ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

HONORIS UNITED UNIVERSITIES

Développement Web:

JavaScript

Professeur:

Nouhaila MOUSSAMMI

n.moussammi@emsi.ma

2024/2025

Gestion des événements

Gestion des événements

JS JavaScript

Qu'est-ce qu'un événement ?

Un événement est une action effectuée soit par l'utilisateur soit par le navigateur.

Il existe plusieurs types d'événements : événement de souris, un événement de clavier, un événement de document, etc.

Exemples d'événements DOM :

- Clic sur un bouton par l'utilisateur ;
- Déplacement de la souris ;
- Chargement de la page ;
- Clic sur une touche du clavier ;
- Soumissions d'un formulaire ;
- …

Javascript offre des mécanismes de réaction aux événements.

Les événements sont généralement traités par une fonction, qui s'exécute après que l'événement soit produit.

Gestion des événements

JS JavaScript

Définition et exemples

->Certaines actions sur des éléments d'un document web génèrent un événement. Un évènement caractérise l'action réalisée et dépend de l'élément cible (sur lequel porte l'action).

->En JavaScript, un événement est une action qui se déclenche et possède deux caractéristiques principales :

- On peut "écouter" cet événement, c'est-à-dire que le système nous notifie lorsqu'il survient, permettant ainsi de le détecter.
- On peut "réagir" à cet événement, c'est-à-dire qu'il est possible d'associer du code qui s'exécutera automatiquement lorsque l'événement se produit.

1

Écouteur d'événement (Event Listener) :

L'écouteur d'événement est un objet qui attend qu'un certain événement se produise (un clic, un mouvement de souris, etc.).

2

Gestionnaire d'événements :

Le gestionnaire d'événements correspond généralement à une fonction appelée suite à la production de l'événement.

Gestion des événements

JS JavaScript

La programmation événementielle

-> Pour écouter et répondre à un évènement, les gestionnaires d'évènements sont utilisés dans Javascript.

-> 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.

-> Il existe trois grandes façons d'implémenter un gestionnaire d'évènements :

- ❑ 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).

-> La programmation événementielle consiste à associer une fonction à l'occurrence d'un événement sur un élément.

-> La fonction est déclenchée (exécutée) lorsque l'événement se produit sur l'élément cible (target).

-> Fonction Listener : La fonction associée à un événement est appelée « gestionnaire d'événement » (event handler) ou « écouteur d'événement » (event listener).

Gestion des événements

La programmation événementielle

JS JavaScript

1. Utiliser les attributs HTML pour gérer un évènement :

L'utilisation d'attributs HTML pour prendre en charge un évènement est la méthode la plus ancienne à notre disposition.

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.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
        <script src='cours.js' async></script>
    </head>

    <body>
        <h1>Titre principal</h1>
        <p>Un premier paragraphe</p>
        <button onclick="alert('Bouton cliqué')">Cliquez moi !</button>
        <div onmouseover="this.style.backgroundColor='orange'"
             onmouseout="this.style.backgroundColor='white'">
            <p>Un paragraphe dans un div</p>
            <p>Un autre paragraphe dans le div</p>
        </div>
    </body>
</html>
```

Gestion des événements

La programmation événementielle

JS JavaScript

2. Utiliser les propriétés JavaScript pour gérer un évènement

Chaque évènement est représenté en JavaScript par un objet basé sur l'interface Event.

L'interface **Event** est l'interface de base commune pour tout évènement qui se produit dans le **DOM**. Certains types d'évènements sont ensuite définis plus précisément dans des interfaces qui héritent de Event.

Les gestionnaires d'évènements liés à ces évènements sont décrits dans le mixin GlobalEventHandlers qu'implémentent notamment les interfaces HTMLElement et Document.

Les gestionnaires d'événements sont des propriétés nommées avec le préfixe "on" suivi du nom de l'événement (ex. onclick, onmouseover). On associe généralement une fonction (souvent anonyme) à ces propriétés pour exécuter du code lorsqu'un événement est déclenché, et ces propriétés sont souvent utilisées sur des objets de type Element.

```
1
2   //On sélectionne le premier button et le premier div du document
3   let b1 = document.querySelector('button');
4   let d1 = document.querySelector('div');
5
6   //On utilise les propriétés gestionnaires d'évènement avec nos éléments
7   b1.onclick = function(){alert('Bouton cliqué')};
8   d1.onmouseover = function(){this.style.backgroundColor ='orange'};
9   d1.onmouseout = function(){this.style.backgroundColor='white'};
10
11
```

Gestion des événements

La programmation événementielle

JS JavaScript

3. Utiliser la méthode addEventListener() pour gérer un évènement:

Méthode addEventListener()

La méthode **addEventListener** appliquée sur un élément DOM, lui ajoute un gestionnaire pour un événement particulier.

Cette fonction sera appelée à chaque fois que l'événement se produit sur l'élément.

Remarque

- On peut ajouter plusieurs événements à un même élément.

Syntaxe : la méthode prend deux paramètres : le type de l'événement et la fonction qui gère l'événement.

element.**addEventListener**(événement, fonction de rappel)

Elément : un élément HTML auquel l'événement est attaché.

Événement : le nom de l'événement.

Fonction de rappel : la fonction qui va gérer l'événement.

Gestion des événements

La programmation événementielle

JS JavaScript

3. Utiliser la méthode addEventListener() pour gérer un évènement:

-> La méthode **addEventListener()** fait partie de l'interface **EventTarget** et est fréquemment utilisée avec des objets de type **Element**, puisque l'interface Element hérite de **Node**, qui à son tour hérite de **EventTarget**.
-> La méthode **addEventListener()** prend deux arguments :

- ❖ le nom de l'événement à gérer
- ❖ le code à exécuter (souvent sous forme de fonction) lorsque cet événement se déclenche.

-> Cette méthode permet également de réagir plusieurs fois de manière distincte à un même événement ou de gérer différents événements sur un ou plusieurs objets Element.

```
11
12  //On sélectionne le premier button et le premier div du document
13  let b1 = document.querySelector('button');
14  let d1 = document.querySelector('div');
15
16  //On utilise la méthode addEventListener pour gérer des événements
17  b1.addEventListener('click', function(){alert('Bouton cliqué')});
18  d1.addEventListener('mouseover', function(){this.style.backgroundColor ='orange'});
19  d1.addEventListener('mouseover', function(){this.style.fontWeight ='bold'});
20  d1.addEventListener('mouseout', function(){this.style.backgroundColor='white'});
21
```

Gestion des événements

La programmation événementielle

JS JavaScript

3. Utiliser la méthode addEventListener() pour gérer un évènement:

Exemple 1 :

```
let element = document.getElementById("btn");

element.addEventListener("click", message);

//Fonction qui gère l'événement
function message() {
  alert("Vous avez cliqué!")
}
```

Exemple 2 : utiliser une fonction interne dans la méthode **addEventListener**()

```
let element = document.getElementById("btn");
element.addEventListener("click", function () { alert("Vous avez cliqué!")});
```

Gestion des événements

La programmation événementielle

JS JavaScript

3. Utiliser la méthode addEventListener() pour gérer un évènement:

Événements multiples utilisant addEventListener()

La méthode **addEventListener()** permet d'ajouter plusieurs méthodes identiques ou différentes à un seul élément. Ainsi, il est possible d'ajouter plus de deux écouteurs d'événement pour le même événement.

Exemple :

```
let element = document.getElementById("btn");
element.addEventListener("click", fct1);
element.addEventListener("click", fct2);

function fct1() {
  alert("Fonction 1");
}

function fct2() {
  alert("Fonction 2");
}
```

Gestion des événements

La programmation événementielle

JS JavaScript

3. Utiliser la méthode addEventListener() pour gérer un évènement:

Événements multiples utilisant addEventListener()

La méthode **addEventListener()** permet aussi d'attacher plusieurs types d'événements **différents** au même élément HTML .

Exemple :

```
let element = document.getElementById("btn");
element.addEventListener("click", clickFct); //Evénement : clic de la souris
element.addEventListener("mouseover", mouseoverFxn); //Evénement : passage de la souris sur un élément
element.addEventListener("mouseout", mouseoutFxn); //Evénement : la souris quitte l'élément

function clickFct() {
  alert("Vous avez cliqué :");
}
function mouseoverFxn() {
  element.style.background = "red";
  element.style.padding = "8px";
}

function mouseoutFxn() {
  element.style.background = "white";
  element.style.padding = "2px";
}
}
```

Gestion des événements

La programmation événementielle

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().**

```
1
2   //On sélectionne le premier button et le premier div du document
3   let b1 = document.querySelector('button');
4   let d1 = document.querySelector('div');
5
6   function changeCouleur(){
7       this.style.backgroundColor ='orange';
8   }
9
10  //On utilise la méthode addEventListener pour gérer des événements
11  b1.addEventListener('click', function(){alert('Bouton cliqué')});
12  d1.addEventListener('mouseover', changeCouleur);
13  d1.addEventListener('mouseover', function(){this.style.fontWeight ='bold'});
14
15  //On supprime un événement
16  d1.removeEventListener('mouseover', changeCouleur);
17
```

Gestion des événements

La programmation événementielle

JS JavaScript

Supprimer l'écouteur d'événement

La méthode **removeEventListner**() permet de supprimer l'écouteur d'événement d'un élément ou un objet HTML.

Exemple :

```
<p class="style1">Cet élément possède l'événement "mouseover"</p>
<button id="btn" onclick="SupprimerEvnt()">Supprimer l'événement</button>

<script>
  let element = document.querySelector(".style1"); //Sélectionner l'élement button
  element.addEventListener("mouseover", fct1); // Ajouer un évenement de type « mouseover »

function fct1(){
    alert("Evénement déclenché!");
  }

  function SupprimerEvnt(){
    element.removeEventListener("mouseover", fct1);
  }
</script>
```

Gestion des événements

La programmation événementielle

JS JavaScript

Exercice 1 :

Créer la page Web correspondante au code suivant :

```
<!DOCTYPE html>
<html>
    <head>
    <title></title>
</head>
<body>
    <h1>Histoires</h1>
    <ul>
        <li>Nom : <input type="text" id="nom">
        <li>Adjectif: <input type="text" id="adj">
        <li>Nom d'une personne: <input type="text" id="pers">
    </ul>
    <button id="raconter">Raconte une histoire</button>
    <div id="histoire"></div>
</body>
</html>
```

1. Ajoutez une balise de script en bas de la page pour votre code.
2. Ajoutez un écouteur d'événement au bouton afin qu'il appelle une fonction « raconterHistoire » lorsqu'il est cliqué.
3. Dans la fonction « raconterHistoire », récupérez les valeurs des éléments d'entrée du formulaire, puis créez une histoire et affichez-la dans la div correspondante.

Gestion des événements

JS JavaScript

Solution (Exercice 1) :

```
<script>
        function raconterHistoire()
        {
            var histoireDiv = document.getElementById("histoire");
            var personne = document.getElementById("pers").value;
            var adjectif = document.getElementById("adj").value;
            var nom = document.getElementById("nom").value;
            histoireDiv.innerHTML = personne + " mange " + adjectif + " " + nom + "... bizarre !";
        }
        var btn = document.getElementById('raconter');
        btn.addEventListener('click', raconterHistoire);
    </script>
```

Gestion des événements

JS JavaScript

Exercice 2 : **Créer une liste de taches**

Objectif: Développer une application simple permettant d'ajouter et de supprimer des tâches d'une liste.

HTML

```
<input type="text" id="tache" placeholder="Entrez une tâche">
<button id="ajouterTache">Ajouter</button>
<ul id="listeTaches"></ul>
```

JavaScript

```javascript
const inputTache = document.getElementById('tache');
const boutonAjouter = document.getElementById('ajouterTache');
const listeTaches = document.getElementById('listeTaches');

boutonAjouter.addEventListener('click', () => {
    const nouvelleTache = inputTache.value;

    // Vérifier si l'utilisateur a entré une tâche
    if (nouvelleTache.trim() !== '') {
        // Créer un nouvel élément de liste
        const elementListe = document.createElement('li');
        elementListe.textContent = nouvelleTache;

        // Créer un bouton pour supprimer la tâche
        const boutonSupprimer = document.createElement('button');
        boutonSupprimer.textContent = 'Supprimer';

        // Ajouter un écouteur d'événement pour supprimer la tâche
        boutonSupprimer.addEventListener('click', () => {
            listeTaches.removeChild(elementListe);
        });

        // Ajouter le bouton de suppression à l'élément de liste
        elementListe.appendChild(boutonSupprimer);

        // Ajouter l'élément de liste à la liste
        listeTaches.appendChild(elementListe);

        // Effacer le champ de saisie
        inputTache.value = '';
    }
});
```

TP

JS JavaScript