



Chapitre 4 : La programmation événementielle

Objectif : Mettre en place l'aspect interactif grâce aux événements déclenchés par l'utilisateur

Prérequis : La connaissance du langage HTML, CSS et des notions de base en javascript est nécessaire.

Javascript est un langage événementiel. Donc les scripts déclarés dans la page ne sont pas destinés à s'exécuter séquentiellement, ligne après ligne, du début à la fin. Mais ils attendent jusqu'à ce qu'un événement soit détecté pour qu'une partie du code s'exécute.

Les événements correspondent à des actions effectuées soit par un utilisateur, soit par le navigateur lui-même. Par exemple, lorsqu'un utilisateur clique sur un bouton HTML ou lorsque le navigateur va finir de charger une page web, on va parler d'événement.


1. Les événements sans DOM


1.1. Fonctionnement


Les événements sont indiqués dans la balise, d'un formulaire, le plus souvent. Ils apparaissent comme un paramètre supplémentaire et suivent une syntaxe particulière.

Syntaxe :

`<balise onEvent="code">`

 **balise :** désigne le nom de la balise HTML qui supporte l'événement.

 **onEvent** : désigne le gestionnaire d'évènement associé à l'évènement *Event*.

 Le code inséré entre guillemets fait le plus souvent référence à une fonction définie dans les balises **<head>...</head>**. Il peut cependant s'agir d'instructions directement.

Plusieurs gestionnaires d'évènements peuvent être placés dans la même balise. Certaines balises n'appartenant pas à un formulaire peuvent supporter des gestionnaires d'évènement.

Exemple :

```
<a href="http://www.uvs.sn" onClick="alert('vous avez cliqué!');"
onMouseOver="alert('Héhé');" > click?</a>
```

1.2. Quelques évènements

1.2.1. Le clic de souris

Lorsque vous cliquez sur un élément de formulaire de la page que vous consultez.

Le Gestionnaire d'évènement est : **OnClick**

Exemple :

```
<a href="http://www.uvs.sn" onClick="alert('vous avez cliqué!');" > cliquez ici !!!</a>
```

Les balises supportées sont : **<input type="button">**, **<input type="checkbox">**, **<input type="radio">**, **<input type="reset">**, **<input type="submit">**, **<a href..>**

1.2.2. Le chargement

1.2.2.1. Load

Lorsque la page que vous consultez finit de se charger.

Le Gestionnaire d'évènement est : **onLoad**

Exemple :

```
<body onLoad="alert('la page est chargée !');">
```

Les balises supportées sont : **<body>**, **<frameset>**

1.2.2.2. UnLoad

Lorsque vous quittez un document ou une page web. Le Gestionnaire d'évènement est :

onUnLoad.

Exemple :

```
<body onUnLoad="alert('Vous quittez la page !') ;">
```

Les balises supportées sont : **<body>**, **<frameset>**

1.2.2.3. Error

Lorsque le chargement d'une page ou d'une image s'interrompt en erreur. Le Gestionnaire d'évènement est : **onError**

Exemple :

```

```

Les balises supportées sont : **<body>**, **<frameset>**, ****

1.2.3. Abort

Lorsque vous interrompez le chargement d'une image. Le gestionnaire d'évènement est :

onAbort

Exemple :

```

```

Les balises supportées sont : ****

1.2.4. Le passage de la souris

1.2.4.1. MouseOver

Lorsque vous survolez un lien ou une zone d'image activable. Une zone d'image activable est une partie d'image qui a été transformée en lien. Le gestionnaire d'événement est : **onMouseOver**

Exemple :

```
<a href="http://www.google.fr" onMouseOver="alert('Pour aller sur google.fr, cliquer ici');">http://www.google.fr</a>
```

1.2.4.2. MouseOut

Lorsque vous sortez de la zone de survol d'un lien ou d'une zone d'image activable. Le gestionnaire d'événement est : **onMouseOut**

Exemple :

```
<a href="http://www.google.fr" onMouseOut="alert('Vous ne voulez pas y aller ?');">http://www.google.fr</a>
```

1.2.5. Le focus

1.2.5.1. Focus

Lorsque vous activez un contrôle texte ou sélection. Le gestionnaire d'événement est : **onFocus**

Exemple :

```
<input type="text" value="votre nom" name="nom" onFocus="alert('Ecrivez votre nom ici');">
```

1.2.5.2. Blur

Lorsque vous quittez un contrôle texte ou sélection. Le gestionnaire d'événement : **onBlur**

Exemple :

```
input type="text" value="votre nom" name="nom" onBlur="alert('Vous n\'avez rien oublié ?');">
```

1.2.6. Les changements

Lorsque la valeur d'un texte ou d'une option change dans un formulaire. Si vous cliquez dans la zone de texte mais que vous ne touchez pas au texte, rien ne se produit. Le gestionnaire d'évènement est: **onChange**

Exemple :

```
<input type="text" value="votre nom" name="nom" onChange="alert('Vous avez changé votre nom ??') ;">
```

Balises supportées : **<input type="text">**, **<select>**, **<textarea>**, **<input type="password">**

1.2.7. La sélection

Lorsque vous sélectionnez du texte dans un champ de texte. Le gestionnaire d'évènement est : **onSelect**

Exemple :

```
<input type="text" value="votre nom" name="nom" onSelect="alert('Vous avez sélectionné un champ') ;">
```

Balises supportées : **<input type="text">**, **<textarea>**

1.2.8. L'envoi

Lorsque vous cliquez sur un bouton « **submit** » d'un formulaire de type « post » ou « get ».
Le gestionnaire d'évènement est : **onSubmit**

Exemple :

```
<input type="submit" value="Envoyer" name="envoi" onSubmit="alert('C'est parti !') ;">
```

2. Les événements au travers du DOM

La technique classique de gestion des événements pose le problème du mélange des codes

relatifs aux langages de balise HTML ou XHTML et au langage de script JavaScript. Cet aspect nuit à la compréhension et à la maintenabilité des pages Web.

Heureusement, il existe une autre solution pour associer des observateurs d'événements à des balises HTML sans passer par les attributs de ces dernières. Cette solution utilise la technologie DOM, sauf pour le navigateur Internet Explorer, qui propose une approche spécifique.

2.1. Support du DOM niveau 2

La technologie DOM supporte divers événements classifiés en catégories en fonction de leur nom.

Le tableau suivant récapitule ces catégories.

Catégorie	Type	Description
Événement	Events	Couvre l'ensemble des types d'événements.
Événement relatif au HTML	HTMLEvents	Correspond aux événements abort, blur, change, focus, error, load, reset, scroll, select, submit, unload, etc.
Événement relatif à l'interface utilisateur	UIEvents	Correspond aux événements liés à l'interface utilisateur (DOMActivate, DOMFocusIn et DOMFocusOut), ainsi qu'aux événements déclenchés par le clavier (keydown, keypress et keyup). Avec le niveau 3 du DOM, les événements clavier sont gérés dans le type KeyboardEvents.
Événement de la souris	MouseEvents	Correspond aux événements déclenchés par la souris, tels que click, mousedown, mousemove, mouseout, mouseover et mouseup.
Événement de mutation	MutationEvents	Définit les événements impactant la structure logique du modèle.

La technologie DOM met à disposition diverses méthodes afin de gérer les événements. Elles se fondent sur des observateurs pouvant être enregistrés et désenregistrés pour divers événements.

NB: Un **observateur d'Événement** correspond à une fonction ou une méthode d'un objet qui est appelée lors du déclenchement d'un Événement pour un élément d'une page HTML. À cet effet, il doit être enregistré pour un événement. Lorsque l'observateur n'est plus utilisé, il doit être désenregistré.

Les méthodes **addEventListener** et **removeEventListener** permettent de gérer l'enregistrement et le **désenregistrement** de ces observateurs. Elles s'appliquent sur un nœud DOM et prennent en paramètres le nom de l'événement, la fonction à appeler lorsque celui-ci survient et un drapeau définissant dans quelle phase de propagation l'événement doit être traité. La valeur `true` correspond à la phase descendante et `false` à la phase ascendante.

Le code suivant illustre la mise en œuvre de ces deux méthodes (repères ❷ et ❸) afin d'enregistrer et de désenregistrer une fonction (repère ❶) en tant qu'observateur :

```
// Definition de l'observateur
function traitementEvenement(evenement) { ❶
    (...)
}
var monNoeud = document.getElementById("monNoeud");
// Enregistrement d'un observateur
monNoeud.addEventListener("click", traitementEvenement, false); ❷
(...)
// Désenregistrement de l'observateur
monNoeud.removeEventListener("click", traitementEvenement, false); ❸
```

2.1.1. La classe **Event**

Le DOM fournit l'événement déclencheur en paramètre lors de l'appel de la fonction de l'observateur (repère ❶). Cette instance correspond à une instance de la classe **Event**, dont les attributs et méthodes sont respectivement récapitulés aux tableaux suivants.

Attribut	Type	Description
bubbles	Booléen	Indique si l'événement est en phase de propagation ascendante.
cancelable	Booléen	Indique si l'événement peut être annulé.
currentTarget	EventTarget	Correspond au nœud auquel est affecté le gestionnaire d'événement.

eventPhase	Nombre	entier Correspond à un numérique indiquant la phase durant laquelle l'événement a été intercepté. Les valeurs possibles sont CAPTURING_PHASE (valeur 1), AT_TARGET (valeur 2) et BUBBLING_PHASE (valeur 3).
target	EventTarget	Correspond au nœud à partir duquel a été déclenché l'événement.
timeStamp	Nombre entier	Correspond à l'heure à laquelle l'événement est sur venu.
type	Chaîne de caractères	Correspond au type de l'événement.

NB : Les attributs de la classe Event sont en lecture seule.

Méthode	Paramètre	Description
initEvent()	Le type de l'événement	Permet d'initialiser un événement déclenché manuellement par la et deux drapeaux programmation. Les drapeaux en paramètres permettent de spécifier respectivement si l'événement peut être remonté ou annulé.
preventDefault()	-	Permet d'annuler l'événement. Les traitements par défaut associés à l'événement ne sont pas réalisés.
stopPropagation()	-	Permet d'arrêter la propagation de l'événement au cours de la phase de capture ou de remontée.

Le code suivant illustre la mise en œuvre des principaux attributs et méthodes de la classe Event dans une fonction de traitement d'un événement :


```
// Définition de l'observateur
fonction traitementEvenement(evenement) {
    // Affichage du type de l'événement
    alert("Type: "+ evenement.type);
    // Annulation de l'événement
    evenement.preventDefault();
}
```

2.1.2. La classe **MouseEvent**

La classe **MouseEvent** étend la classe **Event** précédente afin de lui ajouter des informations relatives à un événement déclenché par la souris. Le tableau suivant récapitule les attributs supplémentaires de cette classe.

Attribut	Type	Description
altKey, ctrlKey,	Booléens	Indique respectivement si les touches Alt, Ctrl, Meta ou Shift ont été pressées lors metaKey, shiftKey du déclenchement de l'événement.
button	Nombre entier	Correspond au bouton de la souris qui a déclenché l'événement.
clientX, clientY	Nombres entiers	Correspond aux coordonnées de la souris dans la zone affichée par le navigateur (fenêtre ou cadre).
relatedTarget	Booléen	Permet de stocker un autre élément mis en œuvre par l'événement. Il est typiquement renseigné lors des événements de déplacement de la souris (mouseover, mouseenter et mouseout).
screenX, screenY	Nombres entiers	Correspond aux coordonnées de la souris à l'écran.

Le code suivant donne un exemple d'utilisation de cette classe par des observateurs

d'événements déclenchés par la souris :

// Définition de l'observateur

```
function traitementEvenement(evenement) {  
    // Affichage des coordonnées de l'événement  
    alert("Coordonnées du pointeur de la souris: "  
    +evenement.clientX+", "+evenement.clientY);  
    alert("Coordonnées de l'offset de la souris: "  
    +evenement.screenX+", "+evenement.screenY);  
}
```

2.2. Support du navigateur Internet Explorer

Le navigateur Internet Explorer n'implémente pas le niveau 2 de la technologie DOM et possède une façon spécifique de gérer les événements. Bien que les concepts soient similaires à ceux des autres navigateurs, la mise en œuvre des observateurs diffère.

Ainsi, les méthodes **attachEvent** et **detachEvent** permettent de gérer l'enregistrement et le désenregistrement d'observateurs d'événements. Elles s'appliquent sur un nœud DOM et prennent en paramètres le nom de l'événement et la fonction à appeler lorsque celui survient.

NB : Les noms des événements passés en paramètres des méthodes **attachEvent** et **detachEvent** ne correspondent pas exactement à ceux utilisés avec le DOM niveau 2. Internet Explorer impose de les **préfixer** par **on**. Ainsi, l'événement **onclick** correspond dans ce navigateur à l'événement **click** du DOM niveau 2.

Ces deux méthodes s'utilisent de la même manière que celles similaires de la section précédente, comme l'indique le code suivant :

// Définition de l'observateur

```
fonction traitementEvenement() {  
    var evenement = window.event;  
    (...)  
}  
var monNoeud = document.getElementById("monNoeud");  
// Enregistrement d'un observateur  
monNoeud.attachEvent("onclick", traitementEvenement);  
(...)  
// Désenregistrement de l'observateur  
monNoeud.detachEvent("onclick", traitementEvenement);
```

Bien que l'enregistrement et le désenregistrement (respectivement repères et de l'observateur se réalisent de manière similaire qu'avec le DOM, la récupération de l'événement dans la fonction de traitement est spécifique au navigateur.

En effet, l'événement peut être accédé aussi bien par l'attribut **event** de l'objet **window** de la page HTML que par le paramètre de la fonction de l'observateur s'il est spécifié.

De plus, la classe représentant un événement dans Internet Explorer ne possède pas les mêmes attributs et méthodes qu'avec le DOM niveau 2. Cette classe contient des informations relatives aussi bien à la gestion de l'événement qu'au pointeur de la souris.

Ainsi, l'attribut **srcElement** correspond à l'attribut **target** de la classe Event DOM. Les attributs **pageX** et **pageY** ne sont plus présents, mais les valeurs qu'auraient eu ces attributs peuvent être simulées en se fondant sur les attributs **clientX** et **clientY** additionnés aux valeurs de scroll.

Les attributs **offsetX** et **offsetY** sont ajoutés et correspondent aux coordonnées du pointeur de la souris relativement à l'élément sur lequel l'événement est déclenché.

Le code suivant donne un exemple d'utilisation de cette classe par des observateurs d'événements déclenchés par la souris :

```
// Definition de l'observateur
fonction traitementEvenement(evenement) {
    // Affichage des coordonnées de l'événement
    alert("Coordonnées du pointeur de la souris: "
    +evenement.clientX+", "+evenement.clientY);
    alert("Coordonnées de l'offset de la souris: "
    +evenement.offsetX+", "+evenement.offsetY);
}
```

2.3. Les Types d'événements

Parmi les différents types d'événements utilisables dans les navigateurs, nous pouvons tout d'abord distinguer les événements relatifs à l'interface graphique. Ces derniers permettent de détecter ses changements d'état ainsi que ceux de ses composantes.

Le tableau suivant récapitule ces divers types d'événements.

Type	Description
------	-------------

contextmenu	Se produit lorsque l'utilisateur tente d'ouvrir le menu contextuel de la page.
focus, blur	Se produisent respectivement lorsqu'un élément gagne et perd le focus.
load	Se produit lorsqu'une page Web a été chargée complètement.
resize	Se produit lorsque la fenêtre du navigateur est redimensionnée.
scroll	Se produit lorsque l'affichage d'une page Web est scrollée.
unload	Se produit lorsqu'une page Web est déchargée.

Les événements déclenchés par la souris peuvent réagir aussi bien à un déplacement de la souris qu'à l'utilisation de ses boutons. Le tableau suivant récapitule ces divers types d'événements.

Type	Description
click	Se produit sur un clic de la souris pour un élément.
dblclick	Se produit sur un double clic de la souris pour un élément.
mousedown, mouseup	Se produit respectivement lorsqu'un bouton de la souris est enfoncé et relâché.
mousemove	Se produit lorsque le pointeur de la souris est déplacé.
mouseover, mouseout	Se produit respectivement lorsque le pointeur de la souris entre et sort de la zone correspondant à un élément.

Divers événements peuvent se produire sur des éléments de formulaires afin de détecter des modifications dans les valeurs des champs ainsi que sur le formulaire lui-même lors de son

envoi.

Le tableau suivant récapitule ces divers types d'événements.

Type	Description
change	Se produit lors d'un changement dans un élément d'un formulaire. Cet événement n'est la plupart du temps supporté que de manière partielle par les navigateurs.
reset	Se produit lorsque l'utilisateur réinitialise un formulaire.
select	Se produit lorsqu'un utilisateur sélectionne du texte dans un boîte de saisie.
submit	Se produit lorsqu'un utilisateur soumet un formulaire.