Web Developer
Scripts Clients

10 – Les événements

A. Martel 2025-2026

10. Événements – Introduction

Un événement est un mécanisme permettant de détecter une action qui se déroule sur une page, plus précisément sur un élément d'une page.

Il en existe de plusieurs types : click, dblclick, mouseover, mouseout, mousedown, mouseup, mousemove, keydown, keyup, focus, blur, change, select, submit, reset, etc.

10. Événements – Niveaux

Il existe trois niveaux d'événements :

1) Dans la balise
2) Avec le DOM via une propriété (dit DOM-0)
3) Avec le DOM via la méthode addEventListener (dit DOM-2, à privilégier)

10. Événements – Balise

Il s'agit de préciser au travers d'un attribut de la balise quel événement est écouté et quel script sera exécuté :

```
<div class="bleu" onclick="rougir(this);">
  Contenu du bloc
</div>
…
function rougir(element) {
    element.className = "rouge";
}
```

10. Événements – DOM-0

Il s'agit de préciser au travers d'un attribut de l'élément auquel on accède en JS quel événement est écouté et quel script sera exécuté :

```
<div class="bleu" id="div_test">
  Contenu du bloc
</div>
…
const element = document.getElementById('div_test');

element.onclick = function () {
   this.className = "rouge";
};
```

Avantage : découplage HTML et JS

10. Événements – DOM-2

Il s'agit de préciser à l'aide de l'objet Event sur l'élément auquel on accède en JS quel événement est écouté et quel script sera exécuté, à l'aide de la méthode

addEventListener(nom_événement, fonction de callback[, capture]);

L'option capture (booléen) détermine si l'exécution de l'événement se fait dans la phase de capture (vers le bas dans l'arbre du DOM) ou de propagation (vers le haut dans l'arbre du DOM)

10. Événements – DOM-2

Exemple avec une fonction anonyme comme callback :

```
<div class="bleu" id="div_test">
  Contenu du bloc
</div>
...
const bloc = document.getElementById('div_test');

bloc.addEventListener('click', function () {
    this.className = "rouge";
}, false);
```

10. Événements – DOM-2

Ou avec une fonction nommée comme callback (à privilégier car permet de retirer l'écoute) :

```
<div class="bleu" id="div_test">
  Contenu du bloc
</div>
…
const bloc = document.getElementById('div_test');

function rougir(){
   this.className = "rouge";
}

bloc.addEventListener('click', rougir, false);
```

10. Événements – DOM-2

Avec cette solution, il est possible d'écouter plusieurs fois le même événement sur un même élément :

```
const bloc = document.getElementById('div_test');

function rougir(){
   this.className = 'rouge';
}
function blanchir(){
   this.style.color = 'white';
}

bloc.addEventListener('click', rougir, false);
bloc.addEventListener('click', blanchir, false);
```

10. Événements – DOM-2

Test différence capture/propagation → **capture** :

```
const mon_div = document.getElementById("mon_div");
const mon_paragraphe = document.getElementById("mon_paragraphe");

mon_div.addEventListener("click", function () {
   console.log("Vous avez cliqué sur le DIV");
}, true);
mon_paragraphe.addEventListener("click", function () {
   console.log("Vous avez cliqué sur le P");
}, true);
```

10. Événements – DOM-2

Test différence capture/propagation → **propagation** :

```
const mon_div = document.getElementById("mon_div");
const mon_paragraphe = document.getElementById("mon_paragraphe");

mon_div.addEventListener("click", function () {
   console.log("Vous avez cliqué sur le DIV");
}, false);
mon_paragraphe.addEventListener("click", function () {
   console.log("Vous avez cliqué sur le P");
}, false);
```

10. Événements – DOM-2

On peut arrêter cette propagation grâce à la méthode stopPropagation :

```
const mon_div = document.getElementById("mon_div");
const mon_paragraphe = document.getElementById("mon_paragraphe");

mon_div.addEventListener("click", function () {
   console.log("Vous avez cliqué sur le DIV");
}, false);
mon_paragraphe.addEventListener("click", function (e) {
   e.stopPropagation();
   console.log("Vous avez cliqué sur le P");
}, false);
```

10. Événements – DOM-2

Il est possible de retirer un événement via la méthode removeEventListener (si on utilise une fonction nommée) :

```
const bloc = document.getElementById('div_test');

function rouge(){
   this.className = "rouge";
}
function blanc(){
   this.style.color = "white";
}

bloc.addEventListener('click', rouge, false);
bloc.addEventListener('click', blanc, false);
bloc.removeEventListener('click', rouge, false);
```

10. Événements – Objet Event

L'objet Event nous fournit des informations lorsqu'un événement est capté.

Selon le type d'événement, un type plus spécifique est transmis :

- MouseEvent : clientX, clientY, etc.
- KeyboardEvent : key, code, etc.
- InputEvent : input, change, paste
- FocusEvent : focus, blur
- FormEvent : submit, reset
- PointerEvent
- TouchEvent : pour mobiles

10. Événements – Objet Event

Accès à l'objet Event :

```
const bloc = document.getElementById('div_test');

function test_event(e){
      console.log(e);
}
bloc.addEventListener('click', test_event, false);
```

10. Événements – Objet Event

Exemple

Suivi des coordonnées du pointeur de la souris :

```
const coord = document.getElementById('div_test');

function test_event(e) {
    let x = e.clientX;
    let y = e.clientY;
    coord.textContent = '(' + x + ',' + y + ')';
}

document.addEventListener('mousemove', test_event, false);
```

10. Événements – Délégation

La délégation d'événements est une technique qui permet d'écouter sur un seul élément au lieu de n fois sur ses n enfants.

Ça permet d'améliorer les performances.

```
// Sans délégation
const lis = document.querySelectorAll('li');

lis.forEach(element => {
  element.addEventListener('click', () => {
    console.log('CLICK sur un LI');
  })
});
```

10. Événements – Délégation

```
// Avec délégation
const ol = document.querySelector('#liste');

ol.addEventListener('click', (e) => {
  if (e.target.tagName === 'LI') { // Attention, en majuscules
    console.log('CLICK sur un LI');
  }
});
```

10. Événements – Bonnes pratiques

- Toujours utiliser addEventListener
- Préférer des fonctions nommées
- Ne jamais mélanger DOM-0 et DOM-2
- Privilégier la délégation sur les listes dynamiques, les tables, etc.

10. Événements – Exercices

Exercices

Vous devez implémenter :

1. Une fonctionnalité de repli ou d'expansion d'un bloc DIV lors d'un click sur son titre
2. Un champ de recherche qui s'allonge lorsqu'on le sélectionne et qui reprend sa taille initiale lorsqu'on en sort si le contenu est vide, qui garde sa taille sinon
3. Une fonctionnalité de drag and drop d'un DIV
4. Une fonctionnalité de dessin d'un rectangle

10. Événements – Exercices

Exercices

Vous devez implémenter :

5. Idem que l'exercice 4 mais avec une palette de 4 couleurs jaune, vert, bleu et rouge