Apprendre Javascript | Gestion des événements

Comme vu dans les cours précédents, les événements sont des actions qui se produisent dans le navigateur. Par exemple, un clic de souris, un appui sur une touche du clavier, le chargement d'une page, etc.

Nous allons approfondir ce sujet en apprenant à gérer les événements en détails.

Fonction `addEventListener`

Pour gérer un événement, on utilise la méthode `addEventListener` qui permet d'attacher un gestionnaire d'événements à un élément. La fonction prend en paramètre le nom de l'événement à écouter et une fonction qui sera exécutée lorsque l'événement se produira.

Le nom de l'événement est toujours en minuscules, même s'il comprend plusieurs mots. Ex: `click`, `mouseover`, `keydown`, etc.

```
element.addEventListener("click", function (evenement) {
    console.log("Clic !", evenement);
});
```

Les types d'événements

Il existe de nombreux types d'événements que vous pouvez écouter.

- Les événements de souris : `click`, `mouseover`, `mouseout`, `mousedown`, `mouseup`, `mousemove`.
- Les événements de clavier : `keydown`, `keyup`, `keypress`.
- Les événements de formulaire : `submit`, `change`, `focus`, `blur`, `invalid`.
- Les événements de fenêtre : `load`, `resize`, `scroll`.
- Les événements de changement de l'URL : `hashchange`, `popstate`.
- Les événements de changement d'orientation : `orientationchange`.
- Les événements de média : `play`, `pause`, `ended`.
- Les événements de toucher pour appareils mobiles: `touchstart`, `touchmove`, `touchend`, `touchcancel`.
- Les événements d'animation : `animationstart`, `animationend`, `animationiteration`,`transitionend`, `transitionrun`, `transitionstart`.
- Les événements du clipboard : `copy`, `cut`, `paste`.

Liste complète des événements

https://developer.mozilla.org/fr/docs/Web/API/Event#interfaces_bas%C3%A9es_sur_event

L'objet `event`

Lorsqu'un événement se produit, un objet de type `event` est automatiquement créé et passé en argument à la fonction de gestion de l'événement. Cet objet contient des informations sur l'événement qui s'est produit.

```
element.addEventListener("click", function (evenement) {
    console.dir(evenement);
});
```

Les propriétés/méthodes de l'objet `event`

Les propriétés varient en fonction du type d'événement. Voici quelques propriétés courantes :

- `type` : le type de l'événement.
- `currentTarget` : l'élément sur lequel l'événement est attaché.
- `timeStamp` : l'heure à laquelle l'événement s'est produit.
- `clientX` et `clientY` : les coordonnées du curseur de la souris par rapport à la fenêtre.
- `pageX` et `pageY` : les coordonnées du curseur de la souris par rapport à la page.
- `keyCode` : le code de la touche du clavier qui a été pressée.
- `key` : la touche du clavier qui a été pressée.
- `button` : le bouton de la souris qui a été cliqué.
- `buttons` : les boutons de la souris qui ont été cliqués.
- `shiftKey`, `ctrlKey`, `altKey` : indique si les touches `Shift`, `Ctrl`ou `Alt` ont été enfoncées lors de l'événement.
- `preventDefault()` : méthode qui permet d'annuler le comportement par défaut de l'événement.

Bloquer le comportement par défaut d'un événement

Par défaut, certains événements ont un comportement par défaut. Par exemple, un clic sur un lien redirige vers une autre page. Pour empêcher ce comportement par défaut, vous pouvez utiliser la méthode `preventDefault` de l'objet `event`. Un autre exemple est le formulaire qui se soumet automatiquement lorsqu'on appuie sur la touche `Enter`.

```
element.addEventListener("click", function (evenement) {
    evenement.preventDefault();
});

form.addEventListener("submit", function (evenement) {
    evenement.preventDefault();
});
```

Ajouter des paramètres supplémentaires à un gestionnaire d'événement

Il y a deux façons d'ajouter des paramètres supplémentaires à un événement. La première utilise une fonction anonyme. La deuxième méthode "bind" est plus propre et plus lisible et sera vue en détail dans au cours 16

Utiliser une fonction anonyme

Pour rappel, une fonction anonyme est une fonction sans nom.

La première façon est d'utiliser une fonction anonyme pour appeler la fonction de gestionnaire d'événement avec les paramètres supplémentaires.

```
element.addEventListener("click", function (evenement) {
    maFonction(param1, param2);
});
```