



Chapitre 3: Langage JavaScript

**Département Informatique, Ecole Supérieur de Technologie -Guelmim-
Université Ibn Zohr**

Année Universitaire: 2022/2023



Plan de cours

III. Langage JavaScript

- 1) Introduction.**
- 2) Le JavaScript minimum**
- 3) Les variables.**
- 4) Les opérateurs.**
- 5) Les conditions.**
- 6) Les boucles.**
- 7) Les fonctions.**
- 8) Les événements.**
- 9) Les formulaires.**

Introduction: JavaScript

- JavaScript est un langage de scripts qui incorporé aux balises Html, permet d'améliorer la présentation et l'interactivité des pages Web.
- JavaScript est donc une extension du code HTML des pages Web sous forme des scripts.
- Ces scripts vont être gérés et exécutés par le browser lui-même sans devoir faire appel aux ressources du serveur.
- Ces instructions seront donc traitées en direct et surtout sans retard par le navigateur.

Introduction: JavaScript n'est pas Java.

- Il importe de savoir que JavaScript est totalement différent de Java. Bien que les deux soient utilisés pour créer des pages Web évoluées, nous avons là deux outils informatiques bien différents.
- **JavaScript:**
 - Code intégré dans la page Html;
 - Code interprété par le browser au moment de l'exécution;
 - Codes de programmation simples mais pour des applications limitées;
 - Permet d'accéder aux objets du navigateur;
 - Confidentialité des codes nulle (code source visible).

Introduction: JavaScript n'est pas Java.

- Il importe de savoir que JavaScript est totalement différent de Java. Bien que les deux soient utilisés pour créer des pages Web évoluées, nous avons là deux outils informatiques bien différents.
- **Java:**
 - Module (applet) distinct de la page Html;
 - Code source compilé avant son exécution;
 - Langage de programmation beaucoup plus complexe mais plus performant;
 - N'accède pas aux objets du navigateur;
 - Sécurité (code source compilé).

Introduction: JavaScript n'est pas Java.

■ Plus simplement:

- JavaScript est plus simple à mettre en œuvre car c'est du code que vous ajouterez à votre page écrite en Html.
- Java pour sa part, nécessite une compilation préalable de votre code.
- Le champ d'application de JavaScript est somme toute assez limité alors qu'en Java vous pourrez en principe tout faire.
- Le code JavaScript est inclus dans votre page Html, celui-ci est visible et peut être copié par tout le monde. Par contre, en Java, votre code source est broyé par le compilateur et est ainsi indéchiffrable.
- Les codes JavaScripts ne ralentissent pas le chargement de la page alors que l'appel à une applet Java peut demander quelques minutes de patience supplémentaire à votre lecteur.

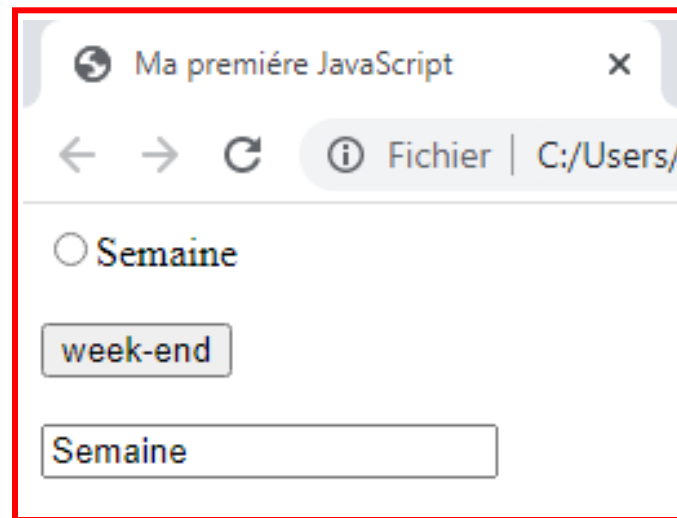
Introduction: Vos outils pour le JavaScript

- Pour apprendre et exploiter le JavaScript, il vous faut :
 - Un browser qui reconnaît le JavaScript.
 - Une solide connaissance du HTML: Comme le code du JavaScript vient s'ajouter au "code" du langage Html, une connaissance approfondie des balises ou tags Html est souhaitable sinon indispensable.
 - Un simple éditeur de texte: Une page HTML n'est que du texte. Le code JavaScript n'est lui aussi que du texte. Quoi de plus simple qu'un éditeur de texte comme le Notepad, Sublime Text...pour inclure votre JavaScript dans votre page Html sera largement suffisant.

Introduction: La théorie d'objet en JavaScript

■ Les objets et leur hiérarchie:

- JavaScript va diviser une page web en objets et surtout va vous permettre d'accéder à ces objets, d'en retirer des informations et de les manipuler.
- Vous avez chargé la page suivante :
 - ✓ Cette page s'affiche dans une fenêtre. C'est l'objet **fenêtre**.



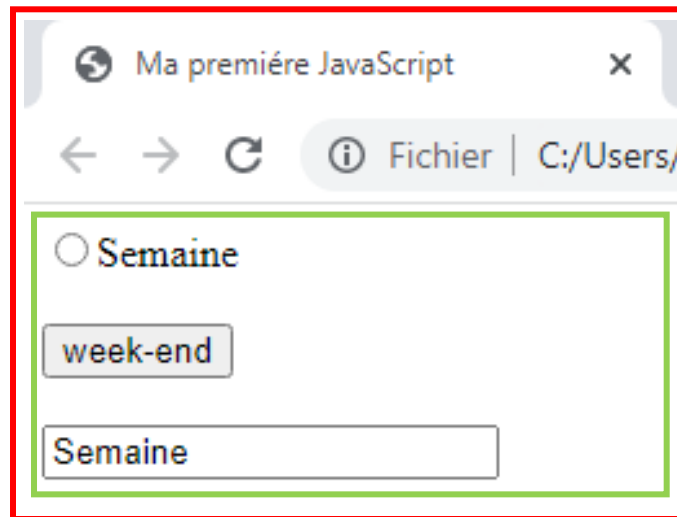
Objet fenêtre

Introduction: La théorie d'objet en JavaScript

■ Les objets et leur hiérarchie:

➤ Vous avez chargé la page suivante :

- ✓ Dans cette fenêtre, il y a un document Html. C'est l'objet **document**.
- ✓ la notion de la hiérarchie des objets JavaScript, l'objet **fenêtre** contient l'objet **document**.



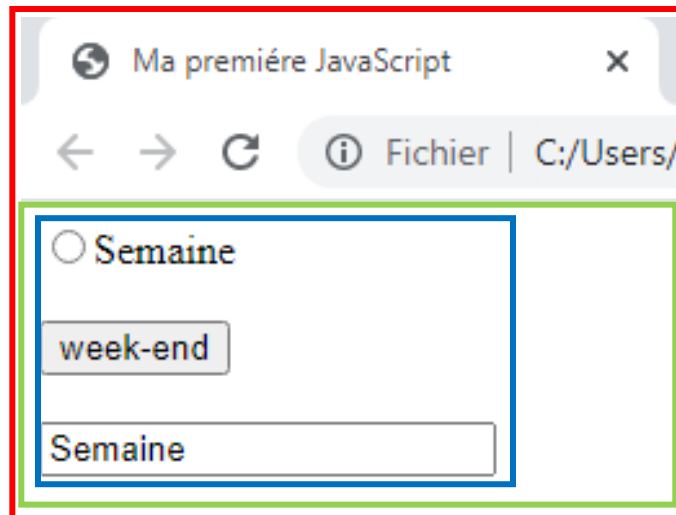
← Objet document

Introduction: La théorie d'objet en JavaScript

■ Les objets et leur hiérarchie:

➤ Vous avez chargé la page suivante :

- ✓ 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**.



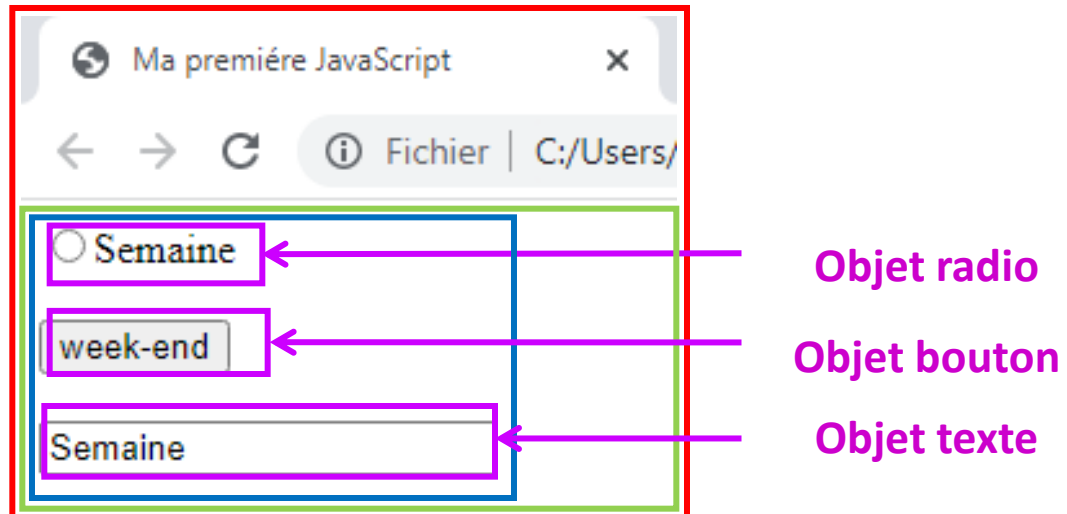
← Objet formulaire

Introduction: La théorie d'objet en JavaScript

■ Les objets et leur hiérarchie:

➤ Vous avez chargé la page suivante :

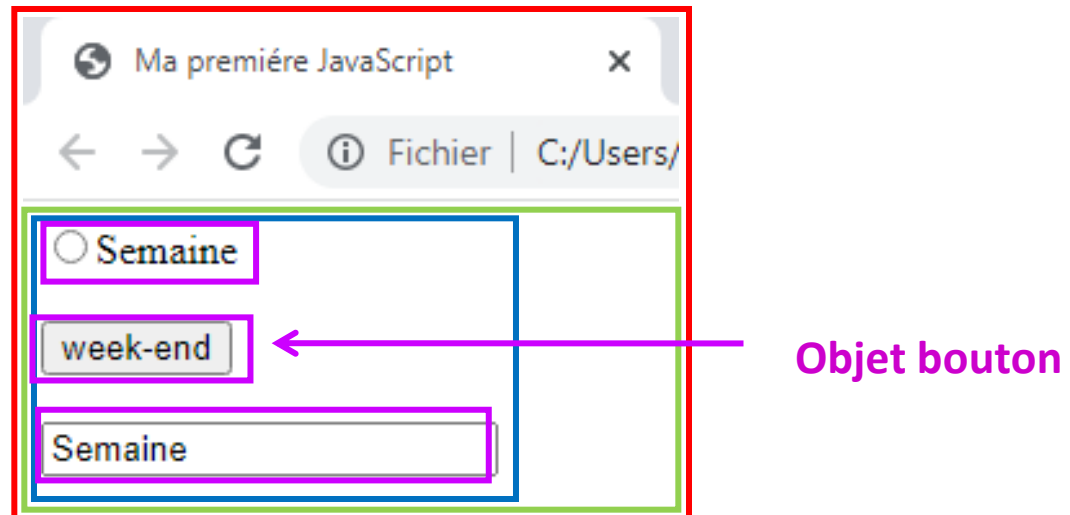
- ✓ Dans ce document, on trouve un formulaire au sens HTML. C'est l'objet **formulaire**.
- ✓ Autrement dit, l'objet **fenêtre** contient un objet **document** qui lui contient un objet **formulaire**.



Introduction: La théorie d'objet en JavaScript

■ Les objets et leur hiérarchie:

- Pour accéder à un objet, 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]`.



Introduction: La théorie d'objet en JavaScript

■ Les propriétés des objets:

- Une propriété est un attribut, une caractéristique, une description de l'objet.
- Les objets JavaScript ont des propriétés personnalisées. Dans le cas des boutons radio, une de ses propriétés est, par exemple, sa sélection ou sa non-sélection (**checked** en anglais).
- En JavaScript, pour accéder aux propriétés, on utilise la syntaxe :
 - ✓ `nom_de_l'objet.nom_de_la_propriété;`
- Dans le cas du bouton radio "semaine", pour tester la propriété de sélection, on écrira:
 - ✓ `document.form.radio[0].checked.`

Le JavaScript minimum:

■ La balise <SCRIPT>:

- Dans la logique du langage HTML, il faut signaler au browser par une balise, que c'est du JavaScript. C'est la balise:
 - ✓ `<SCRIPT LANGUAGE="Javascript">`.
- De même, il faudra informer le browser de la fin du script. C'est la balise `</SCRIPT>`.

■ Il y a deux manières d'ajouter du code JavaScript dans une page :

- en ajoutant le code JavaScript à l'intérieur de l'élément script :
`<SCRIPT LANGUAGE="Javascript"> // Mon code Javascript ... </SCRIPT>`
- En liant depuis la page HTML un fichier externe, dans lequel sont placées les instructions JavaScript.
`<SCRIPT type="text/javascript" src='file.js'></SCRIPT>`

Le JavaScript minimum:

■ Les commentaires:

- Il vous sera peut-être utile d'inclure des commentaires personnels dans vos codes JavaScript.
- JavaScript utilise les conventions utilisées en C et C++ soit **// commentaire** Tout ce qui est écrit entre le **//** et la fin de la ligne sera ignoré.
- Il sera aussi possible d'inclure des commentaires sur plusieurs lignes avec le code **/* commentaire sur plusieurs lignes */**
- Ne confondez pas les commentaires JavaScript et les commentaires HTML (pour rappel **<!-- ...-->**).

Le JavaScript minimum:

■ Où inclure le code en JavaScript ?

- Le principe est simple. Il suffit de respecter les deux principes suivants :
 - ✓ n'importe où.
 - ✓ mais là où il le faut.
- Il faut avoir l'habitude de déclarer systématiquement un maximum d'éléments dans la balises d'en-tête soit entre la balise **<HEAD>** et **</HEAD>** et avant la balise **<BODY>** afin d'assurer que le programme script est chargé dans la page et prêt à fonctionner à toute intervention de votre visiteur.
- Il faut noter que l'usage de la balise script n'est pas toujours obligatoire. C'est le cas des événements JavaScript (par exemple **onClick**) où il faut simplement insérer le code à l'intérieur de la commande HTML comme un attribut. L'événement fera appel à la fonction JavaScript lorsque la commande Html sera activée.

Le JavaScript minimum:

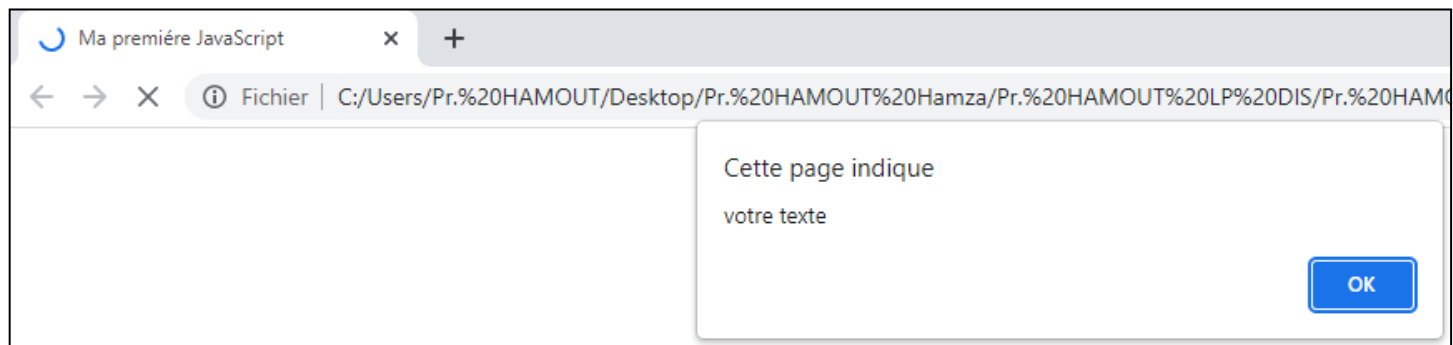
■ Une première instruction JavaScript: alert

- Voyons une première instruction JavaScript: en fait une méthode de l'objet **window**, soit:
 - ✓ **alert("votre texte");** : Cette instruction affiche un message présent votre texte entre les guillemets dans une boîte de dialogue pourvue d'un bouton OK. Pour continuer dans la page, le lecteur devra cliquer ce bouton.
 - ✓ Les guillemets " et l'apostrophe ' font partie intégrante du langage JavaScript. On peut utiliser l'une ou l'autre forme à condition de ne pas les mélanger. Ainsi **alert('...')** donnera un message d'erreur.
 - ✓ Si vous souhaitez utiliser des guillemets dans vos chaînes de caractères, tapez \" ou \' pour les différencier vis à vis du compilateur.

Le JavaScript minimum:

■ Une première instruction JavaScript: alert

```
premier_pas_javascript_alert.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
  </head>
  <BODY>
    Bla-bla en HTML
    <SCRIPT LANGUAGE="Javascript">
      alert("votre texte");
    </SCRIPT>
  </BODY>
</html>
```



Le JavaScript minimum:

■ Méthode de l'objet document:

➤ La méthode `write()` & `writeln()`:

- ✓ La syntaxe est assez simple soit **`write("votre texte");`** et **`writeln("votre texte");`**
- ✓ On peut aussi écrire une variable, soit la variable resultat: **`write(resultat);`** et **`writeln(resultat);`**
- ✓ Pour associer du texte (chaînes de caractères) et des variables, on utilise l'instruction **`write("Le résultat est " + resultat);`**
- ✓ On peut utiliser les balises Html pour agrémenter ce texte:
 - ❖ `write("Le résultat est" + resultat);` ou
 - ❖ `write ("" + "Le résultat est " + "" + resultat);`

Le JavaScript minimum:

■ Méthode de l'objet document:

➤ La méthode `write()` & `writeln()`:

- ✓ La méthode **`writeln()`** ajoute un retour chariot à la fin des caractères affichés par l'instruction. Ce qui n'a aucun effet en HTML. Pour faire fonctionner **`writeln()`** Il faut l'inclure dans des balises **`<PRE>`**.
- ✓ Autre astuce est d'utiliser simplement le tag **`
`** avec la méthode **`write()`**.

Le JavaScript minimum:

■ Méthode de l'objet document:

```
premier_pas_javascript_write_writeln.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
  </head>
  <BODY>
    <pre>
      <SCRIPT LANGUAGE="Javascript">
        document.writeln("<h1> write:</h1>"+
          Du texte avec writeln en
          Javascript"+2022);
        document.write("Du texte avec write
          en Javascript"+"<br>");
        document.write("La finde teste"+"<br>");
      </SCRIPT>
    </pre>
  </BODY>
</html>
```



Le JavaScript minimum:

■ Méthode de l'objet document:

- **variable.big():** L'emploi de **.big()** affichera la variable comme si elle était comprise entre les balises HTML **<BIG></BIG>**.
- **variable.small():** L'emploi de **.small()** affichera la variable comme si elle était comprise entre les balises HTML **<SMALL></SMALL>**.
- **variable.blink():** L'emploi de **.blink()** affichera la variable comme si elle était comprise entre les balises HTML **<BLINK></BLINK>**.
- **variable.bold():** L'emploi de **.bold()** affichera la variable comme si elle était comprise entre les balises HTML ****.
- **variable.fixed():** L'emploi de **.fixed()** affichera la variable comme si elle était comprise entre les balises HTML **<TT></TT>**.

Le JavaScript minimum:

■ Méthode de l'objet document:

- **variable.italics():** L'emploi de **.italics()** affichera la variable comme si elle était comprise entre les balises HTML **<I></I>**.
- **variable.fontcolor(color):** L'emploi de **.fontcolor(color)** affichera la variable comme si elle était comprise entre les balises HTML **<FONTCOLOR="color">**.
- **variable.fontSize(x):** L'emploi de **.fontSize(x)** affichera la variable comme si elle était comprise entre les balises HTML **<FONTSIZE="x">**.
- **variable.strike():** L'emploi de **.strike()** affichera la variable comme si elle était comprise entre les balises HTML **<STRIKE></STRIKE>**.
- **variable.sub():** L'emploi de **.sub()** affichera la variable comme si elle était comprise entre les balises HTML ****.

Le JavaScript minimum:

■ Méthode de l'objet document:

- **variable.sup():** L'emploi de **.sup()** affichera la variable comme si elle était comprise entre les balises Html . **<SUP></SUB>**.
- **document.bgColor:** Cette instruction permet de spécifier la couleur d'arrière-plan d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.
- **document.fgColor:** Cette instruction permet de spécifier la couleur d'avant-plan (texte) d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

Le JavaScript minimum:

■ Méthode de l'objet document:

- **document.alinkColor** Cette instruction permet de spécifier la couleur d'un lien actif d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.
- **document.linkColor**: Cette instruction permet de spécifier la couleur d'un hyperlien d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.
- **document.vlinkColor**: Cette instruction permet de spécifier la couleur d'un hyperlien déjà visité d'un objet document. On peut employer le nom ou la valeur RGB de la couleur.

Le JavaScript minimum:

premier_pas_javascript_document_balises_html.html x

```
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
  </head>
  <BODY>
    <SCRIPT LANGUAGE="Javascript">
      str="ESTG Année 2022-2023";
      document.write(str.big()+"<br>");
      document.write(str.small()+"<br>");
      document.write(str.blink()+"<br>");
      document.write(str.bold()+"<br>");
      document.write(str.fixed()+"<br>");
      document.write(str.italics()+"<br>");
      document.write(str.fontcolor()+"<br>");
      document.write(str.fontSize()+"<br>");
      document.write(str.strike()+"<br>");
      document.write(str.sub()+"<br>");
      document.write(str.sup()+"<br>");
      document.bgColor="green";
      document.fgColor="pink";
      document.alinkColor="gray";
      document.linkColor="yellow";
      document.vlinkColor="red";
    </SCRIPT>
    <a href="www.estg.ui.ac.ma"> Lien: www.estg.uiz.ac.ma </a>
  </BODY>
</html>
```

Ma première JavaScript x

← → ↻ ⓘ Fichier | C:/Us...

ESTG Année 2022-2023

ESTG Année 2022-2023

ESTG Année 2022-2023

ESTG Année 2022-2023

ESTG Année 2022-2023

ESTG Année 2022-2023

ESTG Année 2022-2023

ESTG Année 2022-2023

~~ESTG Année 2022-2023~~

ESTG Année 2022-2023

ESTG Année 2022-2023

Lien: www.estg.uiz.ac.ma

Les variables

■ Les déclaration des variables en JavaScript:

- Les variables contiennent des données qui peuvent être modifiées lors de l'exécution d'un programme. On y fait référence par le nom de cette variable.
- Les variables peuvent se déclarer de deux façons:
 - ✓ Soit de façon explicite. On dit à JavaScript que ceci est une variable on utilisant la commande var. Par exemple :
 - `var Numero = 1`
 - `var Prenom = "Luc"`
 - ✓ Soit de façon implicite. On écrit directement le nom de la variable suivi de la valeur que l'on lui attribue et JavaScript s'en accommode.
 - `Numero = 1`
 - `Prenom = "Luc"`

Les variables

■ Les types variables sous JavaScript:

- JavaScript utilise quatre types de données :
 - ✓ Des nombres Tout nombre entier ou avec virgule tel que 22 ou 3.1416.
 - ✓ Des chaînes de caractères Toute suite de caractères comprise entre guillemets telle que "suite de caractères".
 - ✓ Des booléens Les mots **true** pour vrai et **false** pour faux.
 - ✓ Le mot **null** Mot spécial qui représente pas de valeur.
- Notons aussi que contrairement au langage C/C++, Il ne faut pas déclarer le type de données d'une variable. On n'a donc pas besoin de int, float, double, char et autres long en JavaScript.

Les variables

■ **Les noms des variables réservés:** Les mots de la liste ci-après ne peuvent être utilisés pour des noms de fonctions et de variables:

- A abstract,
- B boolean, break, byte;
- C case, catch, char, class, const, continue;
- D default, do, double;
- E else, extends;
- F false, final, finally, float, for, function;
- G goto;
- I if, implement,s import, in, instanceof, int, interface;
- L long, N, nativ,e new, null;
- P package, private, protected, public;
- R return;
- S short, static, super switch, synchronized;
- T this, throw, throws, transient, true, try;
- V var, void;
- W while, with;

Les variables

■ Les variables locales et globales:

- Les variables déclarées tout au début du script, en dehors et avant toutes fonctions, seront toujours globales, qu'elles soient déclarées avec **var** ou de façon **contextuelle**. On pourra donc les exploiter partout dans le script.
- Dans une fonction, une variable déclarée par le mot clé **var** aura une portée limitée à cette seule fonction. On ne pourra donc pas l'exploiter ailleurs dans le script.
- Par contre, toujours dans une fonction, si la variable est déclarée **contextuellement** sans utiliser le mot **var**, sa portée sera globale.

Les opérateurs

- **Les opérateurs de calcul:** la valeur initiale de x sera toujours égale à 11

Signe	Nom	Signification	Exemple Résultat
+	plus	addition	x + 3 14
-	moins	soustraction	x - 3 8
*	multiplié par	multiplication	x*2 22
/	divisé par	division	x /2 5.5
%	modulo	reste de la division par	x%5 1
=	a la valeur	affectation	x=5 5

Les opérateurs

■ Les opérateurs de comparaison:

Signe	Nom	Exemple	Résultat
==	égal	x==11	true
<	inférieur	x<11	false
<=	inférieur ou égal	x<=11	true
>	supérieur	x>11	false
>=	supérieur ou égal	x>=11	true
!=	différent	x!=11	false

Les opérateurs

- **Les opérateurs associatifs:** la valeur initiale de x sera toujours égale à 11 et y sera toujours égale à 5

Signe	Description	Exemple	Signification	Résultat
+=	plus égal	$x += y$	$x = x + y$	16
-=	moins égal	$x -= y$	$x = x - y$	6
*=	multiplié égal	$x *= y$	$x = x * y$	55
/=	divisé égal	$x /= y$	$x = x / y$	2.2

Les opérateurs

■ Les opérateurs logiques:

- Aussi appelés opérateurs booléens, ses opérateurs servent à vérifier deux ou plusieurs:

Signe	Description	Exemple
&&	et	(condition1) && (condition2)
	ou	(condition1) (condition2)

Les opérateurs

- **Les opérateurs d'incrémentation:** la valeur initiale de x sera toujours égale à 3
 - Ces opérateurs vont augmenter ou diminuer la valeur de la variable d'une unité. Ce qui sera fort utile, par exemple, pour mettre en place des boucles.

Signe	Description	Exemple	Signification	Résultat
x++	incrémentation (x++ est le même que x=x+1)	y = x++	3 puis plus 1	4
x--	décrémentation (x-- est le même que x=x-1)	y = x--	3 puis moins 1	2

Les opérateurs

■ La priorité des opérateurs JavaScript:

- Les opérateurs s'effectuent dans l'ordre suivant de priorité du degré de priorité le plus faible ou degré de priorité le plus élevé. Dans le cas d'opérateurs de priorité égale, de gauche à droite.

➤ ,	virgule ;	➤ < <= >= >	relationnel;
➤ = += -= *= /= %=	affectation;	➤ + -	addition, soustraction;
➤ ? :	opérateur conditionnel;	➤ * /	multiplier, diviser;
➤	ou logique;	➤ ! - ++ --	unaire;
➤ &&	et logique;	➤ ()	parenthèses;
➤ == !=	égalité;		

Les conditions

- A un moment ou à un autre de la programmation, on aura besoin de tester une condition. Ce qui permettra d'exécuter ou non une série d'instructions.

- Dans sa formulation la plus simple, l'expression **if** se présente comme suit:

```
if (condition vraie) {  
    une ou plusieurs instructions;  
}
```

- De façon un peu plus évoluée, il y a l'expression **if...else**:

```
if (condition vraie) {  
    instructions1;  
}  
else {  
    instructions2;  
}
```

Les conditions

- Grâce aux opérateurs logiques "**et**" et "**ou**", l'expression de test pourra tester une association de conditions.
- Ainsi **if ((condition1) && (condition2))**, testera si la **condition1** et la **condition2** est réalisée.
- Et **if((condition1) || (condition2))**, testera si une au moins des conditions est vérifiée.
- il y a aussi : **(expression) ? instruction a : instruction b;**
- Si l'**expression** entre parenthèse est vraie, l'**instruction a** est exécutée. Si l'**expression** entre parenthèses retourne faux, c'est l'**instruction b** qui est exécutée.

Les boucles

- L'expression **for** permet d'exécuter un bloc d'instructions un certain nombre de fois en fonction de la réalisation d'un certain critère. Sa syntaxe est :

```
for (valeur initiale ; condition ; progression) {  
    instructions;  
}
```

- Prenons un exemple concret:

```
for (i=0, i<10, i++) {  
    document.write(i + "<BR>")  
}
```

Les boucles

- L'instruction **while** permet d'exécuter une instruction (ou un groupe d'instructions) un certain nombre de fois.

```
while (condition vraie){  
    continuer à faire quelque chose  
}
```

- Aussi longtemps que la condition est vérifiée, JavaScript continue à exécuter les instructions entre les accolades. Une fois que la condition n'est plus vérifiée, la boucle est interrompue et on continue le script. Prenons un exemple.

```
compt=1;  
while (compt<5) {  
    document.write ("ligne : " + compt + "<BR>");  
    compt++;  
}  
document.write("fin de la boucle");
```


Les boucles

- **Break:** L'instruction **break** permet d'interrompre prématurément une boucle **for** ou **while**. Pour illustrer ceci, reprenons notre exemple:

```
compt=1;
while (compt<5) {
    if (compt == 4)
        break;
    document.write ("ligne : " + compt + "<BR>");
    compt++;
}
document.write("fin de la boucle");
```

- Le fonctionnement est semblable à l'exemple précédent sauf lorsque le compteur vaut 4. A ce moment, par le **break**, on sort de la boucle et "fin de boucle" est affiché.

Les boucles

- **Continue:** L'instruction continue permet de sauter une instruction dans une boucle **for** ou **while** et de continuer ensuite le bouclage sans sortir de celui-ci comme le fait **break**. Reprenons notre exemple:

```
compt=1;
while (compt<5) {
    if (compt == 3){
        compt++;
        continue;
    }
    document.write ("ligne : " + compt + "<BR>");
    compt++;
}
document.write("fin de la boucle");
```

- Lorsque le compteur vaut 3, par l'instruction continue, on saute l'instruction `document.write (ligne : 3 n'est pas affichée)` et on continue la boucle.

Les fonctions

- Une fonction est un groupe de ligne(s) de code de programmation destiné à exécuter une tâche bien spécifique et que l'on pourra, si besoin est, utiliser à plusieurs reprises.

- En JavaScript, il existe deux types de fonctions :
 - Les fonctions propres à JavaScript. On les appelle des "méthodes". Elles sont associées à un objet bien particulier comme c'était le cas de la méthode **Alert()** avec l'objet **window**.

 - Les fonctions écrites par vous-même pour les besoins de votre script. C'est à celles-là que nous nous intéressons maintenant.

Les fonctions

■ Déclaration des fonctions:

- Pour déclarer ou définir une fonction, on utilise le mot (réservé) **function**. La syntaxe d'une déclaration de fonction est la suivante :

```
function nom_de_la_fonction(arguments) {  
    ... code des instructions ...  
}
```

- La mention des arguments est facultative mais dans ce cas les parenthèses doivent rester. C'est d'ailleurs grâce à ces parenthèses que l'interpréteur JavaScript distingue les variables des fonctions.

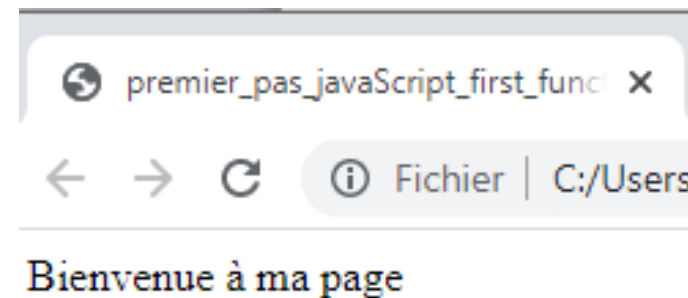
Les fonctions

■ L'appel d'une fonction:

- L'appel d'une fonction se fait le plus simplement du monde par le nom de la fonction avec les parenthèses. Soit par exemple **nom_de_la_fonction**(arguments) ;

- **Les fonctions dans <HEAD>...<HEAD>:** Dans cet exemple, on définit dans les balises HEAD, une fonction appelée **message()** qui affiche le texte "**Bienvenue à ma page**". cette fonction sera appelée au chargement de la page voir **onLoad=....** dans le tag **<BODY>**.

```
premier_pas_javascript_first_function.html
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE="Javascript">
      function message() {
        document.write("Bienvenue à ma page");
      }
    </SCRIPT>
  </head>
  <BODY onLoad="message()">
  </BODY>
</html>
```



Les fonctions

■ Passer des valeurs à une fonction et retourner une valeur:

- On peut passer un paramètre ou plusieurs paramètres à une fonction, on les séparant pas des virgules.

```
function nom_de_la_fonction(arg1, arg2, arg3) {  
    ... code des instructions ...  
}
```

- Le principe de retourner une valeur est simple. Pour renvoyer un résultat, il suffit d'écrire le mot clé **return** suivi de l'expression à renvoyer.

```
function cube(nombre) {  
    var cube = nombre*nombre*nombre;  
    return cube;  
}
```

- Précisons que l'instruction **return** est facultative et qu'on peut trouver plusieurs **return** dans une même fonction.

Les fonctions

■ Passer des valeurs à une fonction et retourner une valeur:

- On peut passer un paramètre ou plusieurs paramètres à une fonction, on les séparant pas des virgules.

```
function nom_de_la_fonction(arg1, arg2, arg3) {  
    ... code des instructions ...  
}
```

- Le principe de retourner une valeur est simple. Pour renvoyer un résultat, il suffit d'écrire le mot clé **return** suivi de l'expression à renvoyer.

```
function cube(nombre) {  
    var cube = nombre*nombre*nombre;  
    return cube;  
}
```

- Précisons que l'instruction **return** est facultative et qu'on peut trouver plusieurs **return** dans une même fonction.

Les fonctions

■ Variables locales et variables globales:

- Une variable déclarée dans une fonction par le mot clé **var** aura une portée limitée à cette seule fonction. On ne pourra donc pas l'exploiter ailleurs dans le script. On l'appelle donc variable locale.

```
function cube(nombre) {  
    var cube = nombre*nombre*nombre  
}
```

- Ainsi la variable **cube** dans cet exemple est une variable locale. Si vous y faites référence ailleurs dans le script, cette variable sera inconnue pour l'interpréteur JavaScript et affichera un message d'erreur.

Les fonctions

■ Variables locales et variables globales:

- Si la variable est déclarée contextuellement, sans utiliser le mot **var**, sa portée sera globale et pour être tout à fait précis, une fois que la fonction aura été exécutée.

```
function cube(nombre) {  
    cube = nombre*nombre*nombre;  
}
```

- Les variables déclarées tout au début du script, en dehors et avant toutes fonctions, seront toujours globales, qu'elles soient déclarées avec **var** ou de façon contextuelle.

```
var cube=1;  
function cube(nombre) {  
    var cube = nombre*nombre*nombre;  
}
```

Les événements

- Avec les événements et surtout leur gestion, nous abordons le côté "magique" de JavaScript.
- Les événements JavaScript, associés aux fonctions, aux méthodes et aux formulaires, ouvrent grand la porte pour une réelle interactivité de vos pages.
- Les événements en JavaScript:
 - ✓ Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément: **Clik**
 - ✓ Lorsque la page est chargée par le browser ou le navigateur. **Load**
 - ✓ Lorsque l'utilisateur quitte la page. **Unload**

Les événements

■ Les événements en JavaScript:

- ✓ Lorsque l'utilisateur place le pointeur de la souris sur un lien ou tout autre élément. **MouseOver**.
- ✓ Lorsque le pointeur de la souris quitte un lien ou tout autre élément. **MouseOut**.
- ✓ Lorsque un élément de formulaire a le focus c-à-d devient la zone d'entrée active. **Focus**.
- ✓ Lorsque un élément de formulaire perd le focus c-à-d que l'utilisateur clique hors du champs et que la zone d'entrée n'est plus active. **Blur**.

Les événements

■ Les événements en JavaScript:

- ✓ Lorsque la valeur d'un champ de formulaire est modifiée. **Change.**
- ✓ Lorsque l'utilisateur sélectionne un champ dans un élément de formulaire. **Select.**
- ✓ Lorsque l'utilisateur clique sur le bouton Submit pour envoyer un formulaire. **Submit.**

Les événements

■ Les gestionnaires d'événements:

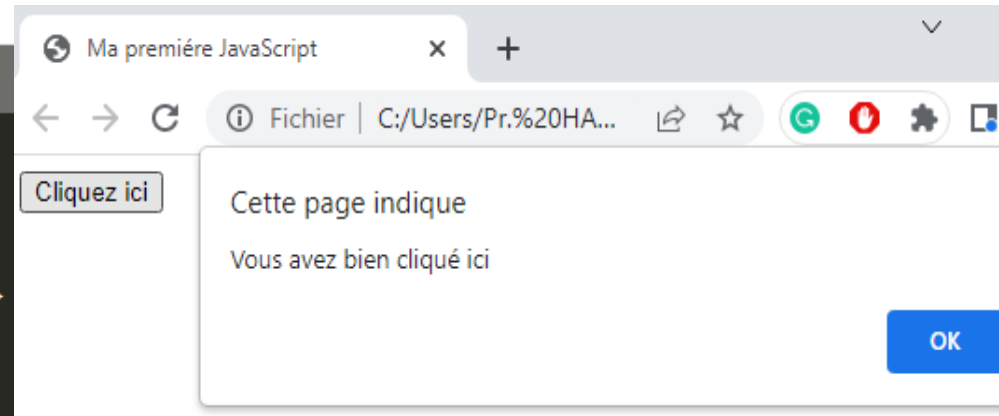
- Pour être efficace, il faut qu'à ces événements soient associées les actions prévues par vous. C'est le rôle des gestionnaires d'événements. La syntaxe est:
 - ✓ `onÉvénement="fonction()"`
- Par exemple, `onClick="alert('Vous avez cliqué sur cet élément')"`. De façon littérale, au clic de l'utilisateur, ouvrir une boîte d'alerte avec le message indiqué.

Les événements

■ Les gestionnaires d'événements:

- **OnClick:** Événement classique en informatique, le clic de la souris. Le code de ceci est :

```
premier_pas_javascript_evenement_onclick.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
  </head>
  <BODY>
    <FORM>
      <INPUT TYPE="button" VALUE="Cliquez ici"
        onClick="alert('Vous avez bien cliqué ici')">
    </FORM>
  </BODY>
</html>
```

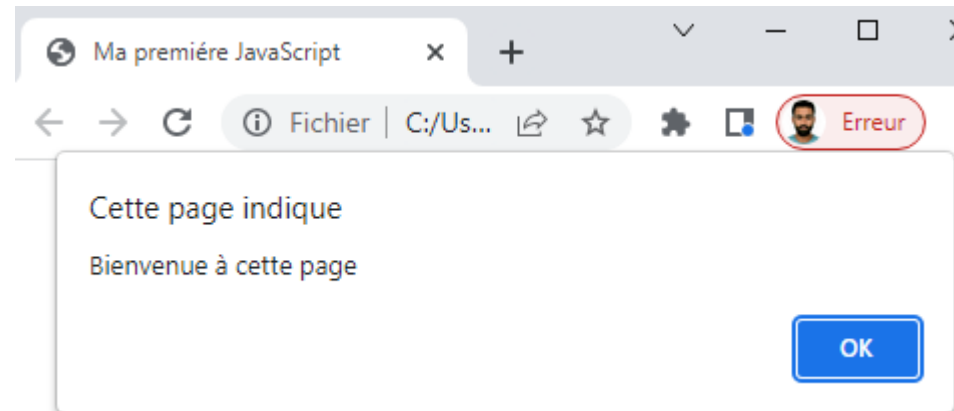


Les événements

■ Les gestionnaires d'événements:

- **onLoad et onUnload**: L'événement **Load** survient lorsque la page a fini de se charger. A l'inverse, **Unload** survient lorsque l'utilisateur quitte la page.
- Les événements **onLoad** et **onUnload** sont utilisés sous forme d'attributs de la balise **<BODY>** ou **<FRAMESET>**.

```
premier_pas_javascript_evenement_onload_unload.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function bienvenue() {
        alert("Bienvenue à cette page");
      }
      function au_revoir() {
        alert("Au revoir");
      }
    </SCRIPT>
  </head>
  <BODY onload='bienvenue()' onunload='au_revoir()'>
  </BODY>
</html>
```



Les événements

■ Les gestionnaires d'événements:

➤ **onmouseover** et **onmouseout**:

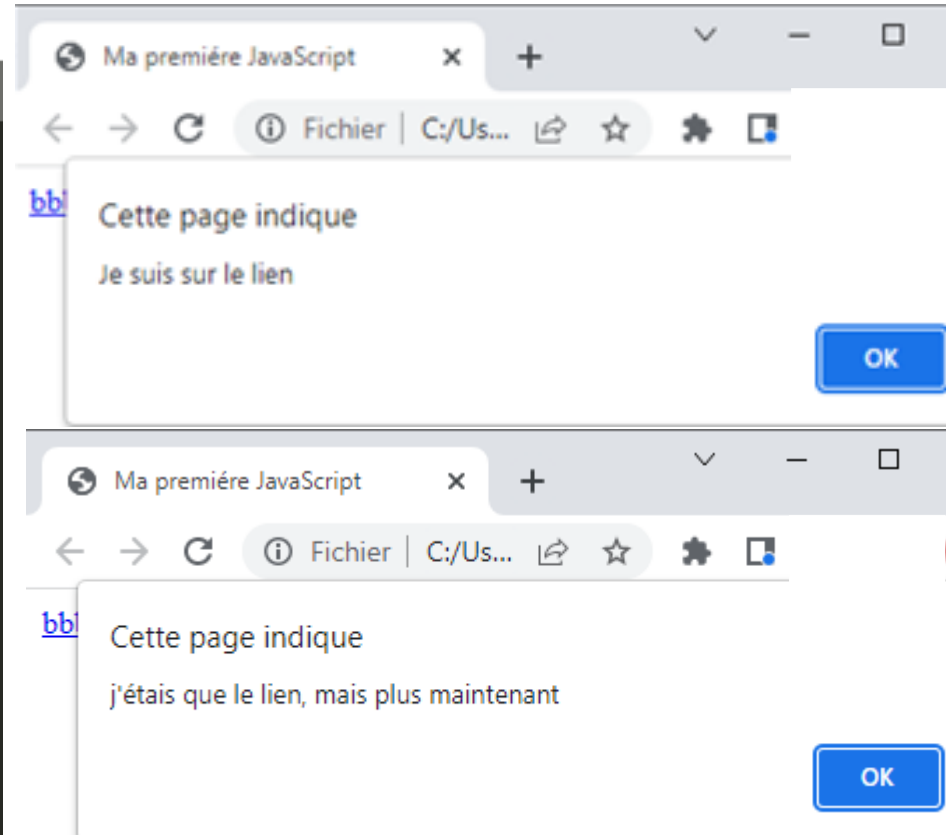
- ✓ L'événement **onmouseover** se produit lorsque le pointeur de la souris passe au dessus (sans cliquer) d'un lien ou d'une image.
- ✓ L'événement **onmouseout**, généralement associé à un **onmouseover**, se produit lorsque le pointeur quitte la zone sensible d'un lien ou une image.

Les événements

■ Les gestionnaires d'événements:

➤ onMouseOver et onMouseOut:

```
premier_pas_javascript_evenement_onmouseover_onmouseout.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function mouseOver() {
        alert("Je suis sur le lien");
      }
      function mouseOut() {
        alert("j'étais que le lien, mais plus maintenant");
      }
    </SCRIPT>
  </head>
  <BODY>
    <a href="txt" onMouseOver='mouseOver()' onMouseOut='
mouseOut()'>bbbbbb</a>
  </BODY>
</html>
```

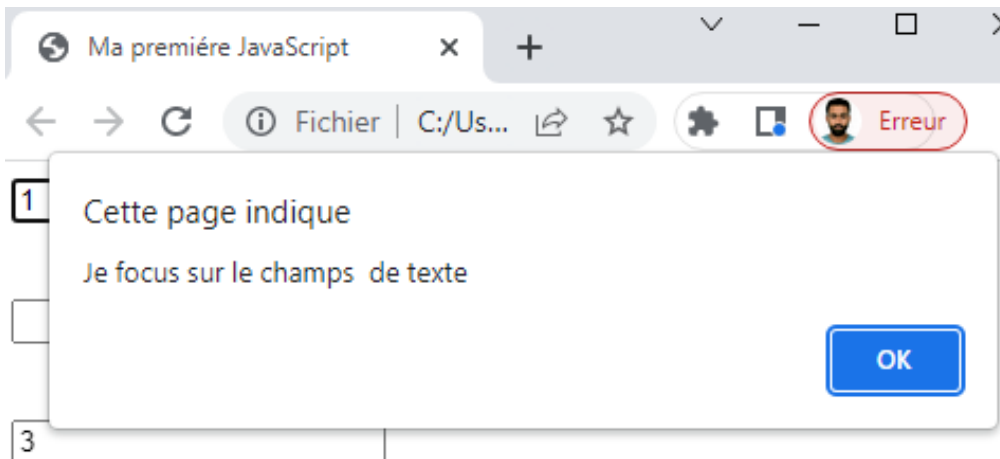


Les événements

■ Les gestionnaires d'événements:

➤ onFocus:

- ✓ L'événement **onFocus** survient lorsqu'un champ de saisie a le focus c.-à-d. quand son emplacement est prêt à recevoir ce que l'utilisateur a l'intention de taper au clavier.
- ✓ C'est souvent la conséquence d'un clic de souris ou de l'usage de la touche "Tab".



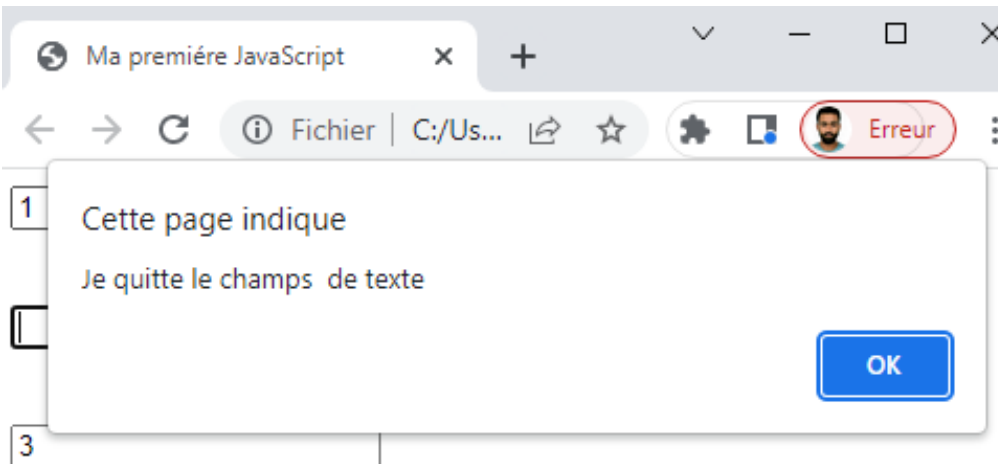
```
premier_pas_javascript_evenement_onFocus.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function focuss() {
        alert("Je focus sur le champs de texte");
      }
      function blure() {
        alert("Je quitte le champs de texte");
      }
    </SCRIPT>
  </head>
  <BODY>
    <input type="text" value="1"><br><br><br>
    <input type="text" onFocus='focuss()'><br><br><br>
    <input type="text" value="3" >
  </BODY>
</html>
```

Les événements

■ Les gestionnaires d'événements:

➤ onBlur :

- ✓ L'événement **onBlur** a lieu lorsqu'un champ de formulaire perd le focus. Cela se produit quand l'utilisateur ayant terminé la saisie qu'il effectuait dans une case, clique en dehors du champ ou utilise la touche "Tab" pour passer à un champ. Cet événement sera souvent utilisé pour vérifier la saisie d'un formulaire.



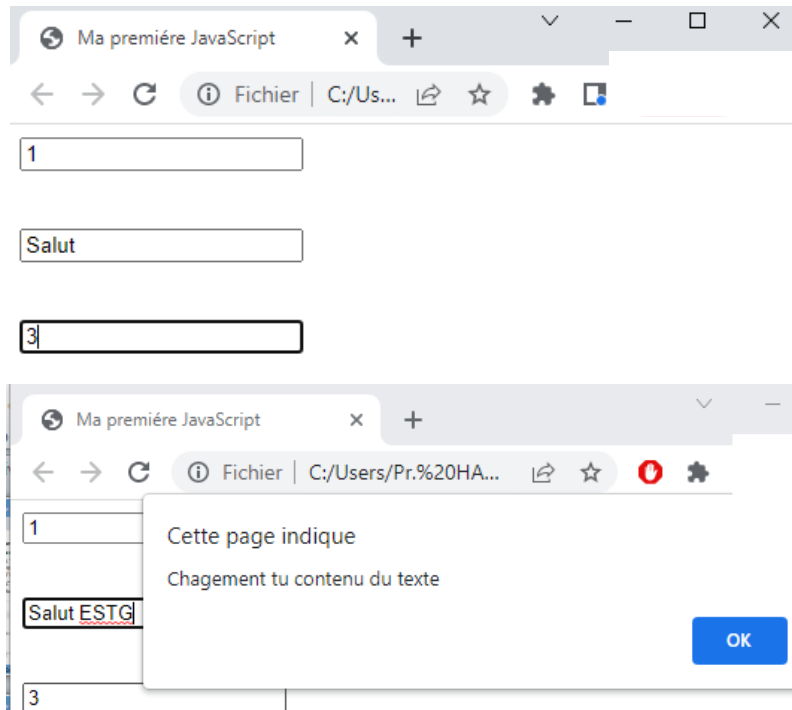
```
premier_pas_javascript_evenement_onBlur.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function focuss() {
        alert("Je focus sur le champs de texte");
      }
      function bluree() {
        alert("Je quitte le champs de texte");
      }
    </SCRIPT>
  </head>
  <BODY>
    <input type="text" value="1" autofocus="true"><br><br>
    <input type="text" onBlur='bluree()'><br><br><br>
    <input type="text" value="3" > |
  </BODY>
</html>
```

Les événements

■ Les gestionnaires d'événements:

➤ onChange :

- ✓ Cet événement s'apparente à l'événement **onBlur** mais avec une petite différence. Non seulement la case du formulaire doit avoir perdu le focus mais aussi son contenu doit avoir été modifié par l'utilisateur.



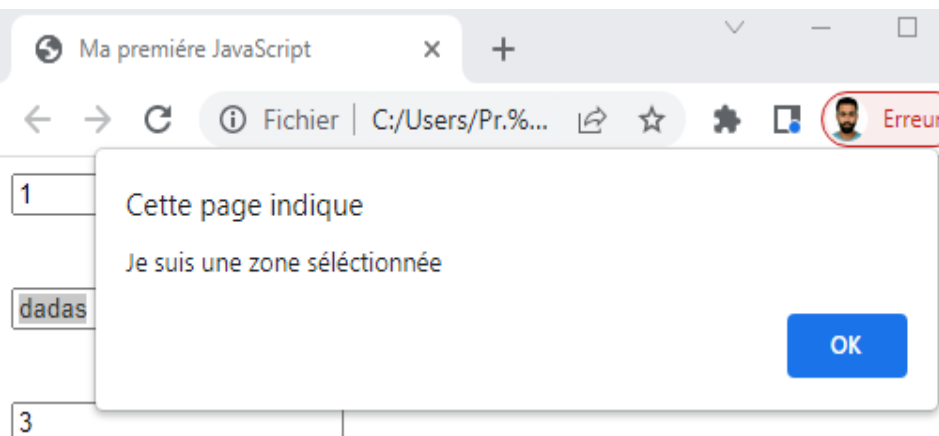
```
premier_pas_javascript_evenement_onChange.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function changes() {
        alert("Chagement tu contenu du texte");
      }
    </SCRIPT>
  </head>
  <BODY>
    <input type="text" value="1" autofocus="true"><br><br>
    <input type="text" onChange='changes()'><br><br><br>
    <input type="text" value="3" >
  </BODY>
</html>
```

Les événements

■ Les gestionnaires d'événements:

➤ onSelect :

- ✓ Cet événement se produit lorsque l'utilisateur a sélectionné tout ou partie d'une zone de texte dans une zone de type **text** ou **textarea**.



```
premier_pas_javascript_evenement_onSelect.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function selects() {
        alert("Je suis une zone sélectionnée");
      }
    </SCRIPT>
  </head>
  <BODY>
    <input type="text" value="1" autofocus="true"><br><br><br>
    <input type="text" onSelect='selects()'><br><br><br>
    <input type="text" value="3" >
  </BODY>
</html>
```

Les événements

■ Gestionnaires d'événement disponibles en JavaScript:

Objets

Fenêtre:

Lien hypertexte:

Élément de texte:

Élément de zone de texte:

Élément bouton:

Case à cocher:

Bouton Radio:

Liste de sélection:

Bouton Submit:

Bouton Reset:

Gestionnaires d'événement disponibles

onLoad, onUnload;

onClick, onMouseOver, onMouseOut;

onBlur, onChange, onFocus, onSelect;

onBlur, onChange, onFocus, onSelect;

onClick;

onClick;

onClick;

onBlur, onChange, onFocus;

onClick;

onClick;

Les formulaires

- Un formulaire est l'élément Html déclaré par les balises **<FORM></FORM>**.
- Un formulaire contient un ou plusieurs éléments que nous appellerons des contrôles. 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.
 - Une erreur classique en JavaScript est, emporté par son élan, d'oublier de déclarer la fin du formulaire **</FORM>** après avoir incorporé un contrôle.

Les formulaires

- **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.

Les formulaires

■ Lire et écrire une valeur dans une zone de texte:

- Pour lire le texte saisi dans la zone texte, voilà le syntaxe correspond:

`document.[Name_form].[Name_input].value`

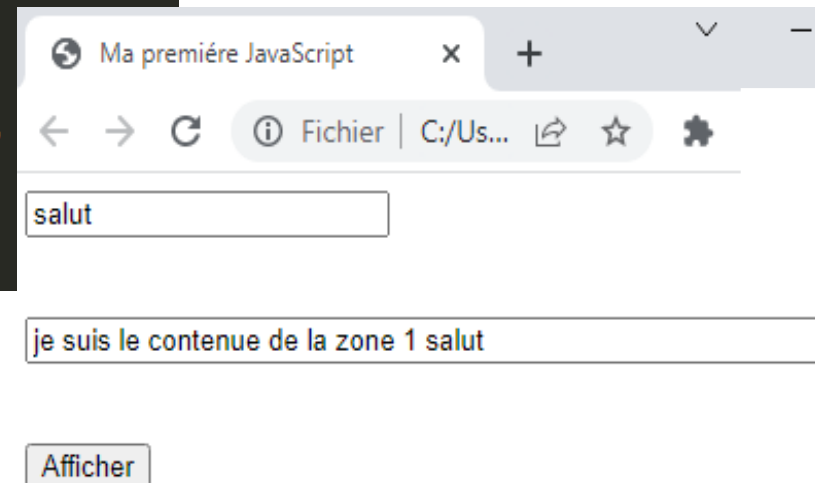
- Pour écrire du texte dans une zone texte, voilà le syntaxe correspond:

`document.[Name_form].[Name_input].value=" votre texte "`

Les formulaires

■ Lire et écrire une valeur dans une zone de texte:

```
premier_pas_javascript_form_input_text.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function afficher_ecrire(form1){
        var zone1 = document.form1.zone1.value;
        document.form1.zone2.value="je suis le contenu de la zone 1 "+zone1;
      }
    </SCRIPT>
  </head>
  <BODY>
    <form name="form1">
      <input type="text" name="zone1"><br><br><br>
      <input type="text" name="zone2" SIZE=50<br><br><br>
      <INPUT TYPE="button" NAME="bouton" VALUE="Afficher" onClick="
        afficher_ecrire(form1)"><BR>
    </form>
  </BODY>
</html>
```



The screenshot shows a web browser window titled "Ma première JavaScript". The address bar shows "Fichier | C:/Us...". The form contains two text input fields and a button. The first text field, with name "zone1", contains the text "salut". The second text field, with name "zone2", contains the text "je suis le contenu de la zone 1 salut". Below the text fields is a button labeled "Afficher".

Les formulaires

- **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.

Les formulaires

The image shows a web browser window with a single tab titled "Ma première JavaScript". The address bar shows the file path "C:/Us...". A JavaScript alert dialog is displayed in the foreground with the message "Vous avez choisi la proposition 2" and an "OK" button.

The background shows the source code of the file "premier_pas_javascript_form_button_radio.html". The code is as follows:

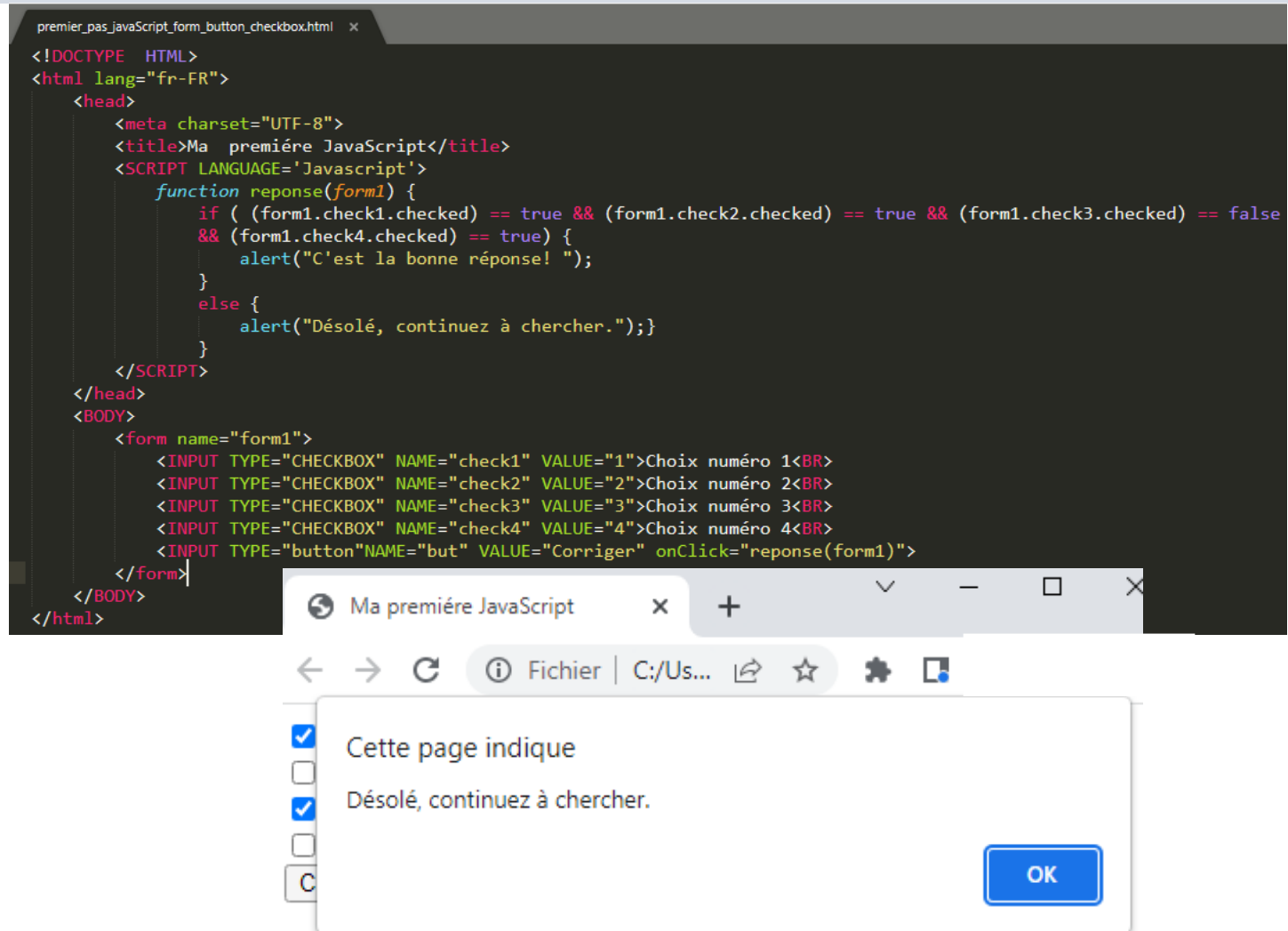
```
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function choixprop(form1) {
        if (form1.choix[0].checked) { alert("Vous avez choisi la proposition " + form1.choix[0].value) };
        if (form1.choix[1].checked) { alert("Vous avez choisi la proposition " + form1.choix[1].value) };
        if (form1.choix[2].checked) { alert("Vous avez choisi la proposition " + form1.choix[2].value) };
      }
    </SCRIPT>
  </head>
  <BODY>
    <form name="form1">
      <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 ?" onClick="choixprop(form1)">
    </form>
  </BODY>
</html>
```

Les formulaires

- **Les boutons case à cocher (checkbox):** Les boutons case à cocher sont utilisés pour noter un ou plusieurs 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.

Les formulaires



The screenshot shows a web browser window titled "Ma première JavaScript" displaying a form with four checkboxes and a button. The form is titled "Choix numéro 1" through "Choix numéro 4". The button is labeled "Corriger" and has an onClick event that calls the "reponse(form1)" function. The browser shows an alert message: "Désolé, continuez à chercher." (Sorry, continue searching.)

```
premier_pas_javascript_form_button_checkbox.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function reponse(form1) {
        if ( (form1.check1.checked) == true && (form1.check2.checked) == true && (form1.check3.checked) == false
          && (form1.check4.checked) == true) {
          alert("C'est la bonne réponse! ");
        }
        else {
          alert("Désolé, continuez à chercher.");
        }
      }
    </SCRIPT>
  </head>
  <BODY>
    <form name="form1">
      <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(form1)">
    </form>
  </BODY>
</html>
```

Les formulaires

- **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.

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 boîte 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 boîte.

Les formulaires

The screenshot displays a web browser window with a single tab titled "Ma première JavaScript". The address bar shows the file path "C:/Users/Pr.%20HAM...". The page content includes a form with the label "Entrez votre choix :". The form contains a dropdown menu with the name "list" and three options: "Elément 1", "Elément 2", and "Elément 3". The "Elément 3" option is currently selected. Below the dropdown is a button with the name "b" and the value "Quel est l'élément retenu?". The button has an "onClick" event that calls the "liste(form1)" function. An alert dialog box is open in the foreground, displaying the message "Cette page indique L'élément 3" and an "OK" button. The background shows the source code of the page, which is a simple HTML document with a JavaScript function "liste" that alerts the selected element's index plus one.

```
premier_pas_javascript_form_list_selection.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function liste(form1) {
        alert("L'élément " + (form1.list.selectedIndex + 1));
      }
    </SCRIPT>
  </head>
  <BODY>
    Entrez votre choix :
    <FORM NAME="form1">
      <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(form1)">
    </FORM>
  </BODY>
</html>
```


Les formulaires

- **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.

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 boîte 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 boîte.

Les formulaires

- **Le contrôle textarea:** L'objet textarea est une zone de texte de plusieurs lignes.
- La syntaxe HTML est : où ROWS=x et COLS=y représente le nombre de lignes et le nombre de colonnes respectivement.

```
<FORM>
```

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

```
    texte par défaut
```

```
</TEXTAREA>
```

```
</FORM>
```

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.

Les formulaires

The image shows a web browser window with a single tab titled "Ma première JavaScript". The address bar shows the file path "C:/Us...". A red error icon is visible in the top right corner of the browser window. In the foreground, an alert box is displayed with the text "Cette page indique" followed by two lines of text: "texte par défaut et je suis ici" and "su". The alert box has a blue "OK" button. In the background, the source code of the web page is visible. The code is for a file named "premier_pas_javascript_form_textarea.html". It includes a DOCTYPE declaration, a meta charset declaration, a title "Ma première JavaScript", and a JavaScript function named "textAreas" that takes a form parameter and calls "alert(document.form1.txtarea.value)". The form is defined with a text area named "txtarea" with 20 rows and 20 columns, and a button named "b" with the value "affiche text area" and an "onClick" event that calls "textAreas(form1)".

```
premier_pas_javascript_form_textarea.html x
<!DOCTYPE HTML>
<html lang="fr-FR">
  <head>
    <meta charset="UTF-8">
    <title>Ma première JavaScript</title>
    <SCRIPT LANGUAGE='Javascript'>
      function textAreas(form1) {
        alert(document.form1.txtarea.value);
      }
    </SCRIPT>
  </head>
  <BODY>
    Entrez votre choix :
    <FORM NAME="form1">
      <TEXTAREA NAME="txtarea" ROWS=20 COLS=20>
        texte par défaut
      </TEXTAREA>
      <INPUT TYPE="button" NAME="b" VALUE="affiche text area" onClick="textAreas(form1)">
    </FORM>
  </BODY>
</html>
```

Les formulaires

- **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.

Les formulaires

- **Le contrôle Reset:** 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()**.

Les formulaires

- **Renvoie les données d'un formulaire:** Dans la déclaration d'un formulaire il faut indiquer:
 - **ACTION:** la page ou le script qui récupère l'information.
 - **METHOD:** spécifié comment l'information est envoyée.
 - Ensuite chaque champ de saisie doit avoir un nom pour l'identifier.
 - À l'aide de ce nom, vous pouvez récupérer la valeur entrée par l'utilisateur dans la page HTML définie par l'action.
 - Exemple: `<form method="get" action="page.htm">` On note ici l'utilisation de la méthode « **GET** » obligatoire pour le bon fonctionnement du script. Et l'action qui contient le **URL** de la page qui récupère les valeurs.

Les formulaires

- **Renvoie les données d'un formulaire:** Dans la déclaration d'un formulaire il faut indiquer:
 - **ACTION:** la page ou le script qui récupère l'information.
 - **METHOD:** spécifié comment l'information est envoyée.
 - Ensuite chaque champ de saisie doit avoir un nom pour l'identifier.
 - À l'aide de ce nom, vous pouvez récupérer la valeur entrée par l'utilisateur dans la page HTML définie par l'action.
 - Exemple: `<form method="get" action="page.htm">` On note ici l'utilisation de la méthode « **GET** » obligatoire pour le bon fonctionnement du script. Et l'action qui contient le **URL** de la page qui récupère les valeurs.

FIN.

MERCI DE VOTRE ATTENTION

DES QUESTIONS ?