

Le JavaScript

Javascript

Le Javascript est un langage de script incorporé dans un document HTML. Historiquement il s'agit même du premier langage de script pour le Web. Ce langage est un langage de programmation qui permet d'apporter des améliorations au langage HTML en permettant d'exécuter des commandes du côté client, c'est-à-dire au niveau du navigateur et non du serveur web.

Ainsi le langage Javascript est fortement dépendant du navigateur appelant la page web dans laquelle le script est incorporée, mais en contrepartie il ne nécessite pas de compilateur, contrairement au langage Java, avec lequel il a longtemps été confondu.

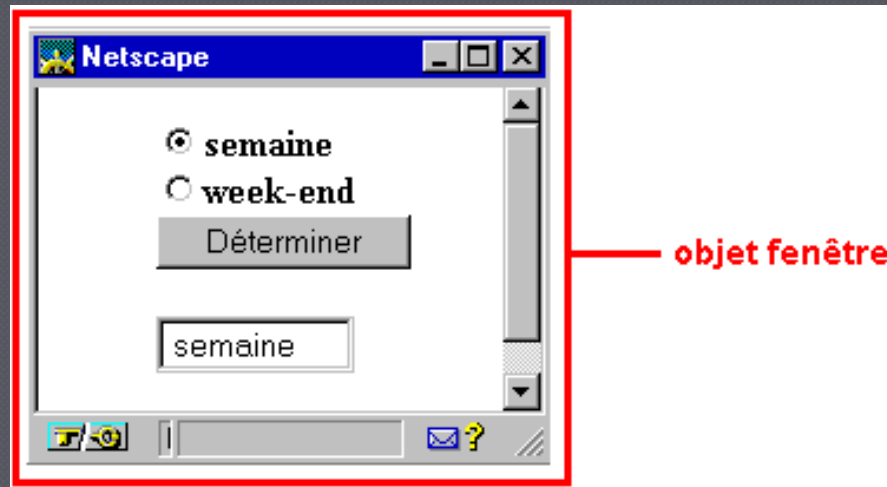
Il importe de savoir que Javascript est totalement différent de Java. Bien que les deux soient utilisés pour créer des pages Web évoluées, bien que les deux reprennent le terme Java (café en américain), nous avons là deux outils informatiques bien différents.

Un peu de théorie objet

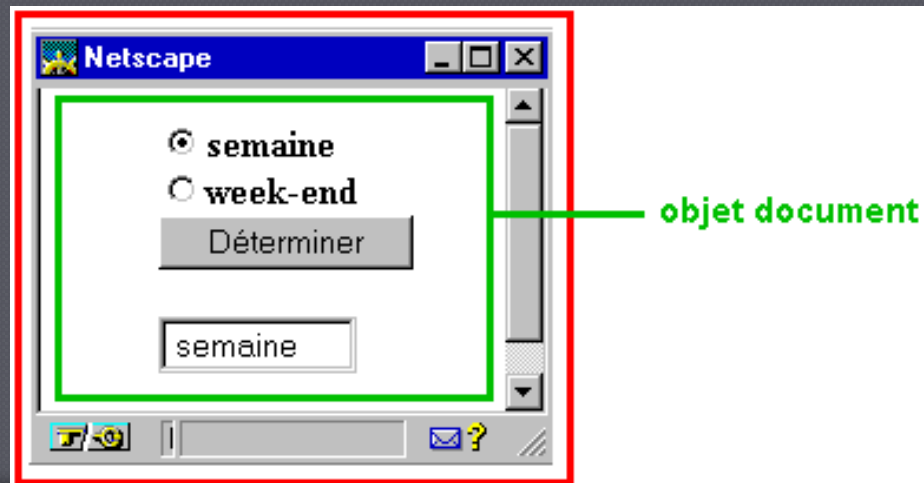
- Javascript va diviser cette page en objets et surtout va vous permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler.
- Voyons d'abord une illustration des différents objets qu'une page peut contenir.
- Vous avez chargé la page suivante :



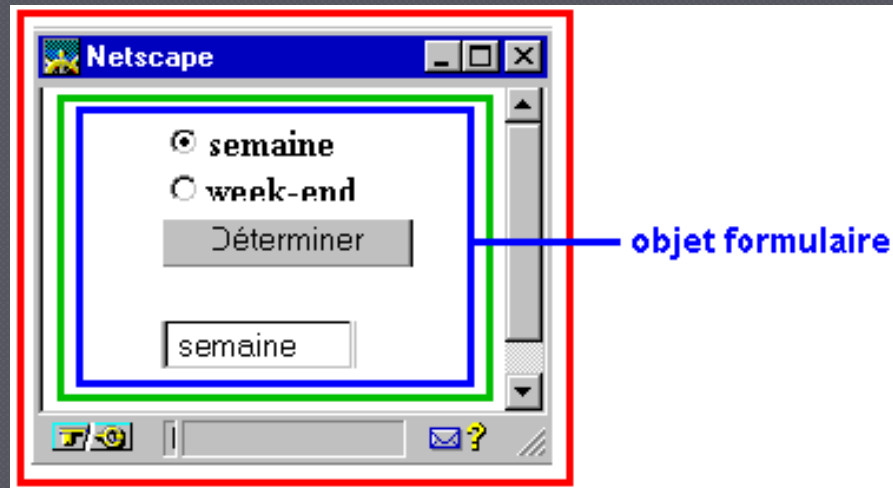
- Cette page s'affiche dans une fenêtre. C'est l'objet fenêtre.



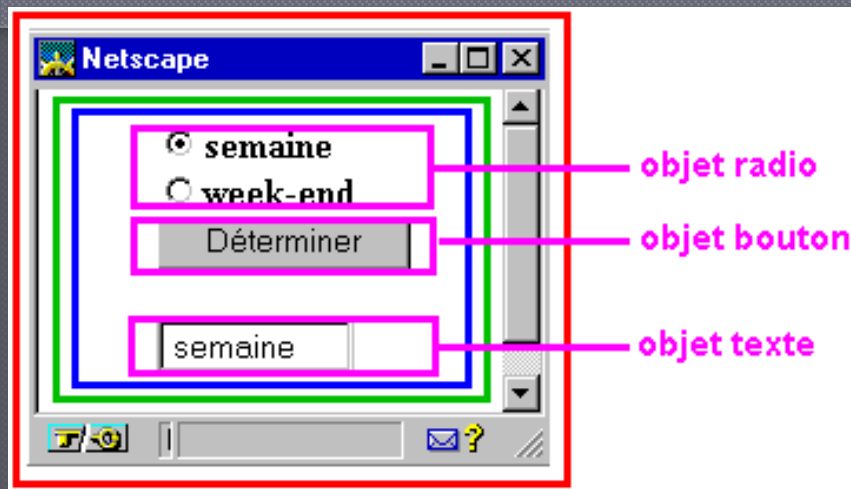
Dans cette fenêtre, il y a un document Html. C'est l'objet document. Autrement dit (et c'est là que l'on voit apparaître la notion de la hiérarchie des objets Javascript), l'objet fenêtre contient l'objet document.



- Dans ce document, on trouve un formulaire au sens Html. C'est l'objet formulaire. Autrement dit, l'objet fenêtre
- contient un objet document qui lui contient un objet formulaire.



Dans ce document, on trouve trois objets. Des boutons radio, un bouton classique et une zone de texte. Ce sont respectivement l'objet radio, l'objet bouton, l'objet texte. Autrement dit l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet radio, l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet bouton et l'objet fenêtre contient l'objet document qui contient l'objet formulaire qui contient à son tour l'objet texte.



- La hiérarchie des objets de cet exemple est donc

Fenêtre document formulaire radio
 bouton
 texte

- Pour accéder à un objet (vous l'avez peut-être déjà deviné), il faudra donner le chemin complet de l'objet en allant du contenant le plus extérieur à l'objet à l'objet référencé.
- Soit par exemple pour le bouton radio "semaine" : `(window).document.form.radio[0]`.
- Nous avons mis l'objet window entre parenthèses car comme il occupe la première place dans la hiérarchie, il est repris par défaut par Javascript et devient donc facultatif.
- Et enfin pour les puristes, Javascript n'est pas à proprement parler un langage orienté objet tel que C++ ou Java.
- On dira plutôt que Javascript est un langage basé sur les objets.

Commentaires

Parfois, un programmeur souhaite annoter le code qu'il vient d'écrire, mais sans que ces annotations n'interfèrent avec les instructions proprement dites. On appelle ces annotations des commentaires du programmes. Pour que l'interprète ne confonde pas ces caractères avec les instructions, (pour qu'il les ignore), on fait précéder un commentaire disposé en n de ligne par les deux barres obliques (//) et on utilise /* et */ pour délimiter un commentaire sur plusieurs lignes. Voici un exemple de code commenté.

Exemple

```
/* test pour poser la bonne question */  
if (reduc && vieux) { // a une reduction et est vieille  
document.write("Voulez que quelqu'un vous accompagne ?");  
}
```

Types de données

- typé. Ceci ne veut pas dire qu'il est capable d'effectuer l'instruction `1+"bonjour"`,
- mais qu'il n'impose pas au programmeur d'indiquer explicitement le type d'une
- variable lors de la déclaration de celle-ci.

Exemples de déclaration

- `var texte;`
- `var cle;`
- `var compteur = 3;`
- `var Erreur = "Connexion perdue.";`
- `var Erreur2 = 'Pas assez de mémoire';`

Opérations, expressions

et Java existent en JavaScript. Nous ne donnerons que quelques exemples ici, et nous découvrirons ces expressions au 1 de ce document. Voici quelques exemples, les instructions sur la même ligne ayant la même signification.

Opérations	Expressions
<code>total += 4;</code>	<code>total = total + 4;</code>
<code>i++;</code>	<code>i = i + 1;</code>

Déroulement d'un programme

• La boucle for

- La boucle est introduite par le mot-clé for.
- Pour afficher nos cinquante premiers entiers, nous pouvons écrire une boucle qui nous permet de réduire le texte de programme à trois lignes (au lieu de cent).
- Cette boucle utilise la variable `i` qui vaut 1 initialement. Ensuite le corps de la boucle, c'est-à-dire les instructions à l'intérieur des crochets (`{` et `}`) est exécuté tant que la condition `i<=50` est vraie. A chaque tour de boucle, on incrémente la variable `i`.

```
for (i=1; i<=50; i++) {  
document.writeln(i);  
}
```

● La boucle while

- Une autre structure itérative plus simple est la boucle while. Celle-ci dit simplement
- qu'on doit répéter un ensemble d'instructions tant qu'une condition est vraie.
- while (condition){
- instructions }

Exemple :

```
var recommence = true;
    while (recommence) {
        m = calc_moyenne();
        recommence=prompt("La moyenne est " + m + ".Recommencer ?");
    }
```

- Notons qu'il est important d'initialiser la variable recommence à la valeur vraie an
- d'entrer dans la boucle la première fois.

● L'alternative

- condition est vraie. Le type de la condition est booléen (la condition est soit vraie
- soit fausse). La structure est la suivante :
- `if (condition1){`
- `instructions1 }`
- `else {`
- `instructions2 }`
- Exemple : on peut simplement tester la valeur d'une variable, comme dans l'exemple
- suivant.
- `if (i==50) {`
- `document.writeln("i est égal à 50")`
- `}`
- `else {`
- `document.writeln("i n'est pas égal à 50")`
- `}`

Fonctions

De même que la boucle permet de répéter plusieurs instructions, la fonction est très utile pour accomplir une tâche répétitive. Le principe consiste à regrouper dans un bloc un ensemble d'instructions nécessaire pour accomplir une tâche, à nommer ce bloc, et à l'appeler à chaque fois qu'on a besoin d'effectuer la tâche.

La définition d'une fonction est introduite par le mot clé `function` suivi de ses arguments, puis du code de la fonction entre accolades (`{ }`). Le résultat de la fonction est indiqué par le mot-clé `return`. Par exemple, ci-dessous est définie la fonction `carre` qui rend le carré de son argument.

```
<script language="javascript">  
function carre(i) {  
    return (i*i)  
}  
</script>
```

Notons que rien ne se passe si on charge seulement la page contenant ce script. Il faut en effet appeler la fonction avec un argument effectif pour que l'évaluation se fasse. Pour cela, on pourra par exemple ajouter après le code de la fonction, un appel à cette fonction :

```
<script language="javascript">  
document.write("La fonction a retourné ",carre(4),".")  
</script>
```

Pour appeler la fonction, il faut bien sûr qu'elle ait été définie auparavant. Par conséquent, il est conseillé de placer la définition de fonction dans l'entête de la page HTML, et les appels dans le corps de cette page. Pour notre exemple, la page HTML a donc la structure suivante :

Exemple 1 :

```
<HEAD>
<script language="javascript">
function carre(i) {
document.write("on a appelé carre avec l'argument ",i,"<br>")
return i * i
}
</script>
</HEAD>
<BODY>
<script language="javascript">
document.write("La fonction a retourné ",carre(4),".")
</script>
</BODY>
```

Exemple 2 :

```
<SCRIPT language="Javascript">
```

```
<!--var a = 12;var b = 4;function MultipliePar2(b)
```

```
{  
  var a = b * 2;  
  return a;}  
document.write("Le double de ",b," est ",MultipliePar2(b));  
document.write("La valeur de a est ",a);// -->  
</SCRIPT>
```

Dans l'exemple ci-dessus, la variable a est déclarée explicitement en début de script, ainsi que dans la fonction. Voici ce qu'affiche ce script :

Le double de 4 est 8

La valeur de a est 12

Exemple 3 :

Voici un autre exemple dans lequel a est déclarée implicitement dans la fonction :

```
<SCRIPT language="Javascript">
```

```
<!--var a = 12;var b = 4;function MultipliePar2(b)
```

```
{  
  a = b * 2;  
  return a;}  
document.write("Le double de ",b," est ",MultipliePar2(b));  
document.write("La valeur de a est ",a);// --> </SCRIPT>
```

L'instruction switch...case

L'instruction switch permet de faire plusieurs tests de valeurs sur le contenu d'une même variable. Ce branchement conditionnel simplifie beaucoup le test de plusieurs valeurs d'une variable, car cette opération aurait été compliquée (mais possible) avec des if imbriqués. Sa syntaxe est la suivante :

```
switch (Variable) {  
  case Valeur1:  
    Liste d'instructions;  
    break;  
  case Valeur2:  
    Liste d'instructions;  
    break;  
  case ValeurX:  
    Liste d'instructions;  
    break;  
  default:  
    Liste d'instructions;  
    break;}
```


La méthode write

- La méthode write() de l'objet document permet de modifier de façon dynamique le contenu d'une page HTML. Voici la syntaxe de la méthode write() :
- `window.document.write(expression1, expression2, ...)`
- Cette méthode permet d'écrire le résultat des expressions passées en paramètre dans le document dans lequel elle est utilisée
- Il est notamment possible d'utiliser des balises HTML à l'intérieur même de la méthode write :
- `document.write('Bonjour');`

La méthode writeln

- La méthode `writeln()` fonctionne exactement comme la méthode `write()` à la seule différence qu'elle ajoute un retour chariot à la fin de la chaîne.
Or un retour chariot (en HTML) est ignoré par le navigateur (Rappel: un retour à la ligne se fait avec la balise `
`). Cette méthode n'a donc un avantage que lorsqu'elle est utilisée dans des éléments HTML sensibles aux retours à la ligne, par exemple entre les balises `<PRE>` et `</PRE>` qui formatent le texte comme dans un fichier texte (et qui prend donc en compte les retours à la ligne).

La méthode alert()

La méthode `alert()` permet d'afficher dans une boîte toute simple composée d'une fenêtre et d'un bouton OK le texte qu'on lui fournit en paramètre. Dès que cette boîte est affichée, l'utilisateur n'a d'autre alternative que de cliquer sur le bouton OK.

Son unique paramètre est une chaîne de caractère, on peut donc lui fournir directement cette chaîne de caractères entre guillemets, lui fournir une variable dont il affichera le contenu, ou bien mêler les deux en concaténant les chaînes grâce à l'opérateur `+`.

```
alert(nom_de_la_variable);alert('Ch  
aîne de caractères');alert('Chaîne de  
caractères' + nom_de_la_variable);
```

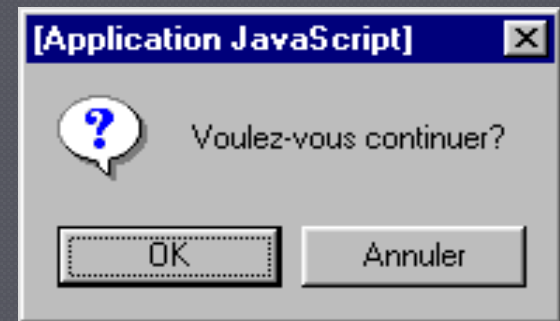


La méthode confirm()

- La méthode confirm() est similaire à la méthode alert(), si ce n'est qu'elle permet un choix entre "OK" et "Annuler". Lorsque l'utilisateur appuie sur "OK" la méthode renvoie la valeur true. Elle renvoie false dans le cas contraire...

Elle admet comme alert() un seul paramètre: une chaîne de caractères...
Sa syntaxe est :

```
confirm('Chaîne de caractères');
```



La méthode prompt()

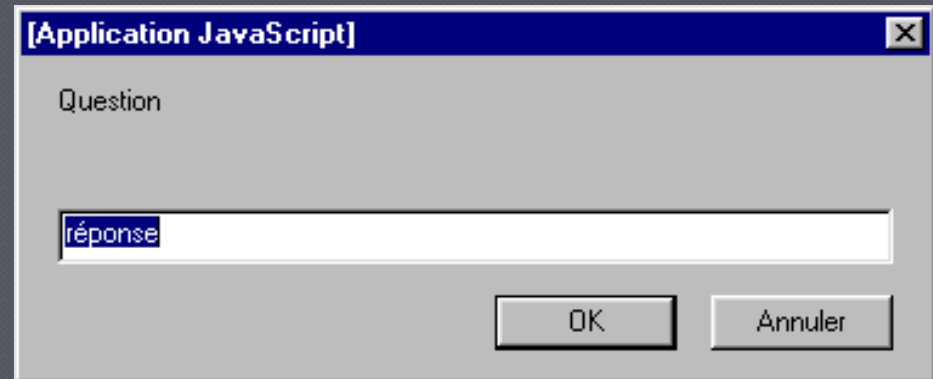
La méthode prompt est un peu plus évoluée que les deux précédentes puisqu'elle fournit un moyen simple de récupérer une information provenant de l'utilisateur, on parle alors de boîte d'invite. La méthode prompt() requiert deux arguments :

- le texte d'invite

- la chaîne de caractères par défaut dans le champ de saisie

Sa syntaxe est donc la suivante :

`prompt('Posez ici votre question','chaîne par défaut');` Cette boîte d'invite retourne la valeur de la chaîne saisie par l'utilisateur, elle retourne la valeur null si jamais aucun texte n'est saisi...



Les formulaires

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= ...>`.

Pour déclarer une formulaire en utilise la balise :
`< Form method = "post" action = "URL" name = "Nom_Form">`
....
`</Form>`

Le contrôle ligne de texte

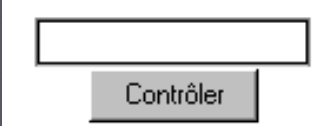
La zone de texte est l'élément d'entrée/sortie par excellence de Javascript. La syntaxe Html est

```
<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.

Exemple 1 :

```
<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>
```



Exemple 2 :

```
<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>
```

Zone de texte d'entrée

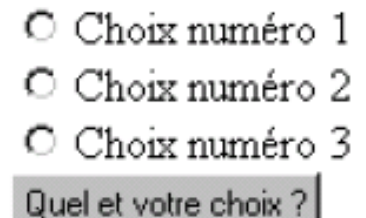
Zone de texte de sortie

Les boutons radio

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

- 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) };
}
28
</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 et votre choix ?" on
</FORM>
</BODY>
</HTML>
```



☐ Choix numéro 1

☐ Choix numéro 2

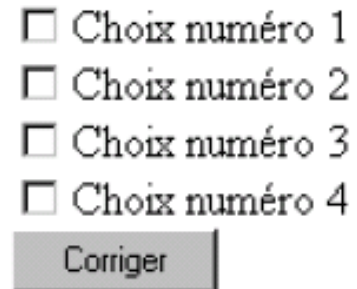
☐ Choix numéro 3

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.

Exemple :

```
<HTML>
<HEAD>
<script language="javascript">
function reponse(form4) {
29
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<
<INPUT TYPE="CHECKBOX" NAME="check2" VALUE="2">Choix numéro 2<
<INPUT TYPE="CHECKBOX" NAME="check3" VALUE="3">Choix numéro 3<
<INPUT TYPE="CHECKBOX" NAME="check4" VALUE="4">Choix numéro 4<
<INPUT TYPE="button"NAME="but" VALUE="Corriger" onClick="reponse(form4)">
</FORM>
</BODY>
</HTML>
```



A screenshot of a web form. It contains four checkboxes, each followed by a label: "Choix numéro 1", "Choix numéro 2", "Choix numéro 3", and "Choix numéro 4". Below the checkboxes is a button labeled "Corriger".

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.

Exemple :

```
<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">
```

```
30
```

```
<OPTION VALUE="1">Elément 1
```

```
<OPTION VALUE="2">Elément 2
```

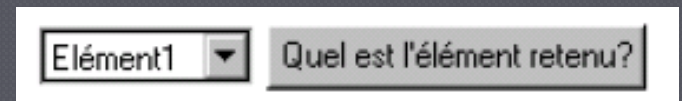
```
<OPTION VALUE="3">Elément 3
```

```
</SELECT>
```

```
<INPUT TYPE="button" NAME="b" VALUE="Quel est l'élément retenu?"  
onClick="liste(form5)"> </FORM>
```

```
</BODY>
```

```
</HTML>
```



The screenshot shows a web form with a dropdown menu and a button. The dropdown menu is labeled 'Elément1' and has a downward arrow. The button is labeled 'Quel est l'élément retenu?'.

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.

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().

Le contrôle Hidden (caché)

- Le contrôle Hidden permet d'entrer dans le script des éléments (généralement des données) qui n'apparaîtront pas à l'écran. Ces éléments sont donc cachés. D'où son nom.
- la syntaxe Html est :
- `<INPUT TYPE="hidden" NAME="nom" VALUE "les données cachées">`

Exemple

```
<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>
<FORM>
<TEXTAREA NAME="nom" ROWS=x COLS=y>
texte par défaut
</TEXTAREA>
<INPUT TYPE="reset" NAME="nom" VALUE "texte">
<INPUT TYPE="submit" NAME="nom" VALUE "texte">
<INPUT TYPE="hidden" NAME="nom" VALUE "les données cachées">
</FORM><FORM METHOD="post" ACTION="mailto:votre_adresse_Email">
<INPUT TYPE=text NAME="nom">
<TEXTAREA NAME="adresse" ROWS=2 COLS=35>
</TEXTAREA>
<INPUT TYPE=submit VALUE="Submit">
</FORM>
</BODY>
</HTML>
```


Entrez votre choix :

Élément 1 ▼

Quel est l'élément retenu?

texte par défaut



Submit