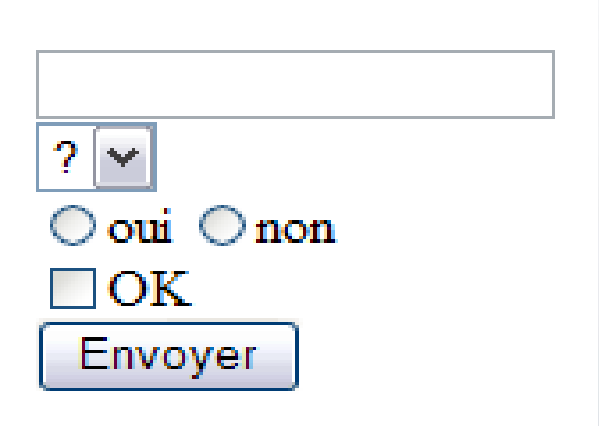
- Vérifier la cohérence de la saisie
- Contrôles sur le client
- Évite les transmissions **client / serveur**
- Contrôles possibles:
  - Présence de valeur
  - Numérique / Chaîne
  - Expressions régulières
- Événement **`onSubmit`**

L'événement **onsubmit** se produit lorsqu'un formulaire est soumis.

```
<html><head>
<title>Contrôle</title>
<script type="text/javascript">
function verif() {
  if (document.formu.txt.value != '')
  return window.confirm('Envoyer ?') ;
  return false ; }
</script>
</head>
<body>
<form name="formu" action= "..." method="GET"
onSubmit="return verif() ;">
  <input type="text" name="txt">
  <input type="submit" value="Envoyer">
</form>
</body></html>
```

Formulaires : Exemple

```
<form name='formu'
onSubmit='return verif(this);'>
 <input type='text' name='texte'><br>
 <select name='sel'>
  <option>?
  <option value=1>Un
  <option value=2>Deux
</select><br>
 <input type='radio' name='rad' id='rad1'>
 <label for='rad1'>oui</label>
 <input type='radio' name='rad' id='rad2'>
  <label for='rad2'>non</label><br>
 <input type='checkbox' name='chk'
 id='chk1'>
<label for='chk1'>OK</label><br>
<input type='submit' value='Envoyer'>
</form>
```



Formulaires : accès aux champs

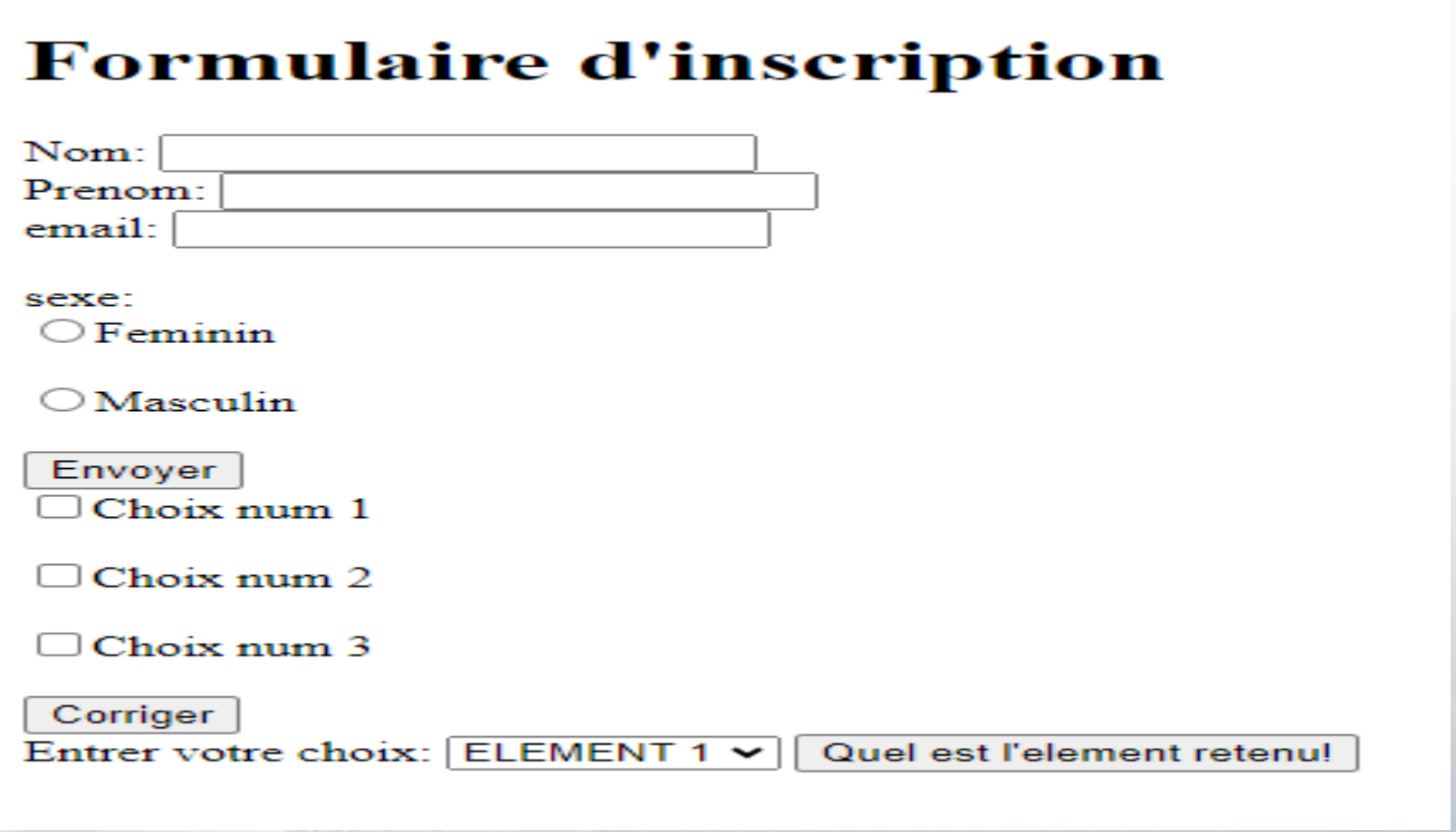
```
<form name='formu' onSubmit='return verif(this);'>
 <input type='text' name='texte'> ...
<script type="text/javascript">
function verif(f) {
 ... f.texte.value ;
 </script>
<form name='formu' onSubmit='return verif();'>
 <input type='text' name='texte'> ...
<script type="text/javascript">
function verif() {
 ... document.formu.texte.value ;
 ... document.forms[0].elements[0].value ;
 ... document.forms["formu"].elements["texte"].value ; }
</script>
```

Formulaires : Exemple

```
script type="text/javascript">
function verif(f)
{
 window.alert(f.texte.value) ;
 window.alert(f.sel.selectedIndex) ;
 window.alert(f.sel[f.sel.selectedIndex].te
 xt) ;
 window.alert(f.sel[f.sel.selectedIndex].va
 lue) ; window.alert(f.rad[0].checked) ;
 window.alert(f.chk.checked) ;
 return fals
}
</script>
```

Formulaire Inscription

Ecrive le code HTML/javascript de la page **exercice.html** qui réalise suivant.



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <script language="javascript">
      function Verifier (fo){
        if(document.fo.nom.value==="")
        {alert("Entrer votre nom!");}

        if(document.fo.prenom.value==="")
        {alert("Entrer votre prenom!");}

        if(document.fo.email.value==="")
        {alert("Entrer votre email!");}

        if((document.fo.choix[0].checked==false)&&
        (document.fo.choix[1].checked==false))
        {alert("specifier votre sexe!");}
      }
```

```html
      function reponse (fo2){

        if((document.fo2.check1.checked)==true &&
        (document.fo2.check2.checked) ==false &&
        (document.fo2.check3.checked) ==false)
        {alert("ok!");}
        else{
         alert("Désoler!");
        }
      }
      function liste (fo3){

        alert(document.fo3.list.selectedIndex +1);
    }

  </script>
```

```html
<body>
   <h1>Formulaire d'inscription</h1>
   <form name="fo">

      Nom: <input type="text" name="nom" value="" /><br>
      Prenom: <input type="text" name="prenom" value="" /><br>
      email: <input type="text" name="email" value="" /><br><br>
      sexe:<br>
      <input type="radio" name="choix" value="1" >Feminin<br><br>
      <input type="radio" name="choix" value="2" >Masculin<br><br>

      <input type="button" value="Envoyer" onClick="Verifier(fo)" />
   </form>
 <form name="fo2">

   <input type="checkbox" name="check1" value="1" >Choix num 1<br><br>
   <input type="checkbox" name="check2" value="2" >Choix num 2<br><br>
   <input type="checkbox" name="check3" value="3" >Choix num 3<br><br>
   <input type="button" value="Corriger" onClick="reponse(fo2)" />
</form>

<form name="fo3">
   Entrer votre choix:
   <select name="list">
      <option value="1">ELEMENT 1</option>
      <option value="2">ELEMENT 2</option>
      <option value="3">ELEMENT 3</option>
   </select>
   <input type="button" name="b" value="Quel est l'element retenu!" onClick="liste(fo3)" />
```

Boutons radio:

L'objet bouton radio possède les propriétés suivantes:

- Name: indique le nom du contrôle. tous les boutons portent le même nom
- Index: l'index ou le rang du bouton radio en commençant par 0
- Checked : indique l'etat en cours de l'element radio
- Defaultchecked: indique l'etat du bouton sélectionné par défaut
- Value: indique la valeur de l'element radio

Boutons case à cocher:

L'objet bouton case à cocher possède les propriétés suivantes:

- Name: indique le nom du contrôle. tous les cases à cocher portent un nom diffèrent.
- Checked : indique l'etat en cours de l'element case à cocher
- Defaultchecked: indique l'etat du bouton sélectionné par défaut
- Value: indique la valeur de l'element case à cocher

Liste de selection :

L'objet Liste de sélection possède les propriétés suivantes:

- Name: indique le nom de la Liste de sélection .
- length : indique le nombre d'element de la liste
- selectedIndex: l'indique le rang a partir de 0 de l'element de la liste qui a été sélectionné
- defaultselected: indique l'element de la Liste de sélectionné par défaut

DOM : Document Objet Model

- L'objet **Document** est hiérarchiquement placé sous l'objet **window**. Il permet de manipuler tous les objets qui sont inclus dans le documents HTML (**images, textes, formulaires...).**
- Il s'agit sans doute de l'objet le plus riche et le plus important
- Quand on parle de l'objet **document** il y a un autre terme qui surgit automatiquement, c'est le **DOM**.

Document Objet Model: DOM

DOM qui signifie **Document objet Model** est un outil qui permet **d'accéder** au document. Il s'agit plus précisément d'une **interface** qui standardise la manière **d'accès aux document structurés en balises** comme HTML ou XML. D'ailleurs, **chaque élément du document fait partie du model DOM.**

Arborescence et nœuds

- Nous avons déjà parlé de la **hiérarchie des objets** en Javascript. Il s'agit en fait de la manière dont sont présentés les différents éléments **constituant le contenu** (*un objet **parent** qui contient des objets **enfants** qui, à leurs tours, contiennent d'autres objets et ainsi de suite).*

- L'ensemble des ces objets constitue une **arborescence**
- Cet arbre est en fait une structure d'éléments qui représentent des ***nœuds***
- Il existe deux types de nœuds, **nœuds éléments et nœuds texte**:

> - Les nœuds éléments sont tout simplement les balises
> - Les nœuds textes sont les textes contenus dans un document HTML

getElementById

- Il existe d'autres méthodes qui permettent **d'accéder** plus facilement aux **différents objets** de tout type d'un document. Il s'agit d'ailleurs des méthodes les plus utilisés par les développeurs:
- ***getElementById():***

Cette méthode permet d'accéder à un objet quelconque en **l'identifiant** par la valeur de son attribut **id**. Nous avons déjà eu l'occasion de vois cet attribut en **CSS**. Sa valeur est **unique** dans tout le document. C'est cette valeur qui sera déclarée dans les **parenthèses de la méthode en tant que chaîne de caractères**.

- ***getElementsByTagName():*** Cette méthode permet d'accéder tous les objets correspondant à la balise (**tag**) déclarée entre les parenthèses en tant que chaîne de caractères. Elle retourne un **tableau** qui renferme tous les nœuds du type désigné.

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <style>
            img{
                width:600px;
            }
        </style>
        <script>
            function over(){
                document.getElementById("monimage").src="
                    images/image2.jpg";
            }
            function out(){
                document.images[0].src="images/image1.jpg";
            }
        </script>
    </head>
    <body>
        <img src="images/image1.jpg" id="monimage" onMouseOver="over
        ()" onMouseOut="out()" />
    </body>
```

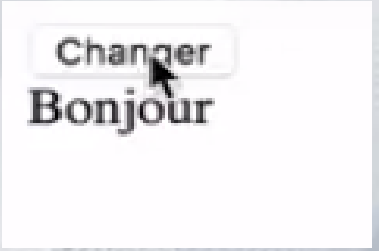
L’image sera changer lorsque la souris move sur l’image.

Attribut innerHTML

Nous avons pu écrire des **textes** dans des <u>champs de formulaire et des boites de dialogue</u>. Mais si on veut écrire n'importe où dans la balise **<body>** alors on peut le faire grâce à l'attribut **innerHTML**.

Exemple:

```
    <script>
        function f(){
            document.getElementById("div1").innerHTML="Salut à
                tous";
        }
    </script>
</head>
<body>
    <input type="button" value="Changer" onClick="f()" />
    <div id="div1">Bonjour</div>
</body>
```



Attribut innerHTML

Nous avons pu écrire des **textes** dans des champs de formulaire et des boites de dialogue. Mais si on veut écrire n'importe où dans la balise **<body>** alors on peut le faire grâce à l'attribut **innerHTML**.
Exemple:

```
<div id="contenu">Bonjour</div>
<script language="javascript">
   document.getElementById("contenu").innerHTML="Bonsoir";
</script>
```

Au chargement de la page, la balise ** ** identifiée par **contenu** contient le texte "Bonjour".
Une fois le script Javascript exécuté, le nœud texte "Bonjour" est remplacé par "**Bonsoir**" grâce à l'attribut **innerHTML**.

Méthode createElement()

La méthode **createElement()** permet de créer un nouveau nœud élément (balise).

Méthode createTextNode()

La méthode **createTextNode** permet de créer un nœud texte.

Méthode appendChild()

La méthode **appendChild** ajoute un enfant à l'objet sur lequel elle est appelée. L'enfant peut être un nœud élément ou un nœud texte

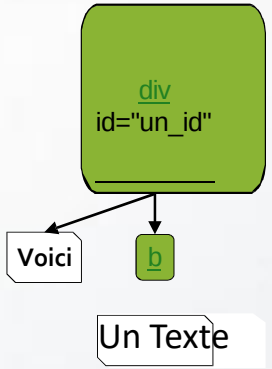
Méthode removeChild()

La méthode **removeChild()** supprime l'enfant passé en paramètre du parent sur lequel il est appelée.

Mot-clé this

Le mot clé **this** fait référence à l'objet sur lequel il est déclaré. Il permet de simplifier la désignation des objets à travers le DOM. Au lieux de cibler un élément à travers son ID ou ton tagName, **this** raccourcie visiblement la syntaxe.

- Identifier un élément HTML
```
<div id="un_id"></div>
```
- Accéder à un élément
```
e = document.getElementById("un_id");
```
- Construire une chaîne contenant du HTML
```
s = "Voici <b>un texte</b>";
```
- Modifier le contenu de l'élément
```
e.innerHTML = s;
```
- Interprétation « automatique » par le navigateur du nouveau contenu pour modifier le document



- Identifier un élément HTML
```
<div id="un_id"></div>
```
- Accéder à un élément
```
e = document.getElementById("un_id");
```
  - Créer un nœud de type « texte»
```
t1 = document.createTextNode('Voici ');
t2 = document.createTextNode('un texte');
```
- Créer un nouveau nœud de type « balise»
```
b = document.createElement('b');
```
- Construire des liens de parenté
- `e.appendChild(t1);`
- `b.appendChild(t2);`
- `e.appendChild(b);`

```
        <script>
            function over(ob){
                ob.src="images/image2.jpg";
            }
            function out(ob){
                ob.src="images/image1.jpg";
            }
        </script>
    </head>
    <body>
        <img src="images/image1.jpg" id="monimage" onMouseOver="over
        (this)" onMouseOut="out(this)" />
    </body>
```

AJAX/JQUERY

- Un **Site web** compose de plusieurs pages web, les **pages** contient plusieurs **informations.**

Il se trouve des fois que le fait de **cliquer sur un hyperlien**, ordonne au navigateur de charger un **nouveau document Web** avec une bonne partie de contenu similaire à celui qui figurait sur la page précédente.

Serte il y a du nouveau dans la page sur laquelle on s'est rendu, mais des éléments comme la bannière, le menu de navigation, le pied de pages... restent dans la plupart du **temps inchangés**, et pourtant ils ont été chargés a nouveau.

- Afin de **garder les éléments** qui ne changent pas et **recharger seulement le nouveau contenu** au sein de la page, les développeurs recouraient à des solutions comme **les frames** qui répartissent la page en plusieurs **zones** dont seulement quelques unes sont rechargées suite à **un clic sur un lien hypertexte ou l'envoi d'un formulaire**. Or, cette solution a des **limites** et, en plus, elle pose de **sérieux problèmes lors du référencement**.

En d'autre part, certains navigateurs arrivent à **identifier les éléments** qui ne changent pas d'une page à une autre et les gardent **en cache**.

De cette manière, ils sollicitent juste le nouveau contenu et **la navigation devient alors plus rapide** suite à un taux de transfert de données réduits entre **le client et le serveur**.

La solution magique qui a ouvert de nouveaux horizons aux développeurs s'appelle AJAX

AJAX : Asynchronous JavaScript and XML

AJAX :une technologie de développent web technique pour la création des applications web interactives.

- N'est pas un nouveau langage de programmation, mais une nouvelle façon d'utiliser les normes existantes.
- Communication client/serveur en mode **asynchrone**
  - l'échange de données avec un serveur, et la mise à jour des parties d'une page Web - **sans recharger la page entière.**
- Utilise XHTML pour le contenu,CSS pour la présentation, avecLe DOM et JavaScript pour afficher du contenu interactif.
- **Plusieurs APIs :**

JQUERY, PROTOTYPE , DOJO,YUI,

ANGULAR JS, EXT JS, …

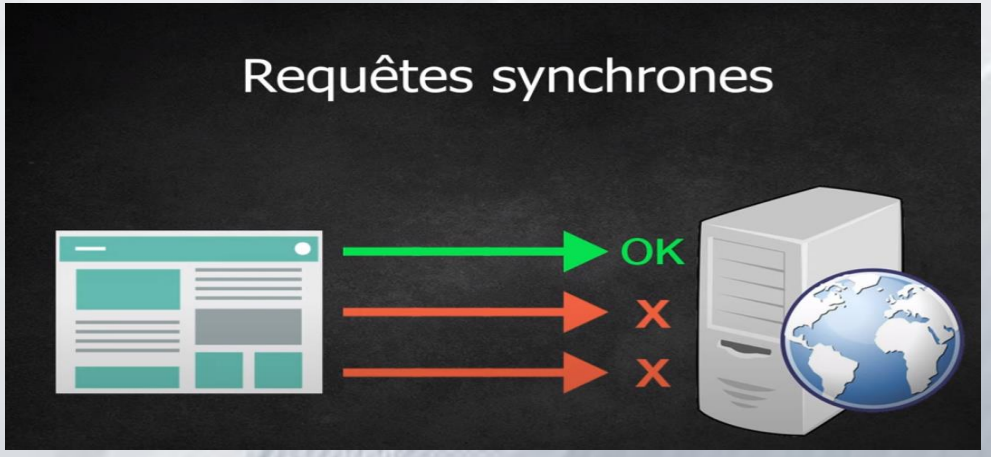
AJAX s'appuie sur les **technologies** suivantes:

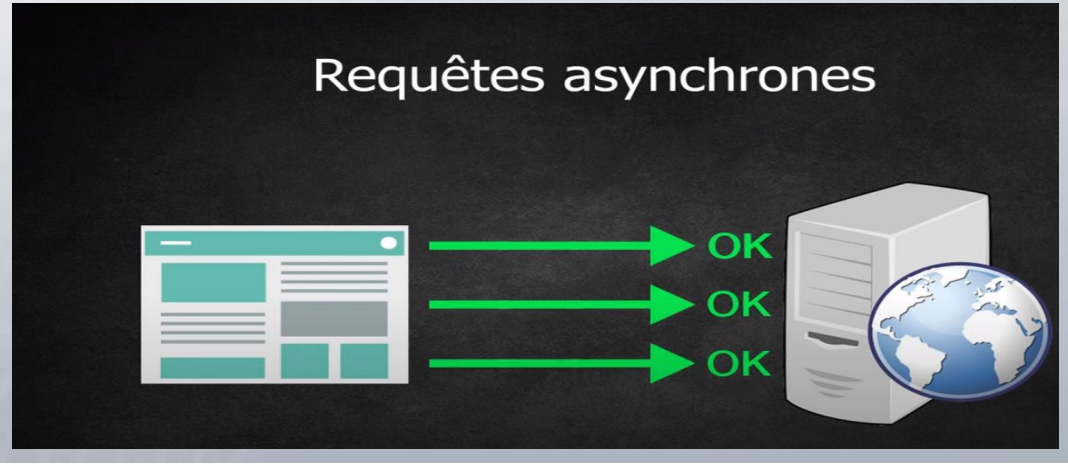
- **Javascript** et **DOM** pour le traitement.
- **XML** ou **JSON** pour l'extraction des données (le contenu à afficher).
- **HTML** et **CSS** pour la présentation.

AJAX est une technologie de navigateur Web (coté client) indépendant de la technologie coté serveur Web.

- Les applications Web **classiques** transmettent des informations vers et depuis le serveur utilisant des **requêtes synchrones**.

  - Cela signifie que vous remplissez un **formulaire,** cliquez sur **Envoyer,** et acheminés vers une nouvelle page avec de nouvelles informations à partir du serveur.
  - Avec AJAX, quand vous cliquez sur **envoyer,** JavaScript communique avec le **serveur, interprète** les résultats, et met à jour **l'écran actuel**.
  - Dans le sens le plus pur, l'utilisateur ne sais jamais que tout ce qui a même été transmis au serveur.



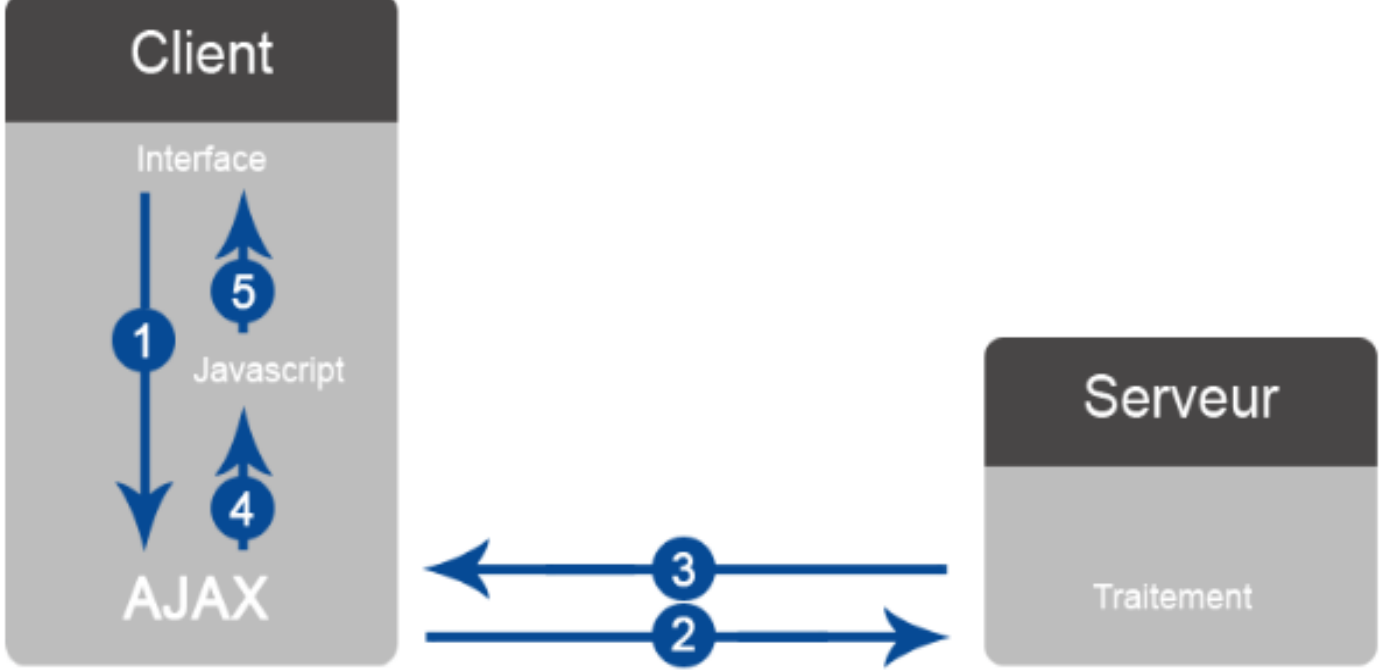


Utilité d'AJAX

Parmi les applications les plus courantes d'AJAX on trouve:

- Rechargement d'une (ou plusieurs) zone de la page Web sans être obligé de recharger celle-ci en entier (ce qui est d'ailleurs l’essence d'AJAX).
- Suggestion automatique lors de la saisi d'un champ, comme c'est le cas quand vous saisissez des mots clés sur le moteur de recherche Google.
- Sauvegarde automatique d'un texte dans la base de données lors de la saisie, sans pour autant quitter le champs dans lequel on est entrain d'écrire.
- Upload d'un fichier sur le serveur tout en visualisant l'état de progression du chargement.
- Affichage d'un contenu supplémentaire quand on atteint le bas de la page avec la barre de défilement, comme c'est le cas sur Youtube.
- Exécution d'opérations telles que la suppression ou la modification suite à un glisser déposer sans quitter la vue en cours.

Principe de fonctionnement



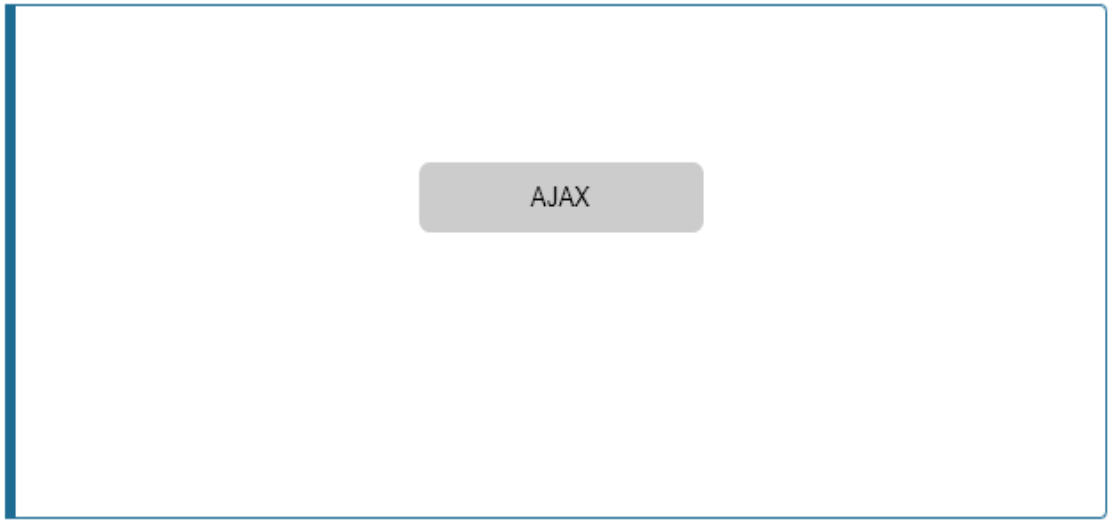
Déroulement d'un dialogue en AJAX entre le client et le serveur

1. Quand le client déclenche un événement, celui-ci et capturé par le script Javascript/AJAX.
2. A l'aide des méthodes d'AJAX, la requête est transférée au serveur.
3. Après traitement de la requête, le serveur renvoie la réponse (sous forme de texte, XML ou HTML) qui sera récupérée par AJAX.
4. AJAX fait appel aux méthodes Javascript (en particulier les méthodes qui implémentent le DOM) afin de traiter la réponse reçue.
5. Le script Javascript place alors la réponse à l'endroit prévu dans le document afin qu'elle soit intégrée au contenu et devenir ainsi visible.

L'API JQUERY

Principe de fonctionnement

- **Définition d'une fonction : jquery() ou $() (raccourci)**
- Accepte des paramètres (en général, un selecteur)
- $() Retourne **un objet JQuery**
- Est exécutée au cours du chargement de l'arbre DOM de la page
- Sinon, il faut utiliser **window.onload**
- **Utilisation de sélecteurs CSS et JQuery pour accéder et manipuler les éléments de l'arbre.**



Le message qui apparaît en dessous du bouton n'a pas été chargé avec la page, mais individuellement suite au clique sur le bouton.
La GIF animée est là pour indiquer au visiteur que les données sont entrain d'être chargées.

Sélection d’éléments

Possibilité de sélectionner :

- par type de bloc
- par identifiant #
- par classe .
- en filtrant sur les noms d’attributs
- en faisant référence aux positions relatives dans le DOM
- en ne récupérant qu’un seul élément parmi les objets sélectionnés
- en filtrant parmi les objets sélectionnés

Sélection d'éléments

Sélecteurs CSS

- Similaires aux éléments d'une feuille de style, ou XPath
- **Exemples :** #menuid, h2, .onglet, *, etc.
- **Exemple :** $ (« body»)
- **Exemple :** **$("div.main") //** tous les divs de classe main
- **Exemple :** sélectionneur de visibilité //

  si l'élément est visible **$("div:visible")**

// sinon $("div:hidden")

http://docs.jquery.com/DOM/Traversing/Selectors#CSS_Selectors

Sélecteurs Jquery

- Sélecteurs **supplémentaires** n'existant pas en CSS ou Xpath (ou raccourcis)

Les méthodes

- Chaque méthode **renvoie** l'objet courant
- **Acces/modification** au contenu d'un objet : **html()**
- **Exemple :**

```
$ ( ' body ' ) . html ( " hello world " ) ;
```

- Forcer l'utilisation de la fonction **jquery**() (à la place de $())

```
<html> <head>
<script type="text/javascript" src="jquery-1.11.3.js"></script>
</head>
<body>
He l l o World
<script> $('body').html("bonjour tout le monde" ) ; </script>
</body>
 </html>
```

Variables

- var nomVaraible=Valinitiale;
- **var body = $ (' body ') ;**
- **Variables standards :**

Exemples :**length** ou **size** : Nombre d'élements contenus dans l'objet JQuery courant:

```
$ ( function ( ) {
var bodycontent = $ ( " d i v " ) ;
$ ( '#id2 ' ). html ( " I l y a " + bodycontent . l e n g t h + " ) " ) ;
// afficher le nombre des <div>
}) ;
```

Modifier le code HTML

- .html('[contenu]') : remplacement du contenu d'un élément (les balises sont considérées comme des balises)
- .text('[contenu]') : remplacement du contenu d'un élément en considérant le tout comme du texte
- .after('[contenu]') : insertion du contenu après l'élément sélectionné
- .before('[contenu]') : insertion du contenu avant l'élément sélectionné
- .append('[contenu]') : insertion du contenu dans l'élément sélectionné à la suite des éléments existants
- .prepend('[contenu]') : insertion du contenu dans l'élément sélectionné avant les éléments existants

text() vs. html()

- **html()** : Remplacement du contenu d'un élément (les balises sont consideres comme des balises)
- **text()** : Remplacement du contenu d'un élément en considérant le tout comme du texte (les caractères < et > des balises sont remplaces par les entités XML (> et <)

```
$ ( function () {
var bodycontent = $ ( " d i v " ) ;
$ ( '#id2' ). html ( " Il ya " + bodycontent . length+ " div ) " ) ;
}) ;
```

Récupérer /modifier les propriétés CSS

- Récupération de la valeur de l'attribut CSS d'un élément :

.css('color') renvoie la couleur de l'élément

- Attribution d'une valeur à l'attribut CSS d'un élément :

.css('color','red') attribue la couleur rouge à l'élément

- Attribution d'une valeur à l'attribut CSS des éléments de classe **CSS «id»** en fonction de leur valeur actuelle à l'aide d'une fonction :

```
var tailleActuelle = parseInt($('#id').css("font-size"));
$('#id').css("fontsize",function(){
 return tailleActuelle+10;
});
```

Augmente de 10 points la taille de police de caractères des éléments de classe «id»

- Attribution d'un ensemble de valeurs à un ensemble d'attributs CSS d'un élément :

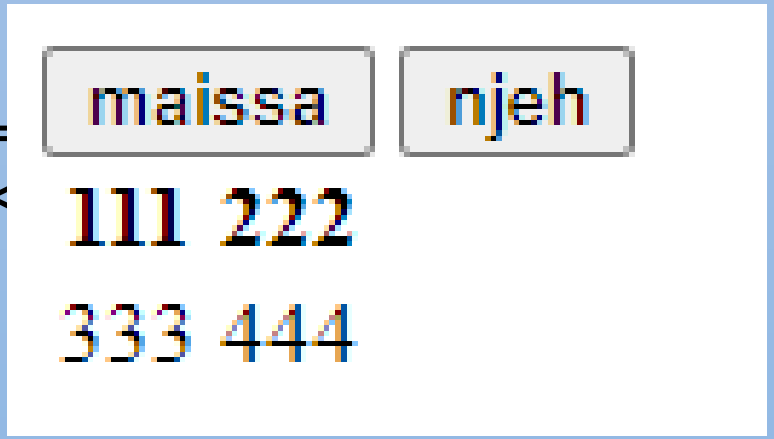
.css({'border' : '1px solid black', 'color' : 'red'})

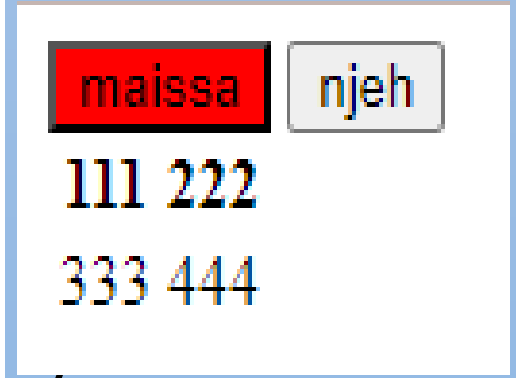
➢ attribue la couleur rouge à l'élément et lui ajoute une bordure noire.

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
</head>
<body>
<button id="btn1"> maissa</button>
<button id="btn2"> njeh</button>
<table id="tab">
   <tr id="entete">
      <th class="col1">111</th>
      <th id="col2">222</th>
   </tr>
   <tr >
      <td class=
      <td>444<
   </tr>
</table>
<script>
```

```jquery
$(document).ready(function(){
  $('#btn1').on("click",function()
 { //un seul attribute css
    $(this).css("background-color","red");
 });

$('#btn2').on("click",function()
 { //plusieurs attributs css
    $(this).css({
       "background-color":"blue",
       "font-size": "50px"
    });
```





Récupérer /modifier les propriétés CSS

Manipulation des classes CSS

- **.addClass('laClasse')**
  - Ajoute la classe CSS ***laClasse*** à l'élément.
- **.removeClass('laClasse')**
  - Retire la classe CSS ***laClasse*** à l'élément.
- **.toggleClass('laClasse')**
  - Ajoute la classe CSS ***laClasse*** à l'élément s'il ne l'a pas, la lui retire sinon.
- **.hasClass('laClasse')**
  - Renvoie **true** si l'élément a la classe CSS ***laClasse***, **false** sinon.

Les évènements

Un évènement est une action remplie par le navigateur, ou plus généralement l'utilisateur. on lance un bout de code quand une action est remplie.

- Un évènement qui pourra écouter la **souris** de l'utilisateur , ou bien écouter le **clavier de l'utilisateur** : quand il presse des touches du clavier; exécuter du code.

⓪ **Un évènement fait le lien entre une action remplie, et une fonction (du code) exécutée.**

```
$("#unDiv").click(function(){
  // Le code a exécuter !
});
```

L'API JQUERY

Les évènements : La souris

Fonction	Action
click()	Clic
dblclick()	Double-clic
hover()	Passage de la souris
mouseenter()	Rentrer dans un élément
mouseleave()	Quitter un élément
mousedown()	Presser un bouton de la souris
mouseup()	Relâcher un bouton de la souris
scroll()	Scroller (utiliser la roulette)

L'API JQUERY

Les évènements : Le clavier

Fonction	Action
keydown()	qui se lance lorsqu'une touche du clavier est enfoncée
keypress()	se lance lorsqu'on maintient une touche enfoncée
keyup()	se lance lorsqu'on relâche une touche préalablement enfoncée.

L'API JQUERY

Les évènements : Document /Fenêtre

Action	Fonction
Load()/ready()	Chargement du document
resize()	Changement de la taille de la fenêtre
scroll()	Défilement avec la barre de défilement
Unload()	A la fermeture document

L'API JQUERY

Les évènements : Formulaire

Action	Fonction
submit()	Envoi du formulaire
change()	Changement de la valeur d'un champ de formulaire
focus()	Mettre le focus dans un champ de formulaire
blur()	Perte du focus dans un champ de formulaire

Les effets

Les effets :Afficher/Cacher

- On peut afficher / cacher un élément HTML respectivement avec les deux fonctions : **show()** et **hide()** , **toggle()** jouent le double rôles

Chargement du document

```
<script>
$(document).ready(function(){
  $("#hide").click(function(){
    $("p").hide();   });
  $("#show").click(function(){
    $("p").show();    });
  });
  </script>
</head>
<body>
<button id="hide">Cacher</button>
<button id="show">Afficher</button>
<p> Le message a afficher /cahcer</p>
</html>
```

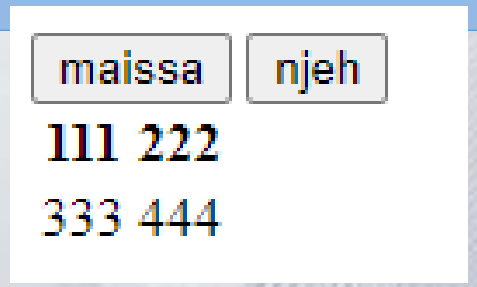
L'événement ready se produit lorsque le DOM (modèle objet de document) a été chargé. Étant donné que cet événement se produit une fois le document prêt, c'est un bon endroit pour avoir tous les autres événements et fonctions jQuery.

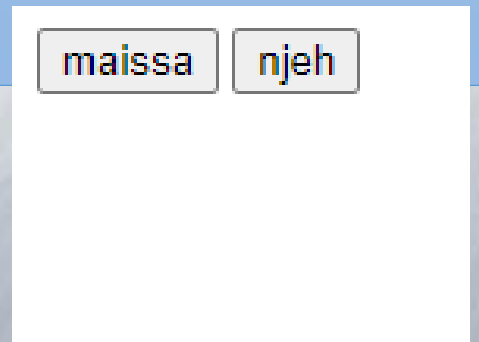
⓪ Le paramètre de vitesse optionnel spécifie la vitesse d'afficher/cacher et peut prendre les valeurs suivantes: **"slow"** , **"fast"**, ou une valeur en **millisecondes**.

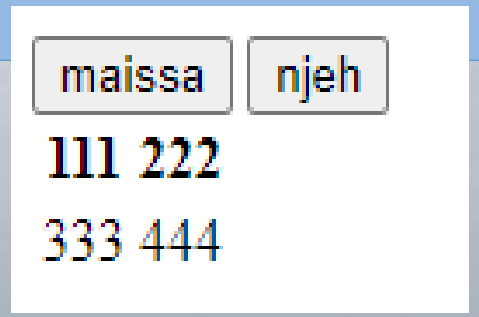
```
$("button").click(function(){
    $("p").hide(1000);});
```

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
</head>
<body>
<button id="btn1"> maissa</button>
<button id="btn2"> njeh</button>
<table id="tab">
   <tr id="entete">
      <th class="col1">111</th>
      <th id="col2">222</th>
   </tr>
   <tr >
      <td class="col1">333</td>
      <td>444</td>
   </tr>
</table>
<<script>
$(document).ready(function(){
 $('#btn1').on("click",function()
 {
   $('#tab').hide();
 });
 $('#btn2').on("click",function()
 {
   $('#tab').show();
 });
});
</script>
</body>
</html>
```

1. Lorsque je clique sur le bouton maissa ‘function hide’ alors le tableau sera cacher
2. si je clique sur le bouton njeh ‘function show’ le tableau s’affiche







L'API JQUERY

Les effets :l'hauteur

- Les effets sur la hauteur sont assurés par les fonctions: **slideUp()** , **slideDown()** , **slideToggle()**
- **slideToggle()** bascule entre les deux méthodes slideDown() et slideUp()

```
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideUp("slow");
  });
});
</script>
 <style>
#panel, #flip {
  padding: 5px;
  text-align: center;
border: solid 1px #c3c3c3;
}
#panel {
  padding: 50px;
}
</style>
</head>
<body>
 <div id="flip">Click to slide up
panel</div>
<div id="panel">Hello world!</div>
</body>
</html>
```

L'API JQUERY

Les effets: opacité

- Modifier l'opacité d'une façon progressive
- jQuery offre les méthodes suivantes :
  - fadeIn()
  - fadeOut()
  - fadeToggle()
  - fadeTo()

```
$("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

```
$("button").click(function(){
$("#div1").fadeTo("slow", 0.15);
$("#div2").fadeTo("slow", 0.4);
});
```

```
$("button").click(function(){
    $("#div1").fadeOut();
});
```

```
$("button").click(function(){
    $("#div1").fadeToggle();
```

L'API JQUERY

Les effets: Animation

- Créer une animation par la modication de certaines propriétes de style CSS(déplacement d'un block par exemple)
- La fonction animate() a le syntaxe suivant :

```
$(selecteur).animate({params},speed,callback);
```

  - Le paramètre **params** définit les propriétés CSS à être animés.

```
$("button").click(function(){
  $("div").animate({
    left: '250px',
    opacity: '0.5',
    height: '150px',
    width: '150px'
  });
});
```

L'API JQUERY

Les effets: Animation

Les valeurs à modifier peuvent être relatives aux valeurs actuelles

- Ceci est assuré par l'ajout de += ou-= devant la valeur :

```
$("button").click(function(){
  $("div").animate({
    left: '250px',
    height: '+=150px',
    width: '+=150px'
  });
});
```

On peut créer une succession d'animation

```
$(document).ready(function(){
  $("button").click(function(){
    var div = $("div");
    div.animate({left: '100px'},
"slow");
    div.animate({fontSize: '3em'},
"slow");
  });
});
```

L'API JQUERY

Les effets: Arrêt des animation

- Pour arrêter les animations on utilise la function **stop()**

```
$(selecteur).stop(stopAll,goToEnd);
```

- Le paramètre optionnel **stopAll** spécifie si toutes les animations seront arrêtées. Par défaut est false, ce qui signifie que seulement l'animation courante sera arrêtée,
- Le paramètre optionnel **goToEnd** spécifie si on complète ou on l'arête immédiatement . Par défaut est false.
- Par défaut, stop() arête l'animation courante appliquée sur l'élément sélectionné.

L'API JQUERY

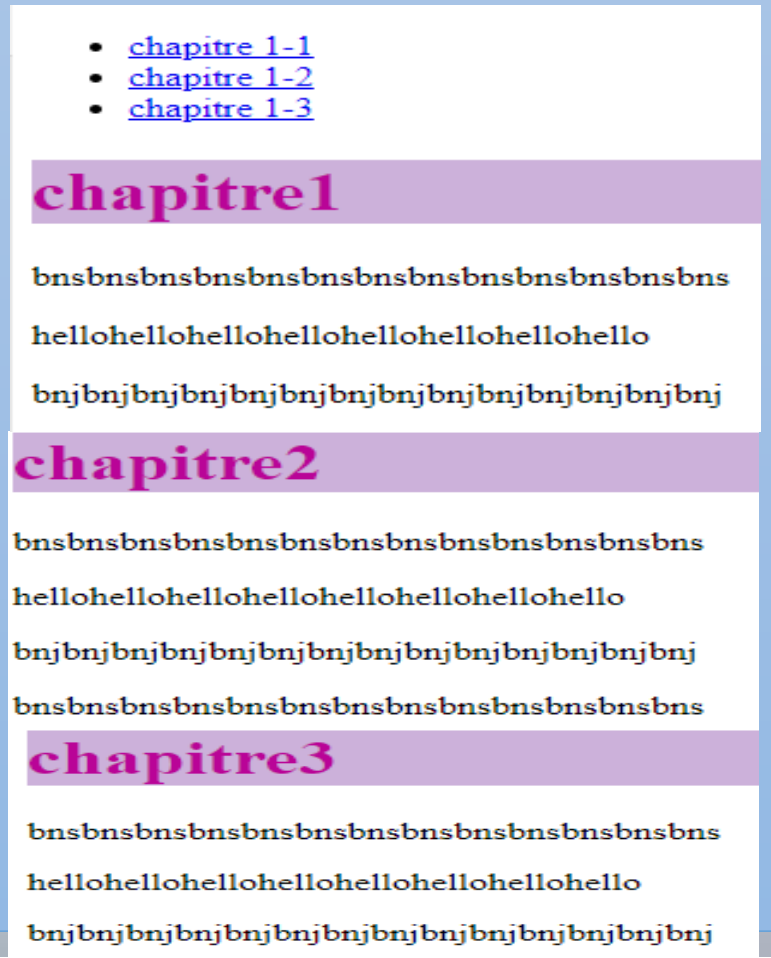
Les effets: l'enchainement des effets

- Il existe une technique appelée l'enchainement , qui nous permet d'exécuter des commandes multiples jQuery, l'un après l'autre, sur le même élément.
- Pour l'enchainement d'une action, on ajoute simplement l'action à l'action précédente
- L'exemple suivant enchaine les methods css(), slideUp(), et slideDown() ensemble. L'élement "p1" changement au rouge d'abord, il glisse vers le haut, puis il glisse vers le bas:

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

```
$("#p1").css("color", "red")
  .slideUp(2000)
  .slideDown(2000);
```

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
</head>
<body>
<ul>
<li ><a href="#c11">chapitre 1-1</a></li>
<li ><a href="#c12">chapitre 1-2</a></li>
<li ><a href="#c13" >chapitre 1-3</a></li>
</ul>

<h1 id="c11">chapitre1</h1>
<p>bnsbnsbnsbnsbnsbnsbnsbnsbnsbnsbnsbnsbns</p>
……
<h1 id="c12">chapitre2</h1>
<p>bnjbnjbnjbnjbnjbnjbnjbnjbnjbnjbnjbnj</p>
……
<h1 id="c13">chapitre3</h1>
<p>hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh</p>
……
```

```jquery
<script>
$(document).ready(function() {
$("li a").on("click",function(e){
 $("html, body").animate({
   scrollTop: $(
$(this).attr("href")).offset().top
 },2000);
})
});
</script>
</body>
</html>
```