



# Javascript



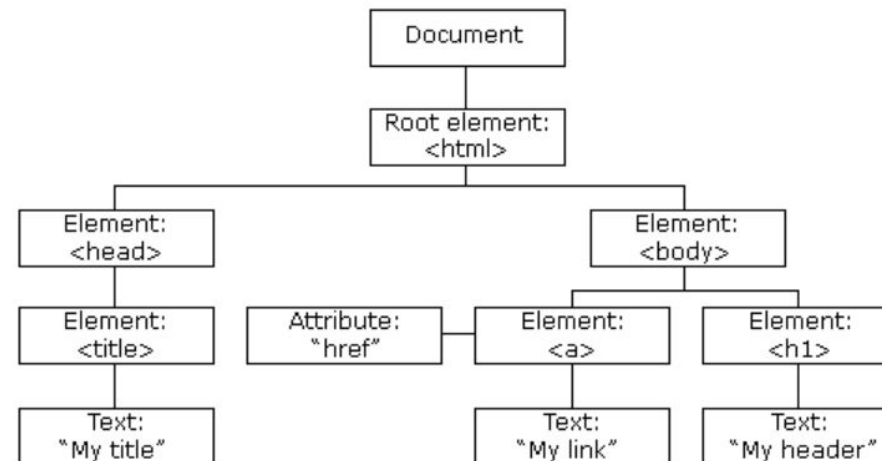
# JAVASCRIPT

## Javascript HTML DOM

Quand une page web est chargée, le navigateur crée un **M**odèle d'**O**bjets **D**ocument (DOM : Document Object Model)

Grâce à ce modèle, Javascript peut accéder à des éléments HTML de la page Web et les modifier.

The HTML DOM Tree of Objects



# Javascript

Avec le modèle d'objet, JavaScript obtient toute la puissance nécessaire pour créer du HTML dynamique :

- JavaScript peut modifier tous les éléments HTML de la page
- JavaScript peut modifier tous les attributs HTML de la page
- JavaScript peut modifier tous les styles CSS de la page
- JavaScript peut supprimer des éléments et attributs HTML existants
- JavaScript peut ajouter de nouveaux éléments et attributs HTML
- JavaScript peut réagir à tous les événements HTML existants dans la page
- JavaScript peut créer de nouveaux événements HTML dans la page

# JAVASCRIPT

## Exemple

```
<body>
  <h1>Utilisation d'une fonction</h1>
  <script>
    multiplication(5, 6);
    // Déclaration de notre fonction multiplication.
    function multiplication(p1, p2)
    {
      // Affichage du message dans notre page Web.
      document.write("Le résultat de " + p1 + " * " + p2 + " = " + (p1 * p2));
    }
  </script>
</body>
```

## Utilisation d'une fonction

Le résultat de  $8 * 7 = 56$

# Javascript

## Comment accéder à des éléments dans le document HTML ?

Il est possible d'accéder à un élément d'une page Web en utilisant différentes méthodes. Six méthodes de base existent pour cela :

- ***getElementById()*** : retourne une référence à l'élément dont l'identifiant correspond à la valeur recherchée.
- ***getElementsByName()*** : retourne un tableau d'éléments dont le nom correspond à la valeur recherchée.
- ***getElementsByTagName()*** : retourne un tableau d'éléments dont le tag correspond à celui recherché.
- ***getElementsByClassName()*** : retourne un tableau d'éléments dont le nom de la classe correspond à celui recherché.

Les deux prochaines méthodes, qui sont des méthodes de l'objet ***document***, nous permettent d'accéder à des éléments correspondant à un sélecteur spécifié. Ce sélecteur peut être une classe, un id, un type d'élément, un attribut ou autre.

- ***querySelector()*** : retourne des informations sur le premier élément trouvé correspondant au sélecteur sélectionné.
- ***querySelectorAll()*** : retourne des informations sous forme de tableau sur tous les éléments trouvés correspondant au sélecteur sélectionné.

# getElementById

```
const element = document.getElementById("Titre");  
console.log(element);  
console.log(element.attributes);
```

Et si l'objet n'existe pas ?

## getElementsByTagName

```
const elements = document.getElementsByTagName("p");  
console.log(elements);  
console.log(typeof elements);  
console.log(elements.length);
```

## getElementsByClassName

```
const elements = document.getElementsByClassName("class1");  
console.log(elements);  
console.log(typeof elements);  
console.log(elements.length);
```



# getElementsByTagName

```
const elements = document.getElementsByTagName("prenom");  
console.log(elements);  
console.log(typeof elements);  
console.log(elements.length);
```

# querySelector

```
const paragraphElement = document.querySelector("p");
```

```
const classElement = document.querySelector(".class");
```

```
const idElement = document.querySelector("#id");
```

```
const classParagraphElement = document.querySelector("p.class");
```

```
const strongParagraphElement = document.querySelector("p strong");
```

```
const titleParagraphElement = document.querySelector("h1+p");
```

# querySelectorAll

```
const paragraphElements = document.querySelectorAll("p");  
const classElements = document.querySelectorAll(".class");  
const idElements = document.querySelectorAll("#id");  
const classParagraphElements = document.querySelectorAll("p.class");  
const strongParagraphElements = document.querySelectorAll("p strong");  
const titleParagraphElements = document.querySelectorAll("h1+p");
```

Et si l'objet n'existe pas ?

# HTML DOM

Une **NodeList** et une **HTMLCollection** sont très similaires.

Les deux sont des collections semblables à des tableaux (listes) de nœuds (éléments) extraits d'un document. Les nœuds peuvent être accédés par des numéros d'index. L'index commence à 0.

Les deux possèdent une propriété **length** qui renvoie le nombre d'éléments dans la liste (collection).

- Une **HTMLCollection** est une collection d'éléments du document.
- Une **NodeList** est une collection de nœuds du document (nœuds d'élément, nœuds d'attribut et nœuds de texte).

## Accès aux éléments :

- Les éléments d'une **HTMLCollection** peuvent être accédés par leur nom, leur **id** ou leur numéro d'index.
- Les éléments d'une **NodeList** ne peuvent être accédés que par leur numéro d'index.

# HTML DOM

## Collection dynamique vs statique :

- Une **HTMLCollection** est toujours une collection **dynamique**. Exemple : si vous ajoutez un élément `<li>` à une liste dans le DOM, la liste dans l'HTMLCollection sera également mise à jour.
- Une **NodeList** est le plus souvent une collection **statique**. Exemple : si vous ajoutez un élément `<li>` à une liste dans le DOM, la liste dans la NodeList ne sera pas mise à jour.

## Méthodes de retour :

- Les méthodes `getElementsByClassName()` et `getElementsByTagName()` renvoient une HTMLCollection dynamique.
- La méthode `querySelectorAll()` renvoie une NodeList statique.
- La propriété `childNodes` renvoie une NodeList dynamique.

## Autres éléments du document

document.body	Retourne l'élément <body>
document.forms	Retourne tous les éléments <form>
document.images	Retourne tous les éléments <img>
document.links	Retourne tous les éléments <a> qui ont un attribut href
document.title	Retourne l'élément <title>

# JAVASCRIPT

## Accéder au contenu HTML avec JavaScript

Nous allons débiter avec la propriété *innerHTML*. Examinez le prochain bloc de code.

```
<body>
  <h1 id="Titre">Découverte du DOM HTML</h1>
  <p class="Para">Ceci est un premier paragraphe contenant
  |   |   |   |   <a href="http://monsite.com">un lien fictif</a>.</p>
  <p class="Para">Ceci est un deuxième paragraphe.</p>

  <script>
    let para = document.querySelector('.Para').innerHTML;
    let message = "Contenu récupéré avec innerHTML :\n" + para;
    alert(message);
  </script>
</body>
```

Cette page indique

Ceci est un premier paragraphe contenant <a href="http://monsite.com">un lien fictif</a>.

OK

# JAVASCRIPT

## Suite *innerHTML*

Dans ce premier exemple, la propriété *innerHTML* retourne tout le contenu de notre paragraphe incluant les informations de notre élément *a* contenu dans ce dernier. Si nous ne désirons pas obtenir les informations de cet élément mais seulement que le contenu textuel de notre paragraphe, c'est à dire ce qui est affiché sur notre page, nous allons devoir utiliser une autre propriété, la propriété *textContent*. Modifions notre code afin de voir la différence qui existe dans l'utilisation de ces deux propriétés. Examinez le prochain bloc de code.

```
<body>
  <h1 id="Titre">Découverte du DOM HTML</h1>
  <p class="Para">Ceci est un premier paragraphe contenant
    <a href="http://monsite.com">un lien fictif</a>.</p>
  <p class="Para">Ceci est un deuxième paragraphe.</p>

  <script>
    let para = document.querySelector('.Para').innerHTML;
    let text = document.querySelector('.Para').textContent;
    let message = "Contenu récupéré avec innerHTML :\n" + para;
    message += "\n\n Contenu récupéré avec textContent :\n" + text;
    alert(message);
  </script>
</body>
```

Cette page indique

Contenu récupéré avec *innerHTML* :

Ceci est un premier paragraphe contenant <a href="http://monsite.com">un lien fictif</a>.

Contenu récupéré avec *textContent* :

Ceci est un premier paragraphe contenant un lien fictif.

OK



# InnerHTML et textContent

## Découverte du DOM HTML

Ceci est un premier paragraphe contenant [un lien fictif](#).

Ceci est un deuxième paragraphe contenant [un autre lien](#).

Ajoutons maintenant un script dans notre page qui modifiera le texte contenu dans notre élément h1 à l'aide de la propriété innerHTML de notre élément. Examinez le script suivant ajouté à la suite de notre bloc de code composant notre page Web.

```
<script>
  let titre = document.getElementById('Titre')
  titre.innerHTML = "Titre modifié !";
</script>
```

## Titre modifié !

Ceci est un premier paragraphe contenant [un lien fictif](#).

Ceci est un deuxième paragraphe contenant [un autre lien](#).

# InnerHTML et textContent

## *Autres exemples*

```
<script>  
    document.getElementById('Titre').textContent += ", et ses fonctionnalités.";   
</script>
```

## **Le DOM HTML, ses fonctionnalités.**

Ceci est un premier paragraphe contenant [un lien fictif](#).

Ceci est un deuxième paragraphe contenant [un autre lien](#).

# InnerHTML et textContent

```
<script>
  document.getElementById('Titre').innerHTML +=
    ", <mark>et ses fonctionnalités</mark>.";
</script>
```

## Le DOM HTML, ses fonctionnalités.

Ceci est un premier paragraphe contenant [un lien fictif](#).

Ceci est un deuxième paragraphe contenant [un autre lien](#).

## Modification des attributs

**Accéder une valeur :** `element.getAttribute(attribute)` **ou** `element.attribute`

**Modifier une valeur :** `element.setAttribute(attribute, value)` **ou** `element.attribute = nouvelle valeur`

```
<script>
    document.querySelector('a').href = "http://wikipedia.org";
</script>
```

```
<script>
    document.querySelector('img').src = "Images/th.png";
</script>
```

# Modification des attributs

```
const idElement = document.getElementById("Titre");  
idElement.textContent = "Nouveau titre";
```

```
const ulElement = document.getElementsByTagName("ul");  
ulElement[0].setAttribute("type", "circle");
```

# Modification des styles CSS

```
<body>
  <h1 id="Titre">Le DOM HTML</h1>
  <p class="Para">Ceci est un premier paragraphe contenant
    <a href="http://monsite.com">un lien fictif</a>.</p>
  <p class="Para">Ceci est un deuxième paragraphe.
    <a href="http://monsite2.com">un autre lien</a>.</p>

  <script>
    document.body.style.backgroundColor = "yellow";
  </script>
</body>
```

# Modification des styles CSS

```
<body>
  <h1 id="Titre">Le DOM HTML</h1>
  <p class="Para">Ceci est un premier paragraphe contenant
    <a href="http://monsite.com">un lien fictif</a>.</p>
  <p class="Para">Ceci est un deuxième paragraphe.
    <a href="http://monsite2.com">un autre lien</a>.</p>

  <script>
    document.body.style.backgroundImage = "url(Images/reunion.png)";
  </script>
</body>
```



# Modification des styles CSS

## *Autres exemples*

Nous allons maintenant modifier la couleur du texte de nos deux paragraphes ainsi que la grosseur des caractères de chacun d'eux. Nous désirons que le premier paragraphe soit de couleur *navy* et le deuxième paragraphe soit de couleur *orange* et nous affecterons une grosseur de caractère de *20 pixels* à chacun de nos deux paragraphes ce qui représente environ deux fois la grosseur actuelle des caractères. Examinez le prochain script.

```
<script>
  let para = document.querySelectorAll(".Para");
  for(i = 0; i < para.length; i++)
  {
    if(i % 2 == 0)
    {
      para[i].style.color = "navy";
    }
    else
    {
      para[i].style.color = "orange";
    }
    para[i].style.fontSize = "20px";
  }
</script>
```

## Le DOM HTML

Ceci est un premier paragraphe contenant [un lien fictif](#).

Ceci est un deuxième paragraphe contenant [un autre lien](#).

Dans ce script, nous récupérons tous les éléments dont la valeur de l'attribut *class* est *Para* à l'intérieur d'un tableau que nous avons nommé *para* puis, en utilisant une boucle, nous assignons au premier élément de notre tableau la couleur de texte *navy* et au deuxième élément la couleur *orange* en plus de modifier la grosseur de caractères pour les afficher en *20 pixels*. Encore une fois, l'attribut *fontSize* ne s'écrit pas de la même manière en JavaScript que dans un fichier de style. Si nous rafraichissons notre navigateur, nous obtiendrons l'affichage de la prochaine figure à l'écran.



# Attributs pour modifier ou accéder une valeur

- Attributes
- Style
- Value
- textContent
- innerHTML
- nextSibling
- previousSibling
- Parent
- firstChild
- lastChild
- children

## Évaluez vos connaissances nouvellement acquises

2. Parmi les affirmations suivantes, laquelle est vraie concernant les propriétés innerHTML et textContent ?
  - a. La propriété innerHTML retourne seulement que le contenu texte d'un élément HTML et de tous ses descendants.
  - b. La propriété textContent retourne seulement que le contenu texte d'un élément HTML et de tous ses descendants.
  - c. Nous utilisons la propriété textContent seulement que lorsque l'on veut modifier le contenu texte d'un élément.
  - d. Aucune de ces réponses
3. Que retourne la propriété *links* de l'objet *document* ?
  - a. La valeur de l'attribut href de tous les liens contenus dans une page Web.
  - b. La valeur textuelle de tous les liens contenus dans une page Web.
  - c. Une collection de tous les liens contenus dans une page Web sous forme de tableau.
  - d. Toutes les informations du premier lien contenu dans une page Web.

## Évaluez vos connaissances nouvellement acquises

4. Examinez attentivement le prochain bloc de code.

```
<body>
  <h1 id="Titre">Le DOM HTML</h1>
  <p>Ceci est un premier paragraphe contenant
    <a href="http://monsite.com">un lien fictif</a>.</p>
  <p>Ceci est un deuxième paragraphe contenant
    <a href="http://monsite2.com">un autre lien</a>.</p>

  <script>
    let x = document.getElementById("Titre");
    x.className = "Gros_titre";
  </script>
</body>
```

Quel sera le résultat de l'exécution de ce script ?

- a. Un attribut class sera ajouté à l'élément h1 de la page Web.
- b. Tous les éléments de la page Web seront configurés avec un nouvel attribut class.
- c. Rien, il est impossible de modifier la valeur d'un attribut class pour un élément qui n'en contient pas.
- d. Aucune de ces réponses

## *Évaluez vos connaissances nouvellement acquises*

5. Examinez attentivement le prochain bloc de code.

```
<body>
  <h1 id="Titre">Le DOM HTML</h1>
  <p>Ceci est un premier paragraphe contenant
    <a href="http://monsite.com">un lien fictif</a>.</p>
  <p>Ceci est un deuxième paragraphe contenant
    <a href="http://monsite2.com">un autre lien</a>.</p>

  <script>
    let x = document.getElementById("Titre");
    x.style.fontSize = "60px";
  </script>
</body>
```

Quel sera le résultat de l'exécution de ce script ?

- a. La grosseur des caractères de tous les éléments de la page Web sera modifié.
- b. Rien, la propriété `fontSize` n'est pas une propriété de l'objet `style` de JavaScript.
- c. La grosseur des caractères de l'élément `h1` de la page Web sera modifié.
- d. Aucune de ces réponses

# Ajouter et supprimer des éléments

## Méthode

## Description

`document.createElement(element)`

Crée un élément HTML

`document.removeChild(element)`

Supprime un élément HTML

`document.appendChild(element)`

Ajoute un élément HTML

`document.replaceChild(new, old)`

Remplace un élément HTML

`document.write(text)`

Écrire dans le flux de sortie HTML

# Exercices

- Inputs avec nom, prénom et âge + bouton qui affiche le texte “Bonjour ! Je m’appelle PRENOM NOM et j’ai AGE ans.”
- Bouton cacher ou montrer une image d’étoile.

# Exercices

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <title>Exercice DOM</title>
  <style>
    .highlight {
      background-color: yellow;
    }
  </style>
</head>
<body>
  <h1 id="titre-principal">Bonjour, Monde !</h1>
```

```
<p class="description">Ceci est un paragraphe descriptif.</p>
<p class="description">Un autre paragraphe descriptif.</p>
<ul id="ma-liste">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
<div id="conteneur">
  <p id="texte">Texte initial</p>
</div>

<script src="script.js"></script>
</body>
</html>
```

# Exercices

Créez un fichier JavaScript nommé `script.js` et réalisez les tâches suivantes :

- Changer le texte du titre (`h1`) en "Bienvenue dans le monde du DOM !".
- Ajouter une classe `.highlight` aux paragraphes avec la classe `description`.
- Modifier le contenu du 3ème élément de la liste (`li`) pour qu'il affiche "Item Modifié".
- Insérer un nouvel élément `<li>` avec le texte "Nouvel Item" à la fin de la liste.
- Changer le texte du paragraphe avec l'ID `texte` pour qu'il affiche "Texte mis à jour".
- Changer la couleur de fond du conteneur avec l'ID `conteneur` en `lightblue`.