JavaScript - DOM

m2iformation.fr

Le DOM

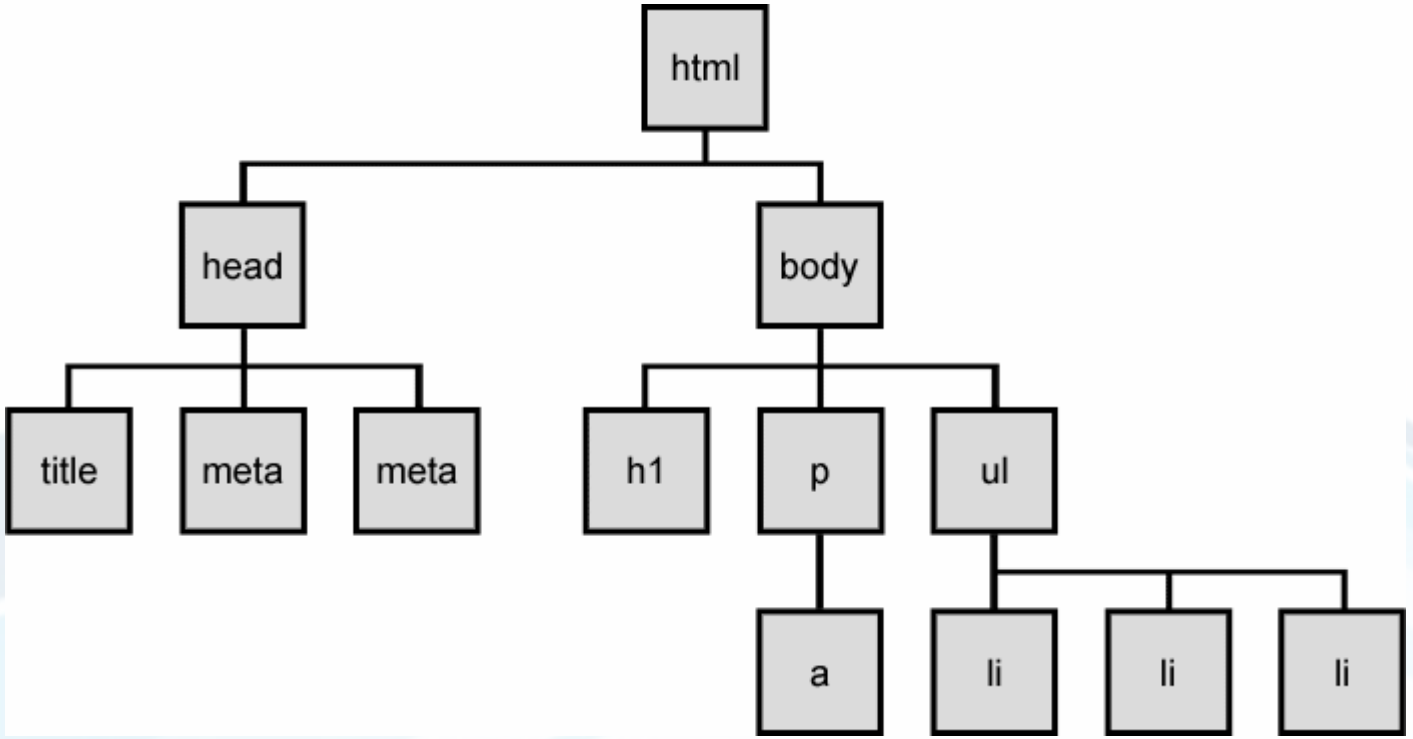
Le Document Object Model

Dans une page internet, il existe ce que l'on appelle le DOM (Document Object Model). Ce DOM est en réalité une hierarchie sous forme d'arbre qui comprend l'ensemble de notre HTML de façon à obtenir les liens entre les parents et les enfants. Commençant au niveau du super-parent Document, on descend ensuite à chaque fois qu'on rencontre une série de balises dans la structure de notre HTML, pour atteindre petit à petit les enfants et leurs contenu potentiel :

Pour le Javascript, cette structure est très avantageuse, car elle correspond à ce que l'on pourrait trouver dans des objets.

Pour atteindre un élément, on peut via le Javascript et l'API du Web utiliser les méthodes de la classe « **document** », telles que :

- **.getElementById()**
- **.getElementsByClassName()**
- **.querySelector()**
- **.querySelectorAll()**



Sélectionner et Manipuler l'HTML

Si l'on veut sélectionner l'HTML, il y a plusieurs méthodes, comme l'utilisation de **.getElementById()** ou de **.getElementsByClassName()**

```
let monHeader = document.getElementById("my-heading");
let mesListItems = document.getElementsByClassName("list-group-items");
```

Malgré tout, la solution la plus simple est sans doute d'utiliser la méthode du .**querySelector()** (et sa version multiple **.querySelectorAll()**). Pour ce faire, il suffit d'avoir recourt à la même façon de sélectionner les éléments que lorsque l'on fait du CSS. C'est-à-dire que la sélection d'une Id et la sélection d'éléments par leur classe commune via des syntaxes de ce type :

```
let monHeader2 = document.querySelector("#my-heading");
let mesListItems2 = document.querySelectorAll(".list-group-items");
```

Une fois les éléments HTML sélectionnés, il est assez facile d'en extraire leur contenu ou de le modifier. Pour ce faire, on passe par les propriétés de l'élément, comme on le ferait avec un objet classique :

```
console.log(monHeader.textContent);
monHeader.textContent = "Mon nouveau titre de site web !";
```

Les Events Listeners

Notre page HTML est capable de créer des évènements que le Javascript peut écouter. Par exemple, le clic d'un bouton, ou le survol d'un input, l'appui sur une touche du clavier ou la modification d'une valeur de <textarea>, tout ceci correspond à un évènement. En écoutant l'évènement que l'on souhaite suivre, on peut ensuit réagir à son déclenchement.

Pour créer un Event Listener, il n'y a rien de plus simple. Il suffit de sélectionner un élément HTML et d'exécuter sur lui la méthode **.addEventListener()** Dans cette méthode, il va falloir passer en premier le type de l'évènement sous la forme d'une **string,** puis une fonction qui sera exécutée lors de l'évènement (Baptisée l'Event Handler). Il est fréquent que l'on passe des fonction fléchées dans cette partie, de sorte à obtenir à la fin ce genre d'instruction :

```
let myButton = document.querySelector("#my-button");

myButton.addEventListener('click', () => {
    console.log(document.querySelector("#mon-input").textContent);
});
```

Manipuler le Style

Il est possible que l'on veuille modifier le style de notre page HTML via le Javascript, par exemple pour changer la couleur de fond d'une image ou d'une pour réagir à des changements effectués par l'utilisateur de notre site web. Pour ce faire, on peut utiliser tout d'abord la sélection d'un élément par les méthodes vues précédemment. Puis, il nous faut accéder à sa propriété **style**. Enfin, il nous est possible de modifier chaque propriété CSS via une syntaxe différente. En CSS, nous aurions par exemple eu une syntaxe de ce genre :

```css
#my-heading {
    background-color: red;
}
```

Cependant, en Javascript, les espaces dans les noms des propriétés CSS se traduisent en notation **camel case** de la sorte :

```javascript
document.querySelector("#my-heading").style.backgroundColor = "blue";
```

Si l'on veut modifier plusieurs styles à la fois, une syntaxe différente est possible. On peut passer dans la propriété **style.cssText** une string de la sorte :

```javascript
document.querySelector("body").style.cssText =
    `
    background-color: red;
    color: white;
    `
```

Manipuler les Classes

En plus de pouvoir modifier le style de nos éléments, le Javascript peut également facilement modifier les classes que portent nos différentes balises. En effet, chaque élément du DOM possède une propriété nommée **classList** qui contient l'ensemble des classes dont dispose l'élément HTML. Cet ensemble peut ensuite aisément se voir retirer ou ajouter des nouveaux éléments via les méthodes ci-dessous :

- **classList.remove(<valeur>) :** Cette méthode va tout simplement retirer la classe de l'élément HTML si elle se trouve parmi la liste des classes.
- **classList.add(<valeur>) :** Cette méthode va ajouter quant à elle une classe à l'élément HTML, lui permettant par exemple de subit ainsi des modification par le CSS.

```
const closeButton = document.querySelector("#close-btn");
const openButton = document.querySelector("#open-btn");

closeButton.addEventListener('click', () => {
    document.querySelector("#my-window").classList.add("hidden");
});

openButton.addEventListener('click', () => {
    document.querySelector("#my-window").classList.remove("hidden");
});
```

Gérer les touches du Clavier

Si l'on veut pouvoir proposer une interactivité entre notre site web et le clavier de l'utilisateur, il nous est possible de le faire via trois évènement que l'on peut retrouver au niveau du document lui-même :

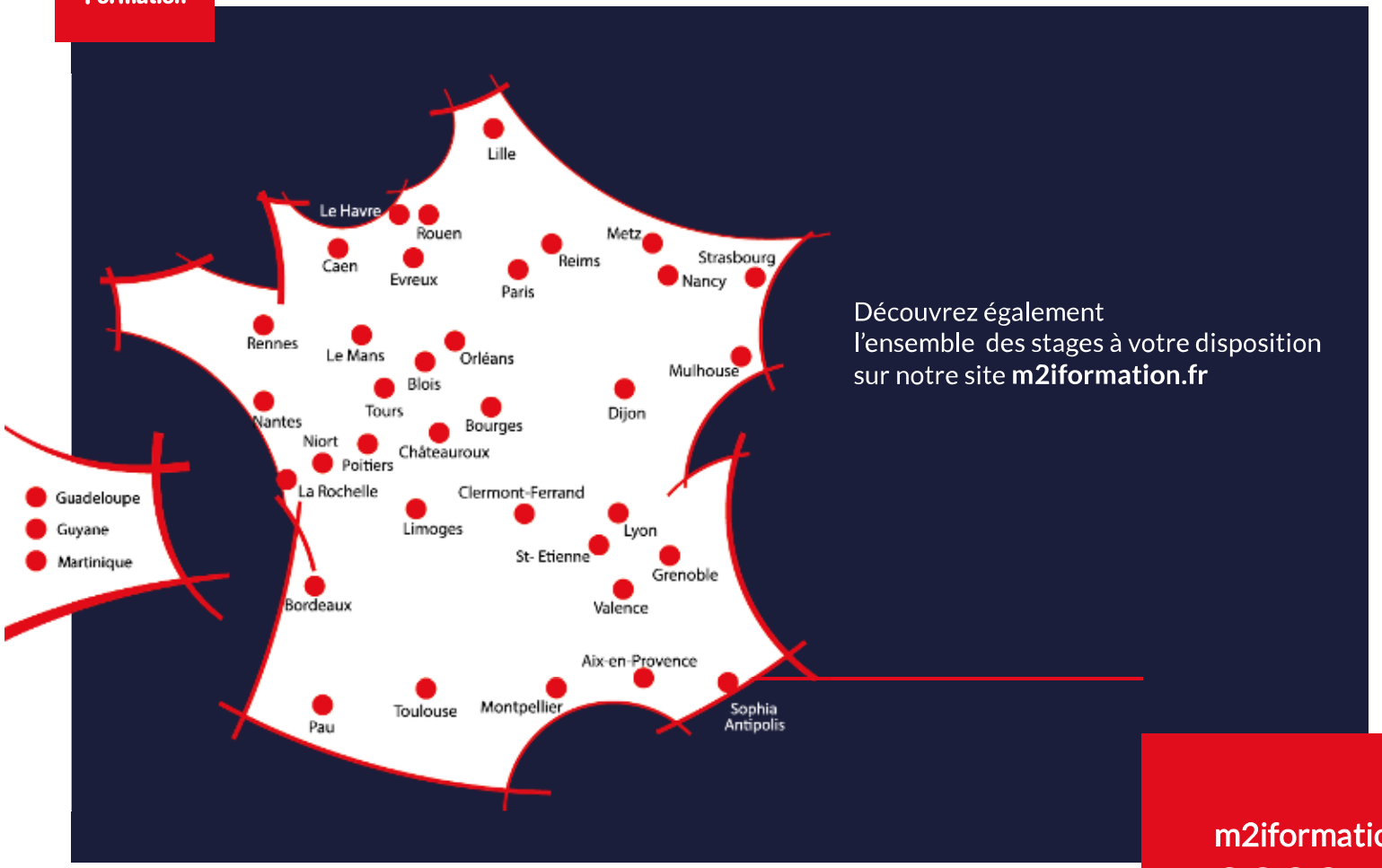
- **keydown** : Lorsque l'utilisateur appuie sur une touche de son clavier
- **keypress**: Lorsque l'utilisateur laisse son doigt sur une touche de son clavier
- **keyup** : Lorsque l'utilisateur relâche la touche de son clavier

```
document.addEventListener('keydown', () => {
    console.log("Une touche a été pressée !");
});
```

Une fois l'évènement écouté via l'ajout d'un « event listener » au niveau du document via l'instruction ci-dessus. Cet évènement sera généré pour n'importe laquelle des touches du clavier. Il nous est cependant possible de modifier la chose de façon à récupérer l'évènement et de ne sélectionner que les touches qui nous intéressent. Pour cela, il faut ajouter un paramètre à l'event listener (le paramètre sera ici automatiquement l'évènement). Un évènement de ce type possède une propriété **.key**, qu'il nous est possible d'utiliser dans le cadre d'une condition pour obtenir une fonction plus aboutie de la sorte :

```
document.addEventListener('keydown', (event) => {
    if (event.key === "Escape") {
        document.querySelector("#my-window").classList.add("hidden");
    }
});
```

Formation



Découvrez également
l'ensemble des stages à votre disposition
sur notre site **m2iformation.fr**

m2iformation.fr