

# Javascript - DOM

Le DOM, Document Object Model correspond à l'ensemble des objets définis de manière arborescente générés par le navigateur depuis l'interprétation d'un document HTML.

Le DOM fourni une API permettant sa manipulation, à partir de Javascript.

Les éléments du DOM sont des objets, disposant de propriétés, de méthodes et peuvent être associés à des évènements.

## document & accès aux objets

La variable **document** est l'objet racine généré par un document HTML. Elle permet d'accéder à l'ensemble des objets instanciés dans une page.

Exemple d'accès à un élément par son **id**,  
et modification de son contenu grace à la propriété **innerHTML** :

```
<html>
<body>
  <p id="demo"></p>
  <script>
    document.getElementById('demo').innerHTML = 'Hello World!';
  </script>
</body>
</html>
```

La propriété **innerHTML** est accessible sur tous les éléments d'une page (y compris son body), et permet d'affecter un contenu HTML à l'élément.

### Accès par id

Un id est unique dans une page Web : le meilleur moyen d'accéder à un élément précis de la page est de le faire par son id :

Accès à l'élément d'id **intro** :

```
const element=document.getElementById('intro');
```

```
<h1>Introduction</h1>
<div id="intro">
  <p>Lorem ipsum</p>
</div>
<p>Lorem ipsum</p>
```

## Accès par tagName

Le tagName correspond au type d'un élément défini par sa balise ouvrante (p, div, a, span...).

Recherche des éléments de type p :

```
const elements=document.getElementsByTagName('p');
```

```
<h1>Introduction</h1>
<div id="intro">
  <p>Lorem ipsum</p>
</div>
<p>Lorem ipsum</p>
```

## Accès par Classe CSS

La classe CSS permet bien évidemment d'attribuer des styles CSS, mais offre également la possibilité de sélectionner des éléments.

Recherche des éléments de la classe **myClass** :

```
const elements=document.getElementsByClassName('myClass');
```

```
<h1>Introduction</h1>
<div id="intro" class="myClass">
  <p>Lorem ipsum</p>
</div>
<p class="para myClass">Lorem ipsum</p>
```

## Accès par sélecteur CSS

Les sélecteurs CSS permettent également de sélectionner des éléments.

Recherche des éléments de type **p** contenus dans un élément de classe **myClass** :

```
const elements=document.querySelectorAll('.myClass p');
```



**querySelectorAll()** retourne la collection d'éléments sélectionnés, tandis que **querySelector()** retourne le premier élément correspondant.

```
<h1>Introduction</h1>
```

```
<div id="intro" class="myClass">
  <p>Lorem ipsum</p>
</div>
<p class="para myClass">Lorem ipsum</p>
```

## Collections existantes

Certains objets ou collections sont accessibles depuis l'instance de **document** :

Objet/collection	Description
<b>document.forms</b>	Formulaires du document
<b>document.links</b>	Liens du document
<b>document.scripts</b>	Scripts du document
<b>document.images</b>	Images du document
<b>document.body</b>	Body du document

## Parcours d'une collection

Il est possible de parcourir une collection d'objets retournée, pour réaliser une opération sur chacun de ses éléments, en faisant un **forEach** :

Paragraphe en rouge :

```
document.querySelectorAll("p").forEach(function (p) {
  p.style.color = "red";
});
```

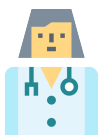
## Attributs

Il est possible d'accéder aux attributs d'un élément du DOM, en lecture pour connaître sa valeur, ou en écriture pour la modifier.

## Lecture

```

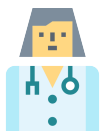
<p id="src-image"></p>
<script>
document.querySelector("#src-image").innerHTML=document.getElementById("img1").src;
</script>
```



## Écriture

```

<script>
document.getElementById("img2").src="https://cdn-icons-png.flaticon.com/512/1214/1214338.png";
document.getElementById("img2").title='Updated user image'
</script>
```



## Modification du DOM

### innerHTML

Tous les éléments du DOM ont une propriété **innerHTML** accessible en lecture ou en écriture.

```
<p id="para1">Lorem <strong>ipsum</strong></p>
<textarea id="para1-content"></textarea>
<script>
document.getElementById('para1-content').value=document.getElementById('para1').innerHTML;
</script>
```

Lorem **ipsum**

L'utilisation de **innerHTML** en lecture est à utiliser avec précaution, étant donné que le contenu HTML ne peut pas être vérifié et peut produire un DOM incohérent.

```
<div id="myDiv">
  <p></p>
  <p></p>
</div>
<script>
document.querySelectorAll("#myDiv p").forEach(function(p){
  p.innerHTML='Hello <strong>world!</strong>';
});
</script>
```

1

2

## write

**document.write()** permet d'écrire sur la page, à l'endroit où la méthode est appelée.

```
<script>
document.write('<p>Hello <strong>world</strong>!</p>');
</script>
```

## createElement & appendChild

La méthode **createElement** crée un élément et le retourne, tandis que **appendChild** permet de le placer dans le DOM.

```
<p id="append">Ancien contenu...</p>
<script>
const h3=document.createElement("h3");
h3.innerHTML="Hello world!";
document.getElementById('append').appendChild(h3);
</script>
```

Ancien contenu...

## Evènements

### Listener

La méthode **addEventListener()** attache un gestionnaire d'événements à l'élément spécifié.

```
<button id="bt-alert">Click me!</button>
<script>
document.getElementById('bt-alert').addEventListener("click", function(){
alert("Hello World!"); });
</script>
```

Click me!

Il est possible d'ajouter plusieurs gestionnaires sur le même évènement d'un élément.

## Suppression de Listener

```
<button id="bt-alert-2">Click me...</button>
<script>
const message=function(){ alert('Hello World!'); };
const bt=document.getElementById('bt-alert-2');
bt.addEventListener('click', message);
bt.removeEventListener('click', message);
</script>
```

Click me...

From:

<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:

<http://slamwiki2.kobject.net/web/js/dom>

Last update: **2023/03/31 03:06**

