1. LES EVENEMENTS

1.1. GENERALITES

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. Avec le bouton *Submit* du formulaire, ce sont hélas les seuls. 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.

1.2. LES EVENEMENTS

Passons en revue différents événements implémentés en JavaScript.

Description	Evénement
Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément.	**click**
Lorsque la page est chargée par le navigateur.	**load**
Lorsque l'utilisateur quitte la page.	**unload**
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**
Lorsqu'un élément de formulaire prend le focus.	**focus**
Lorsqu'un élément de formulaire perd le focus.	**blur**
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**

N’appelez jamais la fonction **document.write()** *depuis un événement car cela remplacerait tout le contenu de la page !*

1.3. LES GESTIONNAIRES D'EVENEMENTS

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 :

Dans le code HTML

Vous pouvez attacher un évènement directement sur la balise HTML en spécifiant l'attribut onEvenement.

Le nom de l'évènement doit être *écrit en CamelCase* préfixé par « **on** » :

Evènement	**Attribut HTML**
click	onClick
mouseover	onMouseOver

```
onEvenement="fonction()"
```

Par exemple, `<p onClick="alert('OK')">Clique ici</p>.`

Dans cet exemple, au clic de la souris, une boite d'alerte avec le message indiqué s'ouvre.

Dans le code javascript

La méthode `addEventListener` permet de connecter un évènement sur un objet. Il faut d'abord sélectionner l'objet à partir de la méthode `getElementById` ou `querySelector`. Le nom de l'évènement n'est pas préfixé par « **on** ».

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <p id="button1">Clique moi</p>
  </body>
</html>

<script>
       var element = document.getElementById("button1");
       element.addEventListener("click", function() {
              alert("OK");
       });
</script>
```

Cette méthode permet de rendre le javascript discret (consultez ce lien http://fr.wikipedia.org/wiki/Javascript_discret)

Testez l'exemple ci-dessus.

1.3.1. click

Evénement classique en informatique, le clic de la souris. La syntaxe est :

```
<form>
	<input type="button" value="Cliquez ici" onClick="alert('Vous avez bien cliqué ici')" />
</form>
```

Un lien hypertexte peut être détourné en appelant l'événement onClick.

1.3.2. load et unload

L'événement ***load*** survient lorsque la page a fini de se charger. A l'inverse, l'événement ***unload*** survient lorsque l'utilisateur quitte la page.

Ces événements sont utilisés sous forme d'attributs des balises *<body>* et *<frameset>*. On peut ainsi écrire un script pour souhaiter la bienvenue à l'ouverture d'une page et un petit mot d'au revoir au moment de quitter celle-ci.

```
<html>
	<head>
		<meta charset="utf-8" />
		<title>Exemple onLoad et onUnLoad</title>
		<script >
			function bienvenue() {
				alert("Bienvenue sur Cette page");
			}
			function au_revoir() {
				alert("Au revoir");
			}
		</script>
	</head>
	<body onLoad='bienvenue()' onUnload='au_revoir()'>
		Actualisez cette page (raccourci touche F5)
	</body>
</html>
```

1.3.3. mouseover

L'événement m*ouseover* se produit lorsque le pointeur de la souris passe au-dessus (sans cliquer) d'un lien ou d'une image. Cet événement est fort pratique pour afficher, par exemple, des explications soit dans la barre de statut soit avec une petite fenêtre genre info bulle.

1.3.4. mouseout

L'événement m*ouseout*, généralement associé à un m*ouseover*, se produit lorsque le pointeur quitte la zone sensible (lien ou image).

1.3.5. focus

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

1.3.6. blur

L'événement b*lur* 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.

1.3.7. change

Cet événement s'apparente à l'événement b*lur* 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.

1.3.8. select

Cet événement se produit lorsque l'utilisateur a sélectionné (mis en surbrillance ou en vidéo inverse) tout ou partie d'une zone de texte dans une zone de type *text* ou *textarea*.

1.4. GESTIONNAIRES D'EVENEMENT DISPONIBLES EN JAVASCRIPT

Il nous semble utile dans cette partie "avancée" de présenter la liste des objets auxquels correspondent des gestionnaires d'événement bien déterminés.

Objets	**Balises**	**Evenements**
Fenetre	<body>	load, unload, focus, blur, error
Ancre	<a>	click,mouseover, mouseout
Formulaire	<form>	reset, submit
Eléments de formulaire	<input>, <select>, <textarea>	click, focus, blur, change, select (<select> et <textarea>)
Elément de zone de texte	<area>, <map>	click, focus, blur
Image	<img>	load , abort, error

Il y a beaucoup d'autres gestionnaires d'événement non traités dans ce cours comme les événements liés au clavier : k***eypress, keydown, keyup***.

1.5. LA SYNTAXE DE MOUSEOVER

Le code du gestionnaire d'événement m*ouseover* s'ajoute aux balises de lien :

```
<a href="" onMouseOver="action();">lien</a>
```

Ainsi, lorsque l'utilisateur passe avec sa souris sur le lien, la fonction *action()* est appelée. L'attribut *href* est indispensable. Il peut contenir l'adresse d'une page Web si vous souhaitez que le lien soit actif ou simplement des guillemets si aucun lien actif n'est prévu. Nous reviendrons ci-après sur certains désagréments du codage *href=""*.

Voici un exemple. Par le survol du lien "message important", une fenêtre d'alerte s'ouvre.

Le code est :

```
<body>
	<a href="" onMouseOver="alert('Coucou');">message important</a>
</body>
```

Ou si vous préférez utiliser les balises <head> :

```
<html>
	<head>
		<script language="JavaScript">
			function message(){
				alert("Coucou");
			}
		</script>
	</head>
	<body>
		<a href="" onMouseOver="message();">message important</a>
	</body>
</html>
```

1.6. PROBLEME! ET SI ON CLIQUE QUAND MEME...

Vous avez codé votre instruction m*ouseover* avec le lien fictif *<a href=""... >*, vous avez même prévu un petit texte, demandant gentiment

à l'utilisateur de ne pas cliquer sur le lien et comme de bien entendu celui-ci clique quand même.

Horreur, le navigateur affiche alors l'entièreté des répertoires de la machine ou du site ou affiche un message d'erreur. Ce qui est un résultat non désiré et pour le moins imprévu.

Pour éviter cela, prenez l'habitude de mettre l'adresse de la page encours ou plus simplement le signe # (pour un ancrage) entre les guillemets de *href*. Ainsi, si le lecteur clique quand même sur le lien, au pire, la page encours sera simplement rechargée et sans perte de temps car elle est déjà dans le cache du navigateur.

Prenez donc l'habitude de mettre le code suivant

```
<a href="#" onMouseOver="action()"> lien </a>.
```

1.7. CHANGEMENT D'IMAGES

Avec le gestionnaire d'événement m*ouseover*, on peut prévoir qu'après le survol d'une image, une autre image apparaisse (pour autant qu'elle soit de la même taille).

Le code est relativement simple.

```
<img src="images/bleu_p.jpg"  border="0" width="38" height="20"
      onMouseOver="this.src='images/bleu_a.jpg'"
      onMouseOut="this.src='images/bleu_p.jpg'" />
```

Compléter toujours en JavaScript les attributs *width="x" height="y"* de vos images.

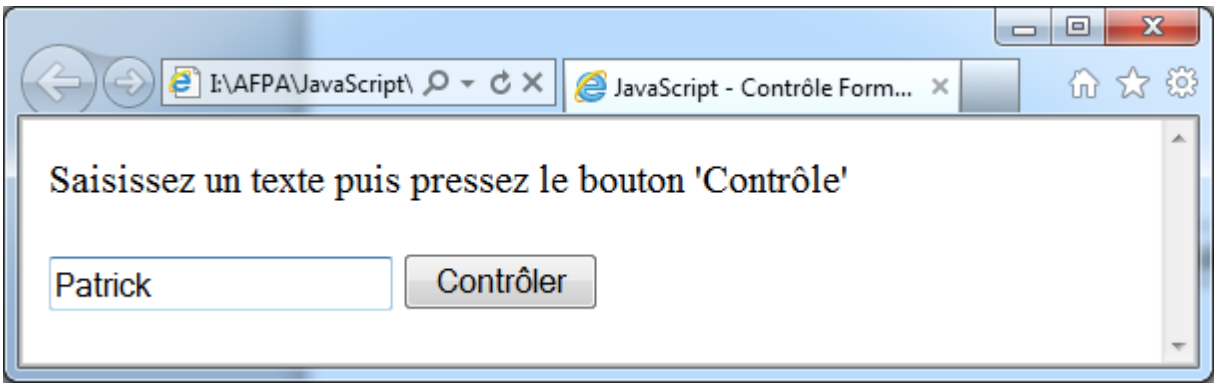
1.8. L'IMAGE INVISIBLE

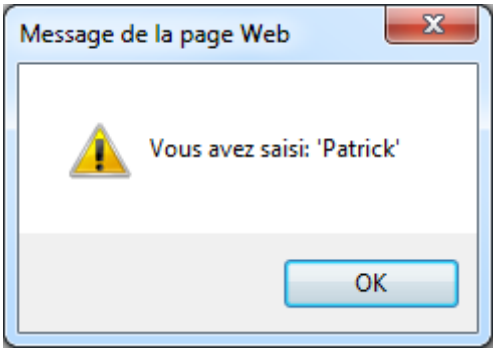
Ce changement d'image ne vous donne-t-il pas des idées?... Petit futé! Et oui, on peut prévoir une image invisible de la même couleur que l'arrière-plan (même transparente). On la place avec malice sur le chemin de la souris de l'utilisateur et son survol peut déclencher, à l'insu de l'utilisateur, un feu d'artifice d'actions de votre choix. Magique le JavaScript ?

Exercice

Le clic sur le bouton « Contrôler » engendre l'appel à la fenêtre d'information.

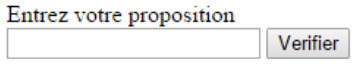
Résultat à obtenir :





Nombre Magique (the Magic Number)

Reprenez l'exercice du nombre magique




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

  </head>
  <body>
      <div id="label1">Entrez votre proposition</div>
      <input id="textBox1" value="">
      <input type="button" id="button1" value="Verifier" onclick="verif();">
  </body>
</html>

<script>

</script>
```

Votre programme doit générer un nombre aléatoire à l'aide de la fonction Math.random.

Ecrivez la fonction verif qui doit vérifier si la saisie de l'utilisateur (dans textBox1) correspond au nombre magique, elle affiche des informations (trop grand, trop petit dans le label1.

Quand votre programme fonctionne, modifiez-le pour rendre le javascript discret.