La gestion d'évènements
En JavaScript

Les évènements sont des actions qui se produisent et auxquelles on va pouvoir répondre en exécutant un code. Par exemple, on va pouvoir afficher ou cacher du texte suite à un clic d'un utilisateur sur un élément, on change la taille d'un texte lors du passage de la souris d'un utilisateur sur un élément.

Les évènements et leur prise en charge sont l'un des mécanismes principaux du JavaScript qui vont nous permettre d'ajouter un vrai dynamisme à nos pages Web.

Présentation des évènements :

En JavaScript, un évènement est une action qui se produit et qui possède deux caractéristiques essentielles :

- C'est une action qu'on peut « écouter », c'est-à-dire une action qu'on peut détecter car le système va nous informer qu'elle se produit ;
- C'est une action à laquelle on peut « répondre », c'est-à-dire qu'on va pouvoir attacher un code à cette action qui va s'exécuter dès qu'elle va se produire.

Exemple : on peut détecter le clic d'un utilisateur sur un bouton d'un document et afficher une boite de dialogue ou un texte suite à ce clic. On parlera donc « d'évènement clic ».

Il existe de nombreux évènements répertoriés en JavaScript (plus d'une centaine). Les évènements qui vont nous intéresser particulièrement sont les évènements liés au Web et donc au navigateur. Ces évènements peuvent être très différents les uns des autres :

- Le chargement du document est un évènement ;
- Un clic sur un bouton effectué par un utilisateur est un évènement ;
- Le survol d'un élément par la souris d'un utilisateur est un évènement ;
- Etc.

Définir des gestionnaires d'évènements

Pour écouter et répondre à un évènement, nous allons définir ce qu'on appelle **des gestionnaires d'évènements**.

Un gestionnaire d'évènements est toujours divisé en deux parties :

- Une partie qui va servir à écouter le déclenchement de l'évènement,
- Et une partie gestionnaire en soi qui va être le code à exécuter dès que l'évènement se produit.

Aujourd'hui, en JavaScript, il existe trois grandes façons d'implémenter un gestionnaire d'évènements :

- On peut utiliser des attributs HTML de type évènement (non recommandé) ;
- On peut utiliser des propriétés JavaScript liées aux évènements ;
- On peut utiliser la méthode **addEventListener()** (recommandé).

a) Utiliser les attributs HTML pour gérer un évènement

L'idée va être ici d'insérer un attribut HTML lié à l'évènement qu'on souhaite gérer directement dans la balise ouvrante d'un élément à partir duquel on va pouvoir détecter le déclenchement de cet évènement.

Ces attributs HTML de « type évènement » possèdent souvent le nom de l'évènement qu'ils doivent écouter et gérer précédé par « on » comme par exemple :

- L'attribut **onclick** pour l'évènement « clic sur un élément » ;
- L'attribut **onmouseover** pour l'évènement « passage de la souris sur un élément » ;
- L'attribut **onmouseout** pour l'évènement « sortie de la souris d'élément » ;
- Etc.

Exemple :

```
<button onclick="alert('Bouton cliqué')"> Cliquez moi !</button>
```

b) Utiliser les propriétés JavaScript pour gérer un évènement

Chaque évènement est représenté en JavaScript par un objet de type **document**.

Ces gestionnaires d'évènements sont des propriétés qui sont de la forme « **on** » + nom de l'évènement géré, c'est-à-dire qui ont des noms similaires aux attributs HTML vus précédemment.

Comment Procédé ?

- ✓ Récupérer l'élément appartenant au document
- ✓ Définir l'événement comme une propriété de l'objet

Exemple :

Dans l'exemple suivant, nous avons un **<bouton>** unique, qui, lorsqu'il est pressé, fera passer l'arrière-plan à une couleur aléatoire:

```
<button>Change color</button>
```

Le JavaScript ressemblera à ça :

```
var btn = document.querySelector('button');

function random(number) {
  return Math.floor(Math.random()*(number+1));
}

btn.onclick = function() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' +
random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}
```

Explication :

Dans ce code, nous stockons une référence au bouton dans une variable appelée btn, en utilisant la fonction `Document.querySelector ()`. Nous définissons également une fonction qui

renvoie un nombre aléatoire. La troisième partie du code est le gestionnaire d'événement. La variable `btn` pointe sur un élément `<button>`, et ce type d'objet a un certain nombre d'événements qui peuvent être déclenchés, et par conséquent, des gestionnaires d'événements sont disponibles. Nous sommes à l'écoute du déclenchement de l'événement click, en définissant la propriété `onclick` du gestionnaire d'événements comme une fonction anonyme contenant du code qui génère une couleur RVB aléatoire et lui affecte la couleur d'arrière-plan `<body>` .

Remarque :

La fonction **`Math.floor(x)`** renvoie le plus grand entier qui est inférieur ou égal à un nombre x.

c) Utiliser la méthode addEventListener() pour gérer un évènement

Le dernier type de mécanisme d'événement fournit aux navigateurs une nouvelle fonction: **addEventListener().**

Cela fonctionne de la même manière que les propriétés du gestionnaire d'événement, mais la syntaxe est évidemment différente.

On va passer deux arguments à cette méthode : le nom d'un évènement qu'on souhaite prendre en charge ainsi que le code à exécuter (qui prendra souvent la forme d'une fonction) en cas de déclenchement de cet évènement.

Exemple :

```
var btn = document.querySelector('button');

function bgChange() {
  var rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' +
random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}

btn.addEventListener('click', bgChange);
```

d) Supprimer un gestionnaire d'évènements avec removeEventListener()

La méthode removeEventListener() de l'interface EventTarget va nous permettre de supprimer un gestionnaire d'évènement déclaré avec addEventListener().

Pour cela, il va suffire de passer en argument le type d'évènement ainsi que le nom de la fonction passée en argument de addEventListener().

Exemple :

```
btn.removeEventListener('click', bgChange);
```

pourquoi supprimer « un écouteur » ?

Ceci n'a pas beaucoup de sens pour les programmes simples et de petite taille, mais pour les programmes plus grands et plus complexes, cela peut améliorer l'efficacité, de nettoyer les

anciens gestionnaires d'événements inutilisés. De plus, par exemple, cela vous permet d'avoir un même bouton qui effectue différentes actions dans des circonstances différentes - tout ce que vous avez à faire est d'ajouter / supprimer des gestionnaires d'événements convenablement.

D'autre part, vous pouvez également enregistrer plusieurs gestionnaires pour le même écouteur. Les deux gestionnaires suivants ne seraient pas appliqués:

```
myElement.onclick = functionA;
myElement.onclick = functionB;
```

Comme la deuxième ligne remplacerait la valeur de onclick définie par le premier. Cependant, ceci fonctionnerait:

```
myElement.addEventListener('click', functionA);
myElement.addEventListener('click', functionB);
```

Les deux fonctions seraient maintenant exécutées lorsque l'élément est cliqué.

Liste des événements Javascript

Il existe des dizaines et des dizaines d'événement en Javascript. Certains sont très anciens mais sont toujours actifs sur les navigateurs récents, d'autres ont été intégrés suite à l'avènement de HTML5 et d'autres sont utiles si on se sert d'AJAX (Asynchronous Javascript And XML). Dans cette liste, nous allons voir les événements les plus utilisés en Javascript. Cependant:

click (onClick)
Cet événement est capturé sur un objet quand on clique dessus. Idéal pour les boutons, images, hyperliens, vidéos...

dblClick (onDblClick)
Quand on clique sur un objet deux fois de suite (double clic).

mouseOver (onMouseOver)
Quand on survole un objet avec le curseur de la sourie.

mouseOut (onMouseOut)
Quand on quitte l'objet avec la sourie après l'avoir survolé.

focus (onFocus)
Quand on active un élément (quand on place le curseur dans un champ de formulaire, par click ou par tabulation, pour commencer la saisie par exemple).

blur (onBlur)
Quand un élément perd le focus (quitter un champ de formulaire après être activé par exemple).

keyDown (onKeyDown)
Quand une touche du clavier est enfoncée.

keyUp (onKeyUp)
Quand une touche du clavier est relâchée.

keyPress (onKeyPress)
Quand une touche du clavier est maintenue enfoncée.

load (onLoad)
Quand un élément est chargé par le navigateur. Elle peut être appliquée à la page entière

(balise **<body**>), dans ce cas l'événement se produira quand tous les éléments de la page seront chargés.

unLoad (onUnload)
Quand la page en cours est fermée ou quittée.

resize (onResize)
Quand l'internaute redimensionne la taille de la fenêtre du navigateur.

select (onSelect)
Quand l'internaute essaie de sélectionner le texte contenu dans l'objet accueillant l'événement.

change (onChange)
Quand l'internaute change le contenu d'un élément (liste de sélection ou zone de texte par exemple).
submit (onSubmit)
Quand l'internaute clique sur n'importe quel bouton de type submit présent dans la page (ou dans le formulaire).

mouseDown (onMouseDown)
Quand l'internaute appuie sur n'importe quel bouton de la souris.

mouseUp (onMouseUP)
Quand le bouton de la souris et relâché.

mouseMove (onMouseMove)
Quand l'internaute fait bouger le curseur de la souris dans la zone accueillant l'événement.
dragStart (onDragStart)
Se produit quand l'internaute commence le déplacement d'un élément par "glisser-déposer" (Drag & Drop).

dragEnter (onDragEnter)
Se produit quand l'internaute entre dans la zone où sera déposé l'élément glissé.
dragOver (onDragOver)
Se produit quand l'internaute survole la zone où sera déposé l'élément glissé (même si le concept est proche de celui de onDragEnter, ils sont des événements légèrement différents).
drop (onDrop)
Se produit quand l'internaute dépose l'élément glissé dans la zone prévue à cet effet.

Les Formulaires et Les Evénements

Parfois, vous rencontrerez une situation où vous voudrez arrêter un événement qui adopte son comportement par défaut. L'exemple le plus courant est celui d'un formulaire Web, par exemple un formulaire d'inscription personnalisé. Lorsque vous remplissez les détails et appuyez sur le bouton "Soumettre", le comportement naturel consiste à soumettre les données à une page spécifiée sur le serveur pour traitement, et le navigateur redirige vers une page de "message de réussite" quelconque (ou la même page, si une autre n'est pas spécifiée.).

Le problème survient lorsque l'utilisateur n'a pas soumis les données correctement. En tant que développeur, vous devez arrêter la soumission au serveur et lui envoyer un message d'erreur indiquant ce qui ne va pas et ce qui doit être fait pour corriger les erreurs.

Tout d'abord, un simple formulaire HTML qui vous oblige à entrer votre nom et votre prénom:

```
<form>
  <div>
    <label for="fname">First name: </label>
    <input id="fname" type="text">
  </div>
  <div>
    <label for="lname">Last name: </label>
    <input id="lname" type="text">
  </div>
  <div>
     <input id="submit" type="submit">
  </div>
</form>
<p></p>
```

Maintenant un peu de JavaScript - ici nous implémentons une vérification très simple dans un gestionnaire d'événement **onsubmit** (l'événement submit est renvoyé sur un formulaire quand il est soumis) qui vérifie si les champs de texte sont vides. Si c'est le cas, nous appelons la fonction **preventDefault()** sur l'objet événement - ce qui stoppe la soumission du formulaire - puis nous affichons un message d'erreur dans le paragraphe sous notre formulaire pour indiquer à l'utilisateur ce qui ne va pas :

```
var form = document.querySelector('form');
var fname = document.getElementById('fname');
var lname = document.getElementById('lname');
var submit = document.getElementById('submit');
var para = document.querySelector('p');
form.onsubmit = function(e) {
  if (fname.value === '' || lname.value === '') {
    e.preventDefault();
    para.textContent = 'You need to fill in both names!';
  }
}
```

Web Biographie :
https://developer.mozilla.org/fr/docs/Apprendre/JavaScript/Building_blocks/Ev%C3%A8nements
https://www.pierre-giraud.com/javascript-apprendre-coder-cours/addeventlistener-gestion-evenement/
https://www.chiny.me/les-evenements-en-javascript-6-7.php