L'objet Event

Sommaire

Les repères de lecture

Retour au chapitre
Définition
Objectif(s)
Espace Élèves
Vidéo / Audio
Point important / À retenir
Remarque(s)
Pour aller plus loin
Normes et lois
Quiz

Introduction

L'objet *event* est toujours passé au gestionnaire d'événement.

Ses propriétés fournissent de nombreuses informations utiles sur l'événement courant. C'est ce que nous allons voir ensemble.

I. Propriétés

Vous trouverez ci-dessous la liste des propriétés de l'objet événement :

Tableau n°1

PROPRIÉTÉ	DESCRIPTION
altKey	Contient la valeur true si la touche « ALT » était enfoncée quand l'événement est advenu.
button	Indique quel bouton de la souris a été utilisé lorsque l'événement est advenu.
clientX	Contient la coordonnée horizontale du pointeur de la souris, relative à la fenêtre du navigateur.
clientY	Contient la coordonnée verticale du pointeur de la souris, relative à la fenêtre du navigateur.
ctrlKey	Contient la valeur true si touche « CTRL » était pressée lorsque l'événement est advenu.
keyIdentifier	Contient l'identificateur d'une touche enfoncée.
keyLocation	Contient la localisation de la touche sur le matériel utilisé.
metaKey	Contient la valeur true si la touche « méta » était enfoncée lorsque l'événement est advenu.
relatedTarget	Contient l'objet Elément relatif à un événement mouseenter ou mouseleave.
screenX	Contient la coordonnée horizontale du pointeur de la souris, relative à l'écran, quand l'événement est advenu.
screenY	Contient la coordonnée verticale du pointeur de la souris, relative à l'écran, quand l'événement est advenu.
shiftKey	Contient la valeur true si la touche « majuscule » était enfoncée lorsque l'événement est advenu.
target	Contient l'objet Elément relatif à l'événement.
currentTarget	Contient l'objet Élément sur lequel on « écoute ».
Type	Type de l'événement (exemple).

A. Accéder à l'objet Event

Avec un attribut en ligne (inline attribut)
Si vous utilisez un attribut « en ligne » (inline attribut) pour traiter un événement, une variable objet globale event est initialisée pour vous par le navigateur. Vous pouvez utiliser cette variable immédiatement. Exemple :

```
<button onclick="alert(event)">See the event</button>
```

Fig. 1

B. À partir d'un gestionnaire d'événement (event handler)

Si vous utilisez un gestionnaire d'événement (event handler) pour traiter un événement, la méthode diffère selon que le navigateur suive ou non les recommandations W3C.

1. Pour les navigateurs autres qu'Internet Explorer ou à partir de IE9

Les navigateurs qui suivent le standard W3C passent toujours l'objet event en premier argument du gestionnaire d'événement. Par exemple :

```
element.addEventListener('click', function(event) {
alert(event) ;
}) ;
```

Fig. 2

2. Pour les navigateurs Internet Explorer moins récent que IE9

Internet Explorer fournit un objet global window.event qui concerne le dernier événement advenu. Avant IE9, vous n'avez donc pas besoin de passer un argument dans la fonction du gestionnaire d'événement.

Exemple :

```
{ element.onclick = function() {
 Alert(window.event) ;
}
```

Fig. 3

Autre exemple :

```
element.addEventListener('click', function() {
 alert(window.event) ;
}) ;
```

Fig. 4

3. Solution pour gérer tous les navigateurs (solution « cross-browser »)

Pour que votre code soit compatible quel que soit le navigateur, vous pouvez vous inspirer du code suivant :

```
element.onclick = function(event) {
 event = event || window.event ;
 alert(event);
}
```

Fig. 5

C. Cas pratique

1. Rechargez la page exemple1.html dans votre navigateur Chrome, puis, à partir de la console JavaScript du debugger, recopiez les commandes suivantes (en bleu) :

```
> var element=document.getElementsByTagName('H1')[2]
  undefined
> element.addEventListener('click', function(event) { alert(event.type) });
  undefined
>
```

Fig. 6

2. Cliquez sur le 3e titre d'article ("Article 3"), une boite d'alerte doit s'afficher :

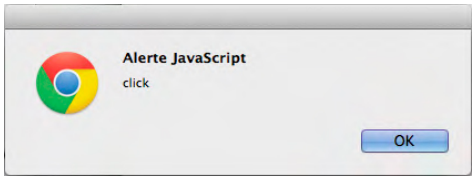

Fig. 7

II. Propagation des évènements dans la hiérarchie du DOM : l'Event bubbling

Dans un document HTML, les éléments sont souvent imbriqués, et si vous programmez une écoute sur un élément parent et qu'un événement advient sur un élément enfant, l'écoute sera déclenchée. C'est ce que l'on nomme en anglais l'Event bubbling.

Lorsqu'un événement advient, le plus proche des éléments pour lesquels une écoute a été programmée est déclenché. Une fois traité, l'écoute du plus proche parent est déclenchée, et ainsi de suite en remontant dans la hiérarchie DOM du document.

A. Cas pratique

Pour vous en persuader, rechargez le document *exemple.html* et utilisez le debugger JavaScript pour ajouter une écoute sur la DIV d'id main et une écoute sur le premier élément **H1**. Le code doit ressembler à ceci :

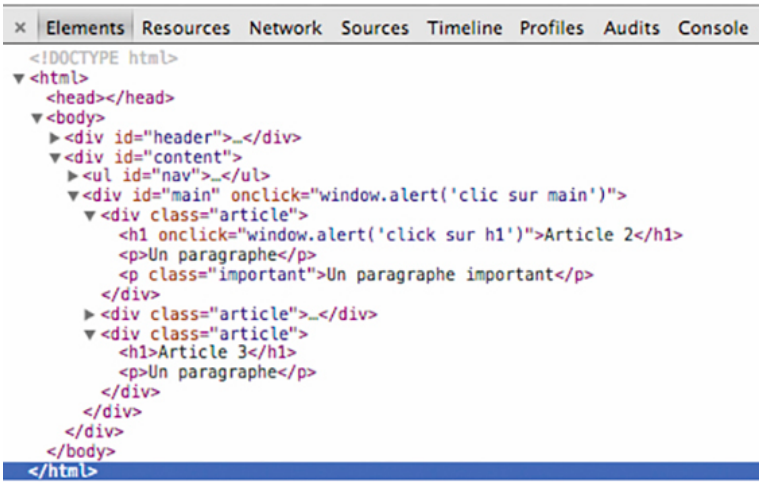

Fig. 8

Maintenant, si vous cliquez sur le premier titre (H1), une première boîte d'alerte apparaît :

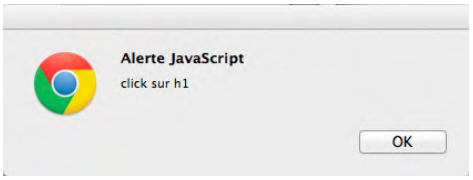

Fig. 9

Lorsque vous cliquez sur le bouton OK, une deuxième boîte d'alerte s'affiche :

Fig. 10

L'événement est bien diffusé en remontant la hiérarchie du DOM.

B. Empêcher la propagation des évènements

Si vous ne souhaitez pas qu'un événement se propage aux éléments parents, il suffit que le code qui traite l'événement utilise la méthode **stopPropagation()** de l'objet event.

*Par exemple, dans l'exercice précédent, si l'on modifie le onclick de l'élément **H1** par :*

```
<h1 onclick="window.alert('clic sur h1'); event.
stopPropagation()">Article 2</h1>
```

Fig. 11

Certains événements ne sont pas propagés, consultez le tableau des évènements ci-dessous, colonne propagée.
L'écoute du clic sur la DIV parent main, ne sera pas déclenchée.
Vérifiez par vous-même.

Pour les versions d'Internet Explorer inférieures à 9, la méthode stopPropagation() n'est pas disponible ; pour empêcher la propagation d'un événement il faut alors positionner la propriété cancelBubble à true :

```
event.cancelBubble = true
```

Fig. 12

Pour rendre votre code compatible, vous pouvez tester la présence de la méthode *stopPropagation()* :

```
If (event.stopPropagation) {
// standard W3C
event.stopPropagation()
} else {
// variante IE
event.cancelBubble = true
}
```

Fig. 13

III. Unobtrusive Javascript

Nous avons vu précédemment comment déclencher l'exécution d'un code JavaScript en ajoutant un attribut (onclick, onmousedown, etc.) à un élément HTML.

Lorsque l'utilisation de JavaScript dans une application est « cosmétique », « mineure », cette approche peut faire l'affaire. Mais aujourd'hui, l'utilisation de JavaScript est devenue, pour beaucoup de sites, essentielle. On préfère alors utiliser des gestionnaires d'événements initialisés lors du chargement du document HTML.

Cette approche a plusieurs avantages, entre autres :
- une séparation du code JavaScript et du code HTML (comme les styles CSS) ;
- une meilleure lisibilité (toute la logique mise en oeuvre par JavaScript est regroupée dans 1 ou plusieurs fichiers de mêmes types ;
- une maintenabilité beaucoup plus grande.

Cette technique est appelée non intrusive (unobtrusive). Elle implique, comme nous l'avons dit, d'initialiser les gestionnaires d'évènements lors du chargement du document, ce que nous allons étudier ensemble maintenant.

A. Initialisation des gestionnaires d'évènements : onload

Pour initialiser des gestionnaires d'événements, mieux vaut attendre que le document HTML soit complètement chargé. En effet, si l'on initialise un gestionnaire d'événements alors que l'élément n'a pas encore été créé par le navigateur, l'écoute ne fonctionnera jamais.
Heureusement, le W3C et les développeurs de navigateurs ont pensé à tout.
Il existe donc un évènement « window.onload » disponible, celui-ci advient lorsque le document est effectivement chargé et prêt à être traité.
Il suffit, donc au début de votre code JavaScript, de créer un gestionnaire d'événement « window.onload » qui initialisera les autres gestionnaires d'événements. Mettons ceci-en pratique.

B. Cas pratique : initialisation des gestionnaires d'évènements

1. Préparez votre fichier HTML en ajoutant une balise Script dans le Head du document et en supprimant les attributs HTML relatifs aux évènements :

```
<!DOCTYPE html>
<html lang=fr>
    <head>
      <meta charset="utf-8">
      <script src="exemple1.js" type="text/javascript"></script>
    </head>
    <body>
        <div id="header">
            <img src="logo.gif" alt="Mon logo">
        </div>
        <div id="content">
            <ul id="nav">
                <li><strong>Accueil</strong></li>
                <li><a href="about.html">A propos</a></li>
                <li><a href="articles.html">Articles</a></li>
                <li><a href="contact.html">Contact</a></li>
            </ul>
            <div id="main">
                <div class="article">
                    <h1>Article 2</h1>
                    <p>Un paragraphe</p>
                    <p class="important">Un paragraphe important</p>
                </div>
                <div class="article">
                    <h1>Article 2</h1>
                    <p>Un paragraphe</p>
                    <p class="important">Un paragraphe important</p>
                </div>
                <div class="article">
                    <h1>Article 3</h1>
                    <p>Un paragraphe</p>
                </div>
            </div>
        </div>
    </body>
</html>
```

Fig. 14

2. Maintenant, vous allez créer le fichier exemple1.js et entrer le code suivant qui attend que le document soit chargé (événement document. onload) pour initialiser les gestionnaires d'évènements onmouseup et onmousedown :

```
window.addEventListener('load', function(event) {
  var element=document.getElementsByTagName('H1')[0];
  element.addEventListener('mousedown', function(event) {
    this.style.color="red";
  });
  element.addEventListener('mouseup', function(event) {
    this.style.color="blue";
  });
});
```

Fig. 15

3. Chargez la page et vérifiez que le script réagit correctement : cliquez sur le premier titre (« Article 2 »), sa couleur doit passer du rouge au bleu.

Vous savez maintenant comment faire de l'unobstrusive JavaScript.
C'est bien, mais pour écrire des scripts vraiment professionnels, il vous faut aller plus loin et apprendre à gérer ce que l'on nomme les closures et les espaces de noms – ce que nous allons voir dans les prochaines leçons.