

# Cours JavaScript

# Généralités

Avec les événements et surtout leur gestion, nous abordons le côté "**magique**" de Javascript. En Html classique, il y a un événement que vous connaissez bien. C'est le clic de la souris sur un lien pour vous transporter sur une autre page Web. Hélas, c'est à peu près le seul. Heureusement, Javascript va en ajouter une bonne dizaine, pour votre plus grand plaisir. Les événements Javascript, associés aux fonctions, aux méthodes et aux formulaires, ouvrent grand la porte pour une réelle **interactivité** de vos pages.



onChange (après modification réussie)

Est prévu pour le cas où un utilisateur appuie sur le bouton stop du navigateur, alors que tous les graphiques ne sont pas encore chargés.

# Exemple

```
<html>
<head>
<title>Test</title>
</head>
<body>
<form name="Test" action="">
  <textarea rows="5" cols="40"
onChange="alert(this.value)">Modifiez ce texte et
cliquez ailleurs!
  </textarea>
</body>
</html>
```

## onClick (en cliquant)

Au cas où l'utilisateur clique sur un élément.

### Exemple

```
<html><head><title>Test</title>
</head><body>
<form name="Test" action="">
  <input size="30" name="sortie" readonly><br>
  <input type="button" value="Dernière
mise à jour"
onClick="this.form.sortie.value=document.lastModified
">
  </form>
</body></html>
```

# onDbClick (en double-cliquant)

Au cas où l'utilisateur double-clique sur un élément.

## Exemple

```
<html><head><title>Test</title>
</head><body>
<form name="calculer" action="">
Valeur: <input size="10" name="valeur">
<input type="button" value="Double-clic = puissance 2"
onDbClick="document.calculer.valeur.value=document
.calculer.valeur.value*document.calculer.valeur.value
">
</form>
</body></html>
```

# onError (en cas d'erreur)

Est approprié pour capter les messages d'erreur et les remplacer par des messages personnalisés. Attention pourtant, les erreurs ne sont pas pour autant écartées!!  
onError est conçu avant tout pour traiter les erreurs intervenant lors du chargement de graphiques..

## Exemple

```
/head><body>  
  
</body></html>
```

# onMousemove (en bougeant la souris)

Entre en action si l'utilisateur bouge la souris indépendamment du fait que la touche en soit appuyée ou non.

## Exemple

```
<html><head><title>Test</title>
<script language="JavaScript">
function controlesouris(Element) {
  var Pos = Element.offsetLeft + "/" + Element.offsetTop;
  window.status = Pos;
  return true;
}
</script>
</head><body>
<p onMousemove="controlesouris(this)">Ici un petit texte</p>
</body></html>
```

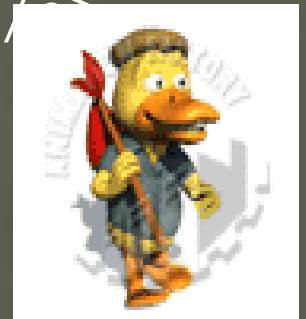


## onMouseout (en quittant l'élément avec la souris)

Entre en action quand l'utilisateur passe avec la souris sur un élément et le quitte ensuite.

### Exemple

```
<html><head>
<title>Test</title>
</head>
<body>
<a href="http://actuel.fr.selfhtml.org/nouvelles.htm"
onMouseout="alert('Vous devriez aller voir les
nouveau&eacute;s!')"><b>Quoi de neuf?</b></a>
</body>
</html>
```



## onMouseover (en passant sur l'élément avec la souris)

### Exemple

```
<html><head><title>Test</title>
</head><body>
<h1 id="Test"
  onMouseover="this.innerHTML='Vous voyez?'
  onMouseout="this.innerHTML='Je suis
    dynamique'">Je suis dynamique</h1>
</body></html>
```

# L'OBJET ARRAY



# Généralités

L'objet Array (ou tableaux) est une liste d'éléments indexés dans lesquels on pourra ranger (écrire) des données ou aller reprendre ces données (lire).



# Généralisation

Évènement	Action de l'utilisateur	Balise HTML
onBlur	Désélection d'un champ de saisie	INPUT
onFocus	Activation d'un champ de saisie	INPUT
onSelect	Sélection d'un champ de saisie	INPUT
onChange	Changement de valeur d'un champs Changement de valeur d'une sélection	INPUT SELECT
onmouseover	Passage de la souris sur un lien	A
onClick	Clique sur un lien Clique sur un élément de formulaire	A FORM
onSubmit	Soumission de formulaire	FORM
onLoad	Chargement de la page	BODY
onUnload	Sortie du document	BODY

Objets	Gestionnaires d'événement disponibles
Fenêtre	onLoad, onUnload
Lien hypertexte	onClick, onMouseOver, onMouseOut
Élément de texte	onBlur, onChange, onFocus, onSelect
Élément de zone de texte	onBlur, onChange, onFocus, onSelect
Élément bouton	onClick
Case à cocher	onClick
Bouton Radio	onClick
Liste de sélection	Blur, onChange, onFocus
Bouton Submit	onClick
Bouton Reset	onClick

# Tableau à une dimension

Pour faire un tableau, il faut procéder en deux étapes :

d'abord construire la structure du tableau. A ce stade, les éléments du tableau sont vides.

ensuite affecter des valeurs dans les cases ainsi définies.

On commence par définir le tableau :

```
nom_du_tableau = new Array (x);
```

où x est le nombre d'élément du tableau.

*On notera que, "le nombre d'éléments est limité à 255. Cette restriction ne figure pas dans la documentation de Netscape mais elle a été constatée expérimentalement." Source : Javascript de Michel Dreyfus Collection Mégapoché.*

Ensuite, on va alimenter la structure ainsi définie :

`nom_du_tableau[i] = "élément";`

où *i* est un nombre compris entre 0 et *x* moins 1.



Exemple : un carnet d'adresse avec 3 personnes  
construction du tableau :

```
carnet = new Array(3);
```

ajout des données :

```
carnet[0]="Dupont";
```

```
carnet[1]="Médard";
```

```
carnet[2]="Van Lancker";
```

pour accéder un élément, on emploie :

```
document.write(carnet[2]);
```

On notera ici que les données sont bien visibles  
au lecteur un peu initié (view source).

## Remarques :

Quand on en aura l'occasion, il sera pratique de charger le tableau avec une boucle. Supposons que l'on ait à charger 50 images. Soit on le fait manuellement, il faut charger 0.gif, 1.gif, 2.gif... Soit on utilise une

boucle du style :

```
Function gifs() {  
  gif = new Array(50);  
  for (var=i;i<50;i++)  
  {gif[i] =i+".gif";}  
}
```

Javascript étant un langage peu typé, il n'est pas nécessaire de déclarer le nombre d'élément du tableau (soit x). Javascript prendra comme nombre d'éléments, le nombre i le plus élevé lors de "l'alimentation" de la structure (en fait i + 1). Ainsi la formulation suivante serait aussi correcte pour un tableau à 3 dimensions.

```
carnet = new Array();  
carnet[2]="Van Lancker";
```



# Méthodes

join(separateur)	concatène les cases du tableau avec le separateur (JavaScript 1.1, ECMA)
pop()	dépile et retourne le dernier élément du tableau (Navigator 4)
push(val,...)	ajoute des éléments au tableau (Navigator 4)
reverse()	renverse le tableau (JavaScript 1.1, ECMA)
shift()	supprime et retourne le 1er élément (Navigator 4)
slice(deb, fin)	retourne une portion du tableau (JavaScript 1.2)
sort()	tri des éléments (JavaScript 1.1, ECMA)
splice(deb, nbel, val, ...)	retire des éléments du tableau (Navigator 4)
toString()	conversion du tableau en chaîne (JavaScript 1.1, ECMA)
unshift(val,...)	insère des éléments (Navigator 4)

# Exemples

var tab = new Array();	vide
tab.push('pomme','ananas','poire','cerise')	pomme ananas poire cerise
tab.push('abricot','raisin')	pomme ananas poire cerise abricot raisin
tab.reverse()	raisin abricot cerise poire ananas pomme
fruit = tab.pop()	raisin abricot cerise poire ananas
	fruit = pomme
fruit = tab.shift()	abricot cerise poire ananas
	fruit = raisin
tab.unshift('kiwi')	kiwi abricot cerise poire ananas
liste_fruits = tab.splice(2,4)	kiwi abricot
	liste_fruits = ["cerise", "poire", "ananas"]
tab.push('pomme','ananas','poire','cerise')	kiwi abricot pomme ananas poire cerise
tab.sort()	abricot ananas cerise kiwi poire pomme
liste_fruits = tab.join(' - ')	liste_fruits = abricot - ananas - cerise - kiwi - poire - pomme
liste_fruits = tab.slice(1,4)	abricot ananas cerise kiwi poire pomme
	liste_fruits = ["ananas", "cerise", "kiwi"]

# Tableau à deux dimensions

On peut créer des tableaux à deux dimensions (et plus encore) par un petit artifice de programmation.

On déclare d'abord un tableau à 1 dimension de façon classique :

```
nom_du_tableau = new Array (x);
```

Ensuite, on déclare chaque élément du tableau comme un tableau à 1 dimension :

```
nom_du_tableau[i] = new Array(y);
```

Pour un tableau de 3 sur 3 :

Tarif	T. Small	T. Médium	T. Large
Chemises	1200	1250	1300
Polos	800	850	900
T-shirts	500	520	540

```
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";
```

# Les objets du navigateur

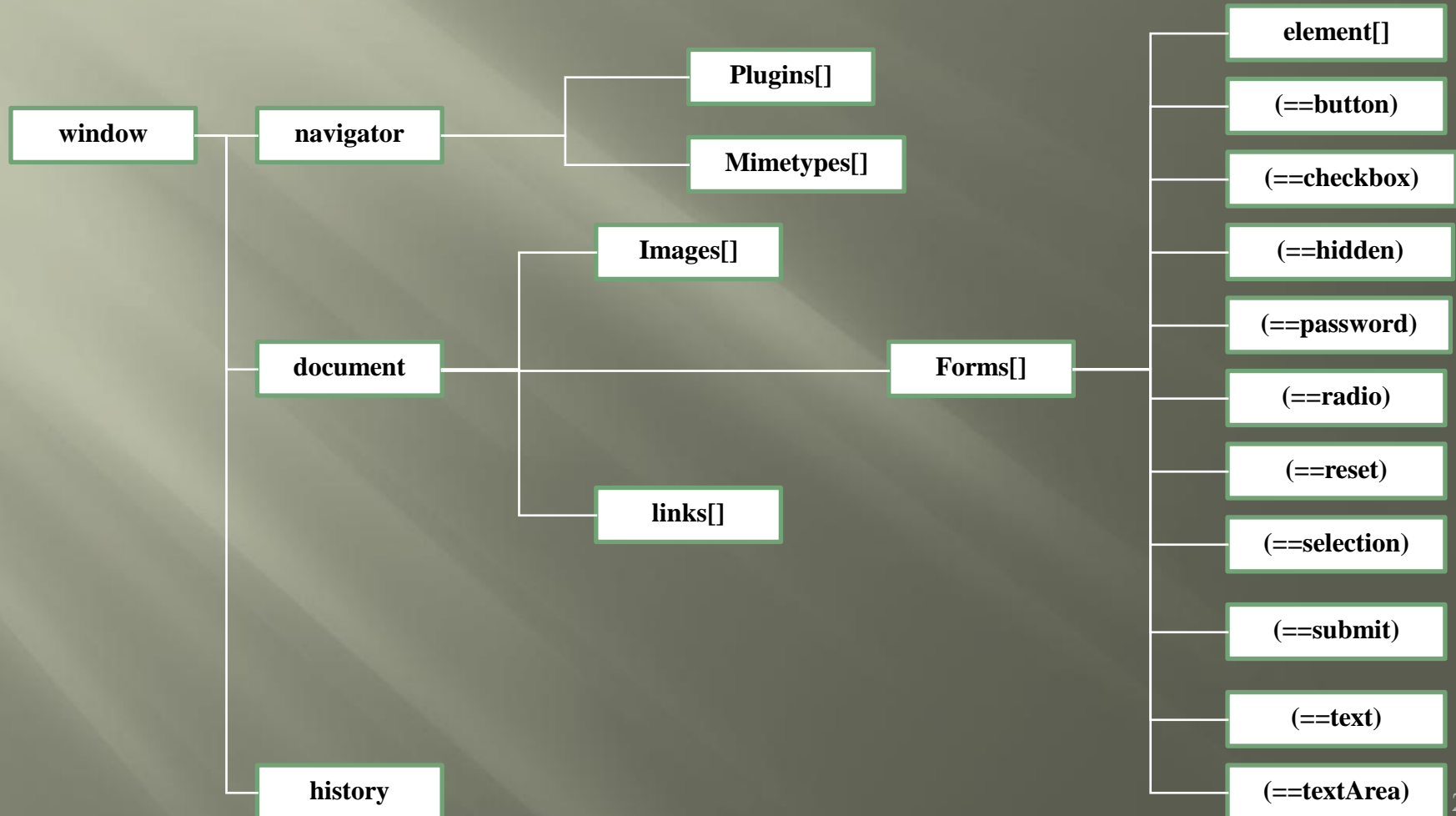


- ▣ Au chargement du navigateur, instantiation automatique de ses objets
- ▣ Agir sur l'état du navigateur et sur les zones des Documents
- ▣ Les descendants d'un objet correspondent à ses propriétés
- ▣ Pas d'héritage comme dans l'OO

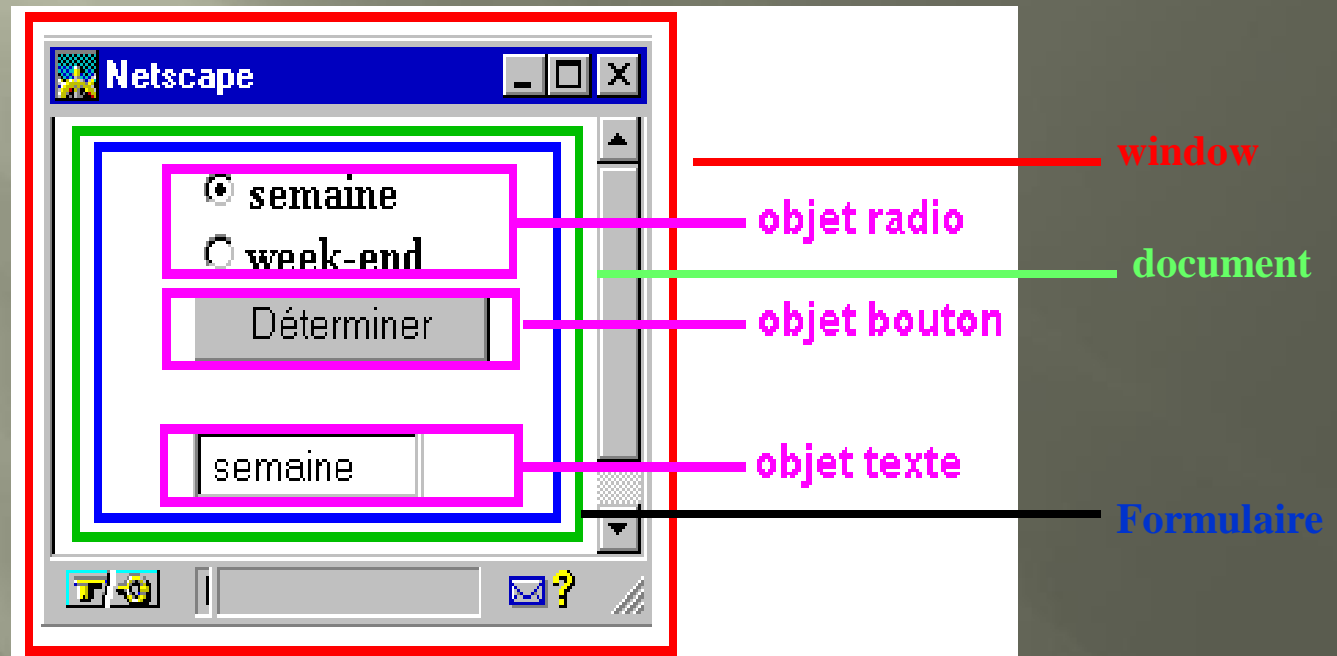




# Structure des objets de navigateur



# Exemple

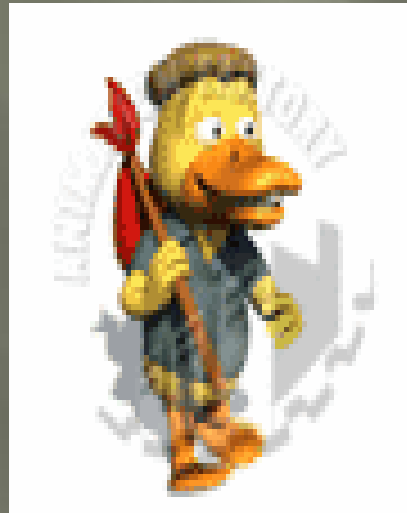


# Objet Navigator

- ❑ Moins souvent utilisé que l'objet window
- ❑ Une utilisation classique consiste à différencier un browser Netscape d'un browser Microsoft
- ❑ Propriétés:

Propriété	Description
appName	Nom du browser
appVersion	Version du browser
Langague	Langage utilisée: fr, en
appCodeName	Nom du code (ex: Mozilla)
userAgent	Chaine envoyée au serveur lors d'une requete HTTP
Plugins	Tableau de tous les plug-ins installés sur le browser
Platform	Type de la machine (ex: Win32, MacPPC,...)
mimeTypes	Tableau de tous les types MIME reconnus par le browser

# Les Formulaires



# GÉNÉRALITÉ

Avec Javascript, les formulaires Html prennent une toute autre dimension. N'oublions pas qu'en Javascript, on peut accéder à chaque élément d'un formulaire pour, par exemple, y aller lire ou écrire une valeur, noter un choix auquel on pourra associer un gestionnaire d'événement... Tous ces éléments renforceront grandement les capacités interactives de vos pages.

Mettons au point le vocabulaire que nous utiliserons. Un formulaire est l'élément Html déclaré par les balises `<FORM></FORM>`. Un formulaire contient un ou plusieurs éléments que nous appellerons des contrôles (widgets). Ces contrôles sont notés par exemple par la balise `<INPUT TYPE= ...>`.

# DÉCLARATION D'UN FORMULAIRE



La déclaration d'un formulaire se fait par les balises `<FORM>` et `</FORM>`. Il faut noter qu'en Javascript, l'attribut `NAME="nom_du_formulaire"` a toute son importance pour désigner le chemin complet des éléments. En outre, les attributs `ACTION` et `METHOD` sont facultatifs pour autant que vous ne faites pas appel au serveur.

# Le contrôle ligne de texte

La zone de texte est l'élément d'entrée/sortie par excellence de Javascript. La syntaxe Html est  
`<INPUT TYPE="text"  
NAME="nom" SIZE=x  
MAXLENGTH=y>` pour un champ de saisie d'une seule ligne, de longueur x et de longueur maximale de y.

# L'objet text possède trois propriétés :

Propriété	Description
name	indique le nom du contrôle par lequel on pourra accéder.
defaultvalue	indique la valeur par défaut qui sera affichée dans la zone de texte.
value	indique la valeur en cours de la zone de texte. Soit celle tapée par l'utilisateur ou si celui-ci n'a rien tapé, la valeur par défaut



## Lire une valeur dans une zone de texte



Voici un exemple que nous détaillerons :

Voici une zone de texte. Entrez une valeur et appuyer sur le bouton pour contrôler celle-ci.

Haut du formulaire

Bas du formulaire

Bas du formulaire

Le script complet est celui-ci :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function controle(form1) {
var test = document.form1.input.value;
alert("Vous avez tapé : " + test);
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form1">
<INPUT TYPE="text" NAME="input"
VALUE=""><BR>
<INPUT TYPE="button" NAME="bouton"
VALUE="Contrôler" onClick="controle(form1)">
</FORM>
</BODY>
</HTML>
```

## Ecrire une valeur dans une zone de texte

Entrez une valeur quelconque dans la zone de texte d'entrée. Appuyer sur le bouton pour afficher cette valeur dans la zone de texte de sortie.

Zone de texte d'entrée

Zone de texte de sortie

## Voici le code :

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function afficher(form2) {
var testin =document. form2.input.value;
document.form2.output.value=testin
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="form2">
<INPUT TYPE="text" NAME="input" VALUE="">
Zone de texte d'entrée <BR>
<INPUT TYPE="button" NAME="bouton"
VALUE="Afficher" onClick="afficher(form2)"><BR>
<INPUT TYPE="text" NAME="output" VALUE="">
Zone de texte de sortie
</FORM>
</BODY>
</HTML>
```

# Les boutons radio

Les boutons radio sont utilisés pour noter un choix, et seulement un seul, parmi un ensemble de propositions.

Propriété	Description
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'état en cours de l'élément radio
defaultchecked	indique l'état du bouton sélectionné par défaut.
value	indique la valeur de l'élément radio.

## Prenons un exemple :

```
<HTML>
<HEAD>
<SCRIPT language="javascript">
function choixprop(form3) {
if (form3.choix[0].checked) { alert("Vous avez choisi la proposition " +
form3.choix[0].value) };
if (form3.choix[1].checked) { alert("Vous avez choisi la proposition " +
form3.choix[1].value) };
if (form3.choix[2].checked) { alert("Vous avez choisi la proposition " +
form3.choix[2].value) };
}
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix :
<FORM NAME="form3">
<INPUT TYPE="radio" NAME="choix" VALUE="1">Choix numéro
1<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="2">Choix numéro
2<BR>
<INPUT TYPE="radio" NAME="choix" VALUE="3">Choix numéro
3<BR>
<INPUT TYPE="button" NAME="but" VALUE="Quel est votre choix ?"
onClick="choixprop(form3)">
</FORM>
</BODY>
</HTML>
```

# Les boutons case à cocher (checkbox)

Les boutons case à cocher sont utilisés pour noter un ou plusieurs choix (pour rappel avec les boutons radio un seul choix) parmi un ensemble de propositions. A part cela, sa syntaxe et son usage est tout à fait semblable aux boutons radio sauf en ce qui concerne l'attribut name.

Propriété	Description
name	indique le nom du contrôle. Toutes les cases à cocher portent un nom différent.
checked	indique l'état en cours de l'élément case à cocher.
defaultchecked	indique l'état du bouton sélectionné par défaut.
value	indique la valeur de l'élément case à cocher.

Entrez votre choix :

Il faut sélectionner les numéros 1,2 et 4 pour avoir la bonne réponse.

Haut du formulaire

☐

Choix numéro 1

☐

Choix numéro 2

☐

Choix numéro 3

☐

Choix numéro 4



```
<HTML>
<HEAD>
<script language="javascript">
function reponse(form4) {
if ( (form4.check1.checked) == true &&
(form4.check2.checked) == true &&
(form4.check3.checked) == false &&
(form4.check4.checked) == true)
{ alert("C'est la bonne réponse! ") }
else
{alert("Désolé, continuez à chercher.")}
}
</SCRIPT>
</HEAD>
<BODY>
```

## Entrez votre choix :

```
<FORM NAME="form4">
<INPUT TYPE="CHECKBOX" NAME="check1"
VALUE="1">Choix numéro 1<BR>
<INPUT TYPE="CHECKBOX" NAME="check2"
VALUE="2">Choix numéro 2<BR>
<INPUT TYPE="CHECKBOX" NAME="check3"
VALUE="3">Choix numéro 3<BR>
<INPUT TYPE="CHECKBOX" NAME="check4"
VALUE="4">Choix numéro 4<BR>
<INPUT TYPE="button"NAME="but"
VALUE="Corriger" onClick="reponse(form4)">
</FORM>
</BODY>
</HTML>
```

# Liste de sélection



Le contrôle liste de sélection vous permet de proposer diverses options sous la forme d'une liste déroulante dans laquelle l'utilisateur peut cliquer pour faire son choix. Ce choix reste alors affiché.

La boîte de la liste est créée par la balise `<SELECT>` et les éléments de la liste par un ou plusieurs tags `<OPTION>`. La balise `</SELECT>` termine la liste.

Propriété	Description
name	indique le nom de la liste déroulante.
length	indique le nombre d'éléments de la liste. S'il est indiqué dans le tag <SELECT>, tous les éléments de la liste seront affichés. Si vous ne l'indiquez pas un seul apparaîtra dans la boite de la liste déroulante.
selectedIndex	indique le rang à partir de 0 de l'élément de la liste qui a été sélectionné par l'utilisateur.
defaultselected	indique l'élément de la liste sélectionné par défaut. C'est lui qui apparaît alors dans la petite boite.

Un petit exemple comme d'habitude :

Entrez votre choix :

Elément 1



```
<HTML>
<HEAD>
<script language="javascript"> function liste(form5) {
alert("L'élément " + (form5.list.selectedIndex + 1)); }
</SCRIPT>
</HEAD>
<BODY>
Entrez votre choix : <FORM NAME="form5">
<SELECT NAME="list">
<OPTION VALUE="1">Elément 1
<OPTION VALUE="2">Elément 2
<OPTION VALUE="3">Elément 3
</SELECT>
<INPUT TYPE="button" NAME="b" VALUE="Quel est l'élément retenu?"
onClick="liste(form5)"> </FORM>
</BODY>
</HTML>
```

# Le contrôle textarea (pour les bavards)

L'objet textarea est une zone de texte de plusieurs lignes.

La syntaxe Html est :

```
<FORM>
```

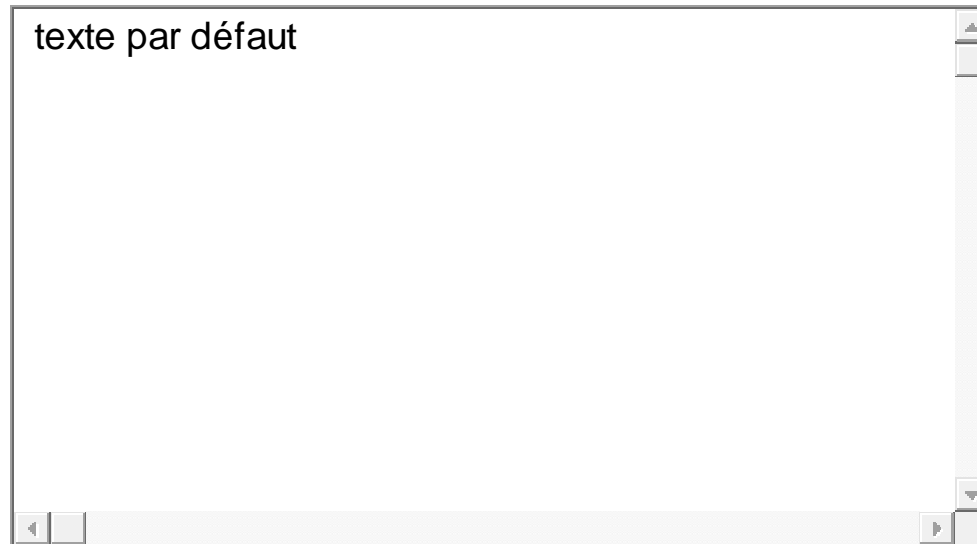
```
<TEXTAREA NAME="nom" ROWS=x  
COLS=y>
```

```
texte par défaut
```

```
</TEXTAREA>
```

```
</FORM>
```

où ROWS=x représente le nombre de lignes et  
COLS=y représente le nombre de colonnes.



L'objet textarea possède plusieurs propriétés :

Propriété	Description
name	indique le nom du contrôle par lequel on pourra accéder.
defaultvalue	indique la valeur par défaut qui sera affichée dans la zone de texte.
value	indique la valeur en cours de la zone de texte. Soit celle tapée par l'utilisateur ou si celui-ci n'a rien tapé, la valeur par défaut



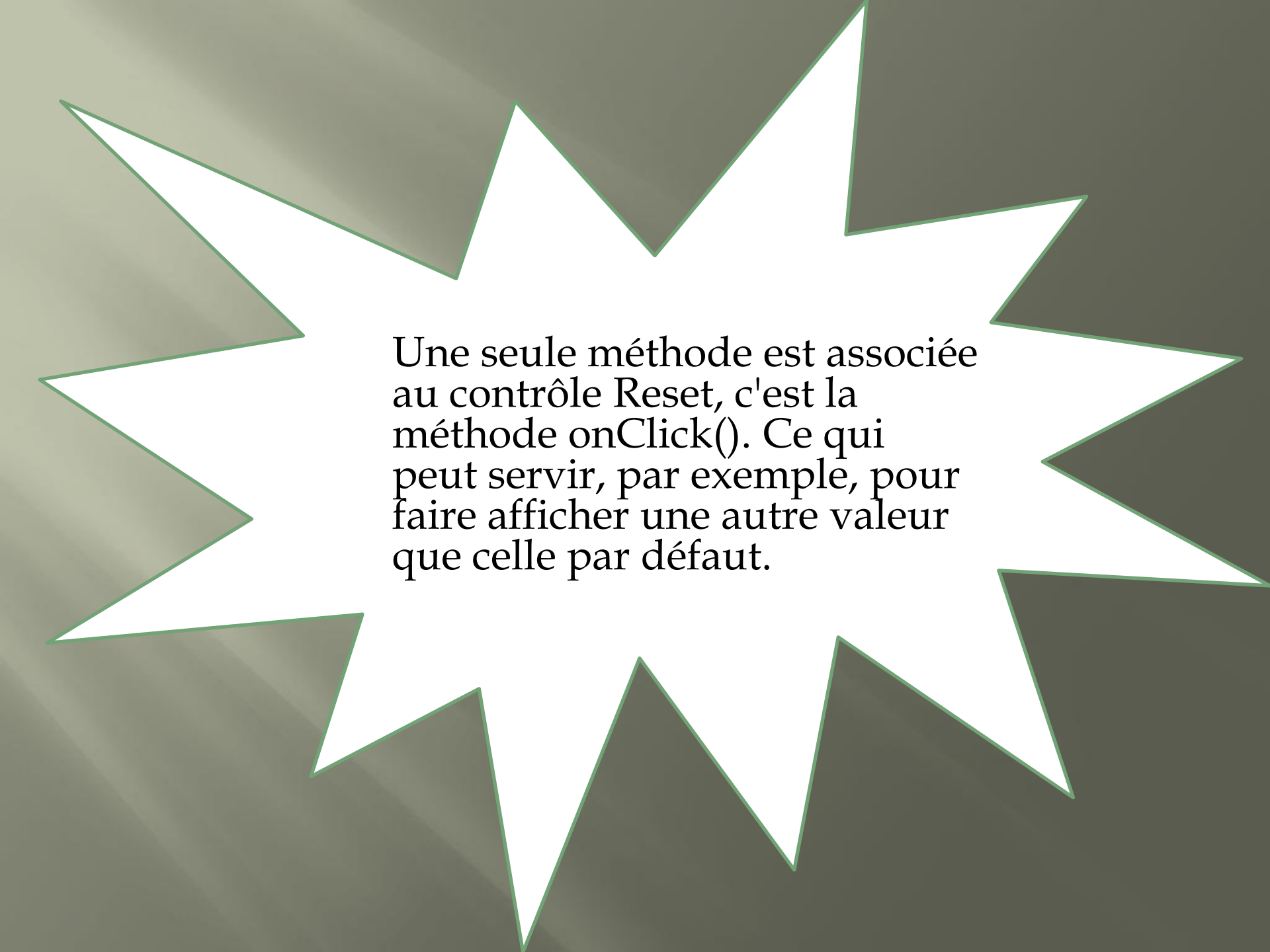
# Le contrôle Reset

Le contrôle Reset permet d'annuler les modifications apportées aux contrôles d'un formulaire et de restaurer les valeurs par défaut.

la syntaxe Html est :

```
<INPUT TYPE="reset" NAME="nom"  
  VALUE="texte">
```

où VALUE donne le texte du bouton.



Une seule méthode est associée  
au contrôle Reset, c'est la  
méthode `onClick()`. Ce qui  
peut servir, par exemple, pour  
faire afficher une autre valeur  
que celle par défaut.

# Le contrôle Submit

Le contrôle a la tâche spécifique de transmettre toutes les informations contenues dans le formulaire à l'URL désignée dans l'attribut ACTION du tag <FORM>.

la syntaxe Html est :

```
<INPUT TYPE="submit" NAME="nom"  
      VALUE="texte">
```

où VALUE donne le texte du bouton.

Une seule méthode est associée au contrôle Submit, c'est la méthode onClick().