

LE DOM HTML

(Partie 2)

Plan du cours

1-LES EVENEMENTS EN JAVASCRIPT

2-LA METHODE JAVASCRIPT

ADDEVENTLISTENER()

3-Test

1-LES EVENEMENTS EN JAVASCRIPT

1-Définition et découverte des événements

- Les évènements 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'évènement.
- Parfois, on va vouloir « attacher » une action spécifique à un évènement, comme par exemple modifier la taille des textes sur une page lorsque l'utilisateur clique sur un bouton.
- Pour faire cela, nous allons utiliser des attributs HTML de « type » évènement, et ensuite en JavaScript créer le code correspondant à l'action que l'on souhaite attacher à notre évènement.
- Les attributs HTML de type évènements sont nombreux. Nous allons passer en revue les plus utilisés d'entre eux dans cette partie.

1-LES EVENEMENTS EN JAVASCRIPT

1-Définition et découverte des événements

- Parmi ceux-ci, nous avons notamment :
- L'attribut **onclick** : se déclenche lorsque l'utilisateur clique sur un élément ;
- L'attribut **onmouseover** : se déclenche lorsque l'utilisateur passe le curseur de sa souris sur un élément ;
- L'attribut **onmouseout** : se déclenche lorsque l'utilisateur sort son curseur d'un élément ;
- L'attribut **onkeydown** : se déclenche lorsque l'utilisateur appuie sur une touche de son clavier sans la relâcher avec son curseur sur l'élément.

1-LES EVENEMENTS EN JAVASCRIPT

1-Définition et découverte des événements

- Voyons immédiatement un premier exemple. Imaginons que l'on souhaite afficher une boîte de dialogue de type `alert()` lorsqu'un utilisateur clique sur un paragraphe dans notre page web.
- Pour faire cela, on va utiliser évidemment `onclick` en HTML sur notre paragraphe puis on va créer notre `alert()` en JavaScript.

1-LES EVENEMENTS EN JAVASCRIPT

1-Définition et découverte des événements

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les évènements</title>
    <meta charset="utf-8">
  </head>

  <body>
    <h1 id="gros_titre">Les évènements</h1>

    <p onclick="alert('Bravo !');">Cliquez-moi, cliquez-moi !</p>
    <p>Un deuxième <strong>paragraphe</strong></p>
  </body>
</html>
```

1-LES EVENEMENTS EN JAVASCRIPT

1-Définition et découverte des événements

- Comme on peut le remarquer, le HTML nous autorise à placer du code JavaScript directement en valeur de l'attribut onclick.
- Ici, on procède de cette manière car l'exemple est très simple. Cependant, dans la suite, nous utiliserons le DOM qui nous offrira bien plus de possibilités.

1-LES EVENEMENTS EN JAVASCRIPT

2-L'utilisation du mot clef `this` dans la gestion d'évènements

- Pour rappel, en JavaScript, le mot clef `this` nous sert de référence à différents objets selon le contexte.
- Dans le contexte de la gestion d'évènements, `this` va faire référence à l'objet (représentant l'élément HTML) qui est le sujet de l'évènement.
- Par exemple, si on reprend l'exemple de notre paragraphe, ce paragraphe est le sujet de l'évènement car c'est lui qui possède l'attribut d'évènement et le code associé qui va déclencher une action lors du clic.
- En utilisant `this` dans notre code JavaScript dans ce cas, on va donc travailler sur l'élément `p` en soi.
- On peut ainsi par exemple très simplement changer le contenu de notre paragraphe lors du clic d'un utilisateur.

1-LES EVENEMENTS EN JAVASCRIPT

2-L'utilisation du mot clef this dans la gestion d'évènements

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les évènements</title>
    <meta charset="utf-8">
  </head>

  <body>
    <h1 id="gros_titre">Les évènements</h1>

    <p onclick="this.textContent='Merci !';">Cliquez-moi, cliquez-moi !</p>
    <p>Un deuxième <strong>paragraphe</strong></p>
  </body>
</html>
```

1-LES EVENEMENTS EN JAVASCRIPT

2-L'utilisation du mot clef this dans la gestion d'évènements

Ici, dès que vous cliquez sur le paragraphe, son contenu change et le texte « Merci ! » apparaît à la place du texte d'origine.

En pratique, cependant, nous utiliserons peu ce genre de code (car nous allons voir par la suite que nous sommes limités lorsque nous n'utilisons pas le DOM), mais il est bien de l'avoir vu au moins une fois pour bien le comprendre et comprendre ce qu'on va faire ensuite.

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM
- La gestion des évènements va devenir véritablement intéressante lorsqu'on va réagir aux évènements via le DOM HTML.
- Le DOM HTML va nous permettre d'assigner des gestionnaires d'évènements spécifiques à des éléments HTML en utilisant le JavaScript.
- Cette fois-ci, nous n'allons donc plus utiliser des attributs HTML mais du code JavaScript à proprement parler pour construire nos évènements.
- Pour réagir aux évènements avec le JavaScript : on peut utiliser la méthode `addEventListener()`.

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM

Afin que vous voyiez tout de même comment on procède avec cette méthode, nous allons voir un exemple rapide avec la propriété onclick. Notez que les propriétés possèdent souvent des noms analogues aux attributs HTML (l'attribut HTML onclick devient la propriété JavaScript onclick par exemple).

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les évènements</title>
    <meta charset="utf-8">
  </head>

  <body>
    <h1 id="gros_titre">Les évènements</h1>

    <p>Cliquez-moi, cliquez-moi !</p>
    <p>Un deuxième <strong>paragraphe</strong></p>
    <script>
      //On accède à notre premier paragraphe
      var p1 = document.querySelector('p');

      /*Création d'un gestionnaire d'évènements pour
      *l'évènement "onclick"*/
      p1.onclick = function(){
        this.innerHTML = '<strong>Bravo !</strong>';
        this.style.color = 'orange';
      };
    </script>
  </body>
</html>
```

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM
- Comme vous pouvez le constater, on commence à réutiliser des outils que nous avons déjà vus précédemment, comme les fonctions anonymes par exemple.
- C'est en effet maintenant que vous possédez de bonnes connaissances en JavaScript que celui-ci va se révéler véritablement intéressant, donc restez attentif !
- Dans l'exemple ci-dessus, on commence par accéder à notre premier paragraphe puis on lui attache un évènement de type onclick.

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM
- Vous pouvez remarquer que nous attachons ensuite une fonction anonyme à notre évènement. Ici, nous utilisons une fonction anonyme afin que celle-ci ne s'exécute pas immédiatement.
- En effet, rappelez vous que pour exécuter une fonction anonyme, nous devions jusqu'à présent soit la transformer en fonction auto invoquée, soit l'enfermer dans une variable puis appeler cette variable avec un couple de parenthèses.
- Ici, nous voyons une troisième façon de faire : l'assigner à un évènement.

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM
- Dans l'exemple ci-dessus, notre fonction va en effet s'exécuter dès que l'évènement sera déclenché, c'est-à-dire dès que l'utilisateur cliquera sur notre paragraphe.
- Si nous avons exécuté la fonction immédiatement, seule la valeur retournée par celle-ci aurait été attachée à notre évènement, ce qui n'est pas du tout le comportement souhaité.
- Notez que nous aurions tout aussi bien pu créer une fonction avec un nom puis n'assigner que le nom de la fonction à notre évènement (sans les parenthèses qui feraient qu'elle s'exécuterait immédiatement).
- Ainsi, le code ci-dessous est équivalent à celui ci-dessus :

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les évènements</title>
    <meta charset="utf-8">
  </head>

  <body>
    <h1 id="gros_titre">Les évènements</h1>

    <p>Cliquez-moi, cliquez-moi !</p>
    <p>Un deuxième <strong>paragraphe</strong></p>
    <script>
      //On accède à notre premier paragraphe
      var p1 = document.querySelector('p');

      /*Création d'un gestionnaire d'évènements pour
      *l'évènement "onclick"*/
      p1.onclick = bravo;

      function bravo(){
        this.innerHTML = '<strong>Bravo !</strong>';
        this.style.color = 'orange';
      };
    </script>
  </body>
</html>
```

1-LES EVENEMENTS EN JAVASCRIPT

- 3-Les évènements et le DOM
- Une nouvelle fois, il est important que vous compreniez bien ces exemples et façons de faire car de nombreux développeurs codent encore comme cela.
- Cependant, je vous déconseille d'utiliser vous même ce genre d'écriture. A la place, nous préférons utiliser la méthode `addEventListener()` qui est beaucoup moins limité et que nous allons étudier dans la prochaine partie.

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 1-Présentation de la méthode JavaScript `addEventListener()`
- La méthode `addEventListener()` va nous permettre de lier du code à un évènement. On parlera alors de gestionnaire d'évènements.
- Le code sera alors exécuté dès le déclenchement de l'évènement.
- Cette méthode appartient à l'objet `Element` et va avoir besoin de deux arguments pour fonctionner : le nom de l'évènement déclencheur de l'action et le code relatif à l'action à effectuer.
- Les événements vont une nouvelle fois avoir des noms similaires aux attributs HTML mais ne vont plus être précédés du « on » (par exemple, `onclick` devient `click`).
- Voyons immédiatement comment fonctionne cette nouvelle méthode à travers des exemples simples.

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 1-Présentation de la méthode JavaScript
addEventListener()

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Cliquez-moi, cliquez-moi !</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
  <script>
    //On accède à notre premier paragraphe
    var p1 = document.querySelector('p');

    //On accroche un gestionnaire d'évènements à p1
    p1.addEventListener('click',changeTexte);

    /*On construit notre fonction changeTexte qui ne sera
    *exécutée que lors du déclenchement de l'évènement*/
    function changeTexte(){
      this.innerHTML = '<strong>Bravo !</strong>';
      this.style.color = 'orange';
    };
  </script>
</body>
```

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 1-Présentation de la méthode JavaScript `addEventListener()`
- Rien de bien compliqué ici, on se contente de faire la même chose que dans la partie précédente mais en utilisant cette fois-ci `addEventListener()`.
- On passe donc un nom d'évènement en premier argument de la méthode `addEventListener()` puis le nom d'une fonction à exécuter en second argument.
- J'attire ici votre attention sur le fait qu'on ne précise pas l'habituel couple de parenthèses après notre fonction ici afin que celle-ci ne s'exécute pas immédiatement mais seulement après le déclenchement de l'évènement (le clic sur notre paragraphe).

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 1-Présentation de la méthode JavaScript `addEventListener()`
- Finalement, on crée notre fonction en soi qui ici modifie le contenu et la couleur du texte à l'intérieur de notre paragraphe après qu'un utilisateur ait cliqué dessus.
- On aurait également tout à fait pu utiliser une fonction anonyme en second argument de notre méthode `addEventListener()`, cependant cela aurait été beaucoup moins clair et lisible.
- Regardez plutôt l'exemple ci-dessous qui produit exactement le même résultat que précédemment pour vous en convaincre.

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 1-Présentation de la méthode JavaScript
addEventListener()

```
<!DOCTYPE html>
<html>
  <head>
    <title>Les évènements</title>
    <meta charset="utf-8">
  </head>

  <body>
    <h1 id="gros_titre">Les évènements</h1>

    <p>Cliquez-moi, cliquez-moi !</p>
    <p>Un deuxième <strong>paragraphe</strong></p>
    <script>
      //On accède à notre premier paragraphe
      var p1 = document.querySelector('p');

      //Utilisation d'une fonction anonyme
      p1.addEventListener('click',function(){
        this.innerHTML = '<strong>Bravo !</strong>';
        this.style.color = 'orange';});
    </script>
  </body>
</html>
```

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode `addEventListener()` ?
- L'un des grands avantages de la méthode `addEventListener()` est de pouvoir lier plusieurs gestionnaires d'évènements de même type sur un élément HTML.
- Par exemple, on va pouvoir afficher différents messages suite à un clic, ce qui aurait été impossible sans `addEventListener()`.

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode `addEventListener()` ?

Dans l'exemple ci-dessus, dès qu'un utilisateur clique sur notre paragraphe, les deux fonctions liées à notre événement vont être exécutées à la suite et donc deux boîtes de dialogue vont apparaître l'une après l'autre.

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Cliquez-moi, cliquez-moi !</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
</body>
<script>
  //On accède à notre premier paragraphe
  var p1 = document.querySelector('p');

  //On accroche deux gestionnaires d'évènements à p1
  p1.addEventListener('click',Message1);
  p1.addEventListener('click',Message2);

  //Création des fonctions
  function Message1(){
    alert('Première boîte !');
  };

  function Message2(){
    alert('Deuxième boîte !');
  };
</script>
```

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode `addEventListener()` ?
- La méthode `addEventListener()` va également nous permettre de lier plusieurs événements différents à un même élément HTML.
- Par exemple, on va pouvoir exécuter un code lorsqu'un utilisateur déplace le curseur de sa souris sur un élément et un autre code lorsqu'il reste cliqué dessus.
- Pour faire cela, nous allons utiliser respectivement les deux événements `mouseover` et `mousedown`.

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode
addEventListener() ?

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Passez sur moi svp</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
  <script>
    //On accède à notre premier paragraphe
    var p1 = document.querySelector('p');

    //On accroche deux gestionnaires d'évènements à p1
    p1.addEventListener('mouseover',Fonction1);
    p1.addEventListener('mousedown',Fonction2);

    //Création des fonctions
    function Fonction1(){
      this.innerHTML = 'Cliquez moi maintenant !';
      this.style.backgroundColor = 'orange';
    };

    function Fonction2(){
      this.innerHTML = 'Bravo !';
      this.style.color = '#26C';
      this.style.fontWeight = 'bold';
      this.style.fontSize= '24px';
    };
  </script>
</body>
```

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode `addEventListener()` ?
- Rien de nouveau ici : on crée nos deux gestionnaires d'évènements avec une fonction pour chacun qui ne s'exécutera que lors du déclenchement de l'évènement choisi.
- Si vous avez essayé ce code, cependant, vous remarquerez qu'une fois qu'on a passé notre souris sur le paragraphe ou une fois celui-ci cliqué, il garde les propriétés liées à l'évènement en question même lorsqu'on relâche le clic de la souris ou qu'on fait sortir le curseur de la souris.
- Vous remarquerez également que si vous rentrez à nouveau dans l'élément ensuite le texte change à nouveau.

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode `addEventListener()` ?
- Cela est tout à fait normal puisque vous déclenchez à nouveau l'évènement `mouseover`.
- Si vous souhaitez rendre son comportement d'origine à notre paragraphe une fois les évènements déclenchés, dans cette situation, vous devrez utiliser également les évènements `mouseout` et `mouseup`.
- L'évènement `mouseout` correspond à l'évènement « l'utilisateur déplace le curseur de sa souris en dehors d'un élément » tandis que `mouseup` se déclenche lorsque « l'utilisateur relâche le clic ».
- Voyons en pratique comment on va s'y prendre :

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode
addEventListener() ?

```
<body>
  <h1 id="gros_titre">Les évènements</h1>

  <p>Passez sur moi svp</p>
  <p>Un deuxième <strong>paragraphe</strong></p>
  <script>
    //On accède à notre premier paragraphe
    var p1 = document.querySelector('p');

    p1.addEventListener('mouseover',Fonction1);
    p1.addEventListener('mouseout',Reset1);
    p1.addEventListener('mousedown',Fonction2);
    p1.addEventListener('mouseup',Reset2);
```

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode
addEventListener() ?

```
function Fonction1(){
    this.innerHTML = 'Cliquez moi maintenant !';
    this.style.backgroundColor = 'orange';
};
function Reset1(){
    this.innerHTML = 'Passez sur moi svp';
    this.style.backgroundColor = '';
};

function Fonction2(){
    this.innerHTML = 'Bravo !';
    this.style.color = '#26C';
    this.style.fontWeight = 'bold';
    this.style.fontSize= '24px';
};
function Reset2(){
    this.innerHTML = 'Passez sur moi svp';
    this.style.color = '';
    this.style.fontWeight = '';
    this.style.fontSize= '';
};
</script>
```

2-LA METHODE JAVASCRIPT ADDEVENTLISTENER()

- 2-Pourquoi utiliser la méthode `addEventListener()` ?
- Dans cet exemple, nous avons attaché des fonctions « reset » dont le rôle est de rendre les valeurs et l'aspect d'origine à notre paragraphe lorsque les évènements `mouseout` et `mouseup` se produisent.
- Comme les évènements `mouseout` et `mouseup` sont les évènements inverses de `mouseover` et `mousedown`, au final, le paragraphe reprend sa forme initiale dès qu'on relâche le clic de souris ou dès que l'on déplace notre curseur en dehors.

3-Test

Question n°1 :

Qu'est-ce que le DOM ?

Afficher la réponse

Le DOM est une interface de programmation qui va nous permettre de manipuler du code HTML en JavaScript.

3-Test

Question n°2 :

Le DOM considère tout ce qui se trouve dans une page HTML comme un...

- 1. document
- 2. élément
- 3. noeud

Afficher la réponse

Réponse 3. Le navigateur va créer un DOM à partir d'une page HTML automatiquement. Dans cette structure, tout ce qui se trouve dans notre page est considéré comme un noeud.

3-Test

Question n°3 :

Laquelle de ces méthodes permet d'accéder à un élément HTML en le ciblant avec un sélecteur CSS ?

1. `getElementById()`
2. `getElementsByTagName()`
3. `querySelector()`

Afficher la réponse

Réponse 3. La méthode `querySelector()` nous permet de cibler un élément HTML en précisant un sélecteur CSS.

3-Test

Question n°4 :

Si je souhaite récupérer le contenu textuel d'un élément, j'utiliserai plutôt...

1. La propriété innerHTML
2. La propriété textContent
3. La propriété body

Afficher la réponse

Réponse 2. La propriété innerHTML va récupérer tout le contenu d'un élément HTML, à la différence de textContent qui ne va récupérer que le contenu textuel.

3-Test

Question n°5 :

La méthode getElementById() est une méthode de l'objet...

- 1. Element
- 2. Document

Afficher la réponse

Réponse 2. La méthode getElementById() est une méthode de l'objet Document.

3-Test

Question n°6 :

Les propriétés innerHTML et textContent sont des propriétés de l'objet...

- 1. Element
- 2. Document

Afficher la réponse

Les propriétés innerHTML et textContent sont des propriétés de l'objet Element. Il faudra donc les appliquer à des objets de type element.

3-Test

Question n°7 :

Pour modifier les styles CSS d'un élément en JavaScript, on utilise généralement...

1. La méthode `setAttribute()`
2. La propriété `style`
3. La méthode `style()`

Afficher la réponse

Réponse 2. On utilisera la propriété `style` suivie du nom de la propriété CSS à modifier / ajouter.

3-Test

Question n°8 :

Citez deux méthodes nous permettant d'insérer du texte ou des éléments HTML dans une page

Afficher la réponse

On peut utiliser les méthodes `appendChild()` et `insertBefore()`.

3-Test

Question n°9 :

Pour créer un gestionnaire d'évènements, il est généralement recommandé d'utiliser...

1. Les attributs HTML relatifs aux évènements
2. Les propriétés JavaScript relatives aux évènements
3. La méthode `addEventListener()`

Afficher la réponse

Réponse 3. Il est recommandé d'utiliser la méthode `addEventListener()` qui nous offre beaucoup plus de possibilités pour gérer nos évènements.