Javascript - Les événements

Septembre 2015

Qu'appelle-t-on un événement?

Les événements sont des actions de l'utilisateur, qui vont pouvoir donner
lieu à une interactivité. L'événement par excellence est le clic de souris, car c'est le seul que le HTML gère. Grâce au Javascript il est possible d'associer des fonctions, des méthodes à des événements tels que le passage de la souris au-dessus d'une zone, le changement d'une valeur, ...

Ce sont les gestionnaires d'événements qui permettent d'associer une action à un événement.
La syntaxe d'un gestionnaire d'événement est la suivante :

```
on<i>Evenement</i>="Action_Javascript_ou_Fonction();"
```

Les gestionnaires d'événements sont associés à des objets, et leur code s'insèrent
dans la balise de ceux-ci...

Liste des événements

Evénement	Description
Abort (onAbort)	Cet événement a lieu lorsque l'utilisateur interrompt le chargement de l'image
Blur (onBlur)	Se produit lorsque l'élément perd le focus, c'est-à-dire que l'utilisateur clique hors de cet élément, celui-ci n'est alors plus sélectionné comme étant l'élément actif.
Change (onChange)	Se produit lorsque l'utilisateur modifie le contenu d'un champ de données.
Click (onClick)	Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement.

dblclick (onDblclick)	Se produit lorsque l'utilisateur double-clique sur l'élément associé à l'événement (un lien hypertexte ou un élément de formulaire).
dragdrop (onDragdrop)	Se produit lorsque l'utilisateur effectue un *glisser-déposer* sur la fenêtre du navigateur.
error (onError)	Se déclenche lorsqu'une erreur apparaît durant le chargement de la page.
Focus (onFocus)	Se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif
keydown (onKeydown)	Se produit lorsque l'utilisateur appuie sur une touche de son clavier.
keypress (onKeypress)	Se produit lorsque l'utilisateur maintient une touche de son clavier enfoncée.
keyup (onKeyup)	Se produit lorsque l'utilisateur relâche une touche de son clavier préalablement enfoncée.
Load (onLoad)	Se produit lorsque le navigateur de l'utilisateur charge la page en cours
MouseOver (onMouseOver)	Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément
MouseOut (onMouseOut)	Se produit lorsque le curseur de la souris quitte un élément.
Reset (onReset)	Se produit lorsque l'utilisateur efface les données d'un formulaire à l'aide du bouton Reset.
Resize (onResize)	Se produit lorsque l'utilisateur redimensionne la fenêtre du navigateur
Select (onSelect)	Se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ de type "text" ou "textarea"
Submit (onSubmit)	Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire (le bouton qui permet d'envoyer le formulaire)
Unload (onUnload)	Se produit lorsque le navigateur de l'utilisateur quitte la page en cours

Association des événements aux objets

Chaque événement ne peut pas être associé à n'importe quel
objet. Il est évident par exemple qu'un événement *OnChange* ne pourra pas s'appliquer à un lien hypertexte. Voici un tableau récapitulant les objets auxquels peuvent être associés chaque événement :

Evénements	**Objets concernés**

abort	Image
blur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window
change	FileUpload, Select, Submit, Text, TextArea
click	Button, document, Checkbox, Link, Radio, Reset, Select, Submit
dblclick	document, Link
dragdrop	window
error	Image, window
focus	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, window
keydown	document, Image, Link, TextArea
keypress	document, Image, Link, TextArea
keyup	document, Image, Link, TextArea
load	Image, Layer, window
mousedown	Button, document, Link
mousemove	Aucun spécifiquement
mouseout	Layer, Link
mouseover	Area, Layer, Link
mouseup	Button, document, Link
move	window
reset	form
resize	window
select	text, Textarea
submit	Form
unload	window

Les evenements des objets et api

contrairement a la pluparts des evenements utilisateur ces evenement servent surtout a informer sur des etats

Historiquement les premier de ces évènements son ceux qui servent a contrôler le préchargement d'une image et appartenant a l'objet image appelé par new image ils sont au nombre de trois 'onload' pour signifier que l'image a été chargé, 'onerror' pour prévenir que le chargement de l'image a échoué et onAbort pour prevenir que le chargement de l'image a été interompu, un autre évènement bien connu en ajax (requête vers un serveur servant a modifier une page sans rechargement de la page) est

'onreadystatechange' de l'objet XMLHttpRequest , cette évènement sert a informer des différentes états d'une requête.
avec l'avenement du html5 et de ces differentes api le nombre d'evenements s'est vu multiplié de facon importante rien qui l'api audio - video comporte 23 evenements (oncanplay ,onended,onpause...etc)

Associer un evenement a un element

Pour ce faire il suffit des mettre dans la balise le nom de l'evenement et de lui associé une fonction
exemple :

```
<img src='une_image.jpg ' onclick='nom_de_lafonction(parametre1,parametre2)' />
```

parametres1 et parametre2 correspondes a des paramètres que l'on veut transmettre a la fonction on peut executer plusieurs fonctions sur le meme evenement pour ce faire il suffit de rajouter un point virgule.

```
<img src='une_image.jpg' onclick='fonction1() ; fonction2() ; fonction3()' />
```

Associer dynamiquement un évènement a un élément

Consiste a ajouter un évènement a une balise dans la page HTML alors que la page est déjà chargé ou a une balise HTML creé dynamiquement. **il existe trois méthodes, Méthode 1,**

- la plus ancienne de ces méthode consiste a pointer vers l'élément cible et a lui associer l'évènement par le nom de l'évènement et lui adjoindre le nom de la fonction sans les parenthèses.

exemple:

```
document.getElementById('mon_element').onclick=nom_de_la_fonction
```

très utile car compatible même avec les anciens navigateurs et simple a mettre en oeuvre on peut aussi lui adjoindre une fonction anonyme (fonction sans nom) afin de faire suivre l'event ou de lui adjoindre un ou des paramètres. exemple :

```
document.getElementById('mon_element').onclick=function(event){du code du code...etc}
```

le probleme de cette methode est que ne l'on peut adjoindre qu' une seule fonction pour le meme evenement . **Methode 2,**

- la deuxièmement méthode s'appel **'addEventListener'** et fait partie de l'objet **event** que l'on abordera plus bas sa syntaxe est légèrement différente

exemple :

```
document.getElementById('mon_element').addEventListener("click', nom_de_la_fonction, false);
```

pour retirer l'evenement on utilise '**removeEventListener**'

```
document.getElementById('mon_element').removeEventListener("click', nom_de_la_fonction, false);
```

a noter que l'evenement n'est pas precedé par "on" contrairement a la méthode précédente on peut adjoindre plusieurs fonction pour le mêmes évènement , avec cette méthode on peut aussi mettre une fonctions anonymes. **Methode 3,**

- un evenement dans une balise est considéré comme un attribut et pour cette troisième méthode nous allons utiliser **'setAttribute'**

exemple:

```
document.getElementById('mon_element').setAttribute('onclick','nom_de lafonction('paramètre',event));
```

la méthode va insérer dans la balise l'évènement de la même manière que si on le faisait soit même dans la balise on peut rajouter des paramètres et plusieurs fonction pour le même évènement sur l'élément. le problème de cette méthode c'est que si il y a plusieurs fonction associé a l'évènement il ne sera pas possible de retirer l'évènement a l'une des fonction on pourra uniquement toutes les retirer en retirant l'attribut de la balise avec removeAttribute

```
document.getElementById('mon_element').removeAttribute('onclick');
```

Pourquoi ajouter plusieurs fonctions sur le même évènement une de raison les plus connus est au chargement d'une page avec l'évènement "onload". imaginons qu'il y ai plusieurs scripts dans une page et que ces script doivent passer par une fonction d'initialisation .

```
document.onload=fonction_1
document.onload=fonction_2
```

dans cette exemple seule la deuxième fonction sera appelé car le première onload aura été écrasé par le second . il faut donc utiliser addEventListener.

```
document.addEventListener("load', fonction_1, false);
document.addEventListener("load',fonction_2, false);
```

et de cette manière les deux fonctions seront exécuté.

les paramètres de fonction dans un évènement

sujet pas toujours abordé mais qui mérite que l'on si attarde surtout le paramètre "event". les paramètres sont mis a l'intérieur des parenthèses de la fonction et chaque paramètre est séparé par une virgule.

```
fonction la_fonction(param_1,param_2,param_3)
```

les paramètres basiques il s'agit soit de valeurs numérique soit de chaines de caractères. imaginons trois boutons auxquelles on y adjoint un évènement "onclick" avec comme paramètre de fonction une couleur et a coté un div , et a chaque fois que l'on cliquera sur un des boutons le div prendra la couleur du paramètre.

```
<!DOCTYPE html>

<script type="text/javascript">

function couleur(parametre_couleur){

document.getElementById('dv').style.backgroundColor=parametre_couleur;

}

</script>

</head>

<body>

<button onclick='couleur("blue")'>bleu</button>

<button onclick='couleur("green")'>vert</button>

<button onclick='couleur("red")'>rouge</button>

<br><br>

<div id='dv' style='height:50px;width:50px;background-color:black'></div>

</body>

</html>
```

le paramètre this this comme son nom l'indique pointe l'Élément d'où provient l'évènement il evitera de mettre un id a l'element et donc d'utiliser document.getElementById(). dans l'exemple suivant on a un div de couleur vert avec un évènement 'onclick' et comme paramètre a la fonction **this** et quand on clique sur le div une alerte donnera la couleur de fond (backgroundColor) du div.

```
<!DOCTYPE html>

<script type="text/javascript">

function couleur(lui){

var couleur_cible=lui.style.backgroundColor;

alert(couleur_cible);

}

</script>

</head>

<body>

<div onclick='couleur(this)' style='height:50px;width:50px;background-color:green'>clic moi</div>

</body>

</html>
```

le paramètre event le paramètre event fait partie de la gestion globale des événements par le navigateur il est considéré comme un objet et possède donc des méthodes et propriété.

Methodes	Description
preventDefault()	empeche l'action par défaut de se produira pas
stopPropagation()	limite l'évènement a l'Élément cible afin d'éviter la propagation aux parent afin d'éviter pour chaque parents possédant le même évènement le déclenchement de la fonction associé.

capture des évènement de la souris et du clavier

propriete	description
type	retourne le type de l'évènement (onclick; onmoseover...etc)
currentTarget	retourne l'Élément qui a déclenché l'évènement
target	retourne l'Élément survolé dont le parent est l'Élément qui a déclenché l'évènement.
button	Retours le bouton de la souris qui a été cliqué (0:gauche; 1:centre; 2:droite)
clientX	Retour de la coordonnée horizontale du pointeur de la souris, par rapport à la fenêtre du navigateur.
clientY	Retour de la coordonnée verticale du pointeur de la souris, par rapport à la fenêtre du navigateur.
pageX	Retour de la coordonnée horizontale du pointeur de la souris, par rapport à la fenêtre du navigateur plus la position de la barre de scroll.

pageY	Retour de la coordonnée verticale du pointeur de la souris, par rapport à la fenêtre du navigateur plus la position de la barre de scroll.
screenX	Retour de la coordonnée horizontale du pointeur de la souris, par rapport à l'écran
screenY	Retour de la coordonnée verticale du pointeur de la souris, par rapport à l'écran
charCode	Récupère le code de caractère Unicode de la touche qui a généré l'événement onkeypress.
altKey	Retours si oui ou non la touche "ALT" a été pressé quand un événement a été déclenché
shiftKey	Retours si oui ou non la touche "SHIFT" a été pressé quand un événement a été déclenché
ctrlKey	Retours si oui ou non la touche "CTRL" a été pressé quand un événement a été déclenché
metaKey	Retours si oui ou non la touche "meta" a été pressé quand un événement a été déclenché
relatedTarget	Retourne l'élément lié à l'élément qui a déclenché l'événement

pour bien comprendre le fonctionnement de event voici un exemple qui reprend une partie de méthodes et propriétés pour le carré gris qui contient un carré bleue qui lui même contient le carré rouge et le carré vert . on cherche a bien comprendre l'utilisation stopPropagation(); quand on clic dans le carré rouge on appel la fonction lien_a qui comprend stopPropagation() et de meme pour le carré vert qui lui appel la fonction lien_b et qui elle ne comprend pas stopPropagation() . le carré de couleur cyan utilise type et button. dans les deux premiers input on affiche les coordonnés de souris donné par clientX et clientY. les deux autres input utilise les evenements de clavier. et enfin preventDefault() est utilisé dans le document ce qui a pour effet d'empeché d'afficher le menu du clic droit de souris et la selection de texte dans la page

```
<!DOCTYPE html>

<title>source javascript</title>

<script type="text/javascript">

function lien_a(evt){

evt.stopPropagation();

alert(evt.currentTarget.id);

}

function lien_b(evt){
```

```
alert(evt.currentTarget.id);

}

function lien2(evt){

alert(evt.currentTarget.id);

}

function lien3(evt){

alert(evt.currentTarget.id);

}

function ev(evt){ // affiche quelle bouton ade souris a été pressé et le type d'evenement

document.getElementById('cc').value=evt.button;

document.getElementById('dd').value=evt.type;

}

document.onmousemove=function(evt){ //affiche la position de la souris en X et Y par rapport au navigateur

document.getElementById('ee').value=evt.clientX;

document.getElementById('ff').value=evt.clientY;

}

document.onkeypress=function(evt){ //affiche le code clavier de la touche pressé

document.getElementById('gg').value=evt.charCode;

document.getElementById('hh').value=String.fromCharCode(evt.charCode);

}

document.onmousedown=function(evt){evt.preventDefault();} //évite la selection du texte dans le document

document.onclick=function(evt){evt.preventDefault()} //evite l'affichage du menu de souris clic droit
```

```
</script>

</head>

<body>


<div  onmousedown='lien3(event)'id='div_gris' style='display:inline-block;height:300px;width:300px;background-color:gray'>

<div  onmousedown='lien2(event)' id='div_bleu' style='margin:auto;margin-top:30px;height:200px;width:200px;background-color:blue'>


<div  onmousedown='lien_a(event)'id='div_rouge' style='margin:auto;height:80px;width:80px;background-color:red'>texte texte  texte texte </div>

<br>

<div  onmousedown='lien_b(event)'id='div_vert' style='margin:auto;margin-top:5px;height:80px;width:80px;background-color:green'></div>


</div>

</div>

<div  onclick='ev(event)' onmouseover='ev(event)' ondblclick='ev(event)' onmouseout='ev(event)' onmousedown='ev(event)'style='display:inline-block;height:150px;width:150px;background-color:cyan'>

bouton de souris

<br>

<input type='texte' id='cc'></input>

evenement

<br>

<input type='texte' id='dd'></input>

</div>

<br>

<br>

<input type='texte' id='ee'></input>

<input type='texte' id='ff'></input>

<br>

evenement de touche

<br>
```

```
<input type='texte' id='gg'></input>correspond a

<input type='texte' id='hh'></input>

</body>

</html>
```

la fonction anonyme

une fonction anonyme est une fonction sans nom que l'on vas associé a un evenement crée dynamiquement elle est en tout point similaire a une fonction classique, une de ces utilité est de pouvoir ajouter des parametres a un evenement cree dynamiquement

```
document.getElementById('mon_element').onclick=function(param1,param2,event,this){ du code du code}
```

```
document.getElementById('mon_element').addEventListener('click', function(param1,param2,,this){ du code du code}, false);
```

pour addEventListener si on veut utiliser uniquement le parametre event on a pas besoin de passer par une fonction anonyme car il est implicitement ajouté ...

‹ Précédent

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10

Suivant ›

Réalisé sous la direction de Jean-François PILLOU, fondateur de CommentCaMarche.net.