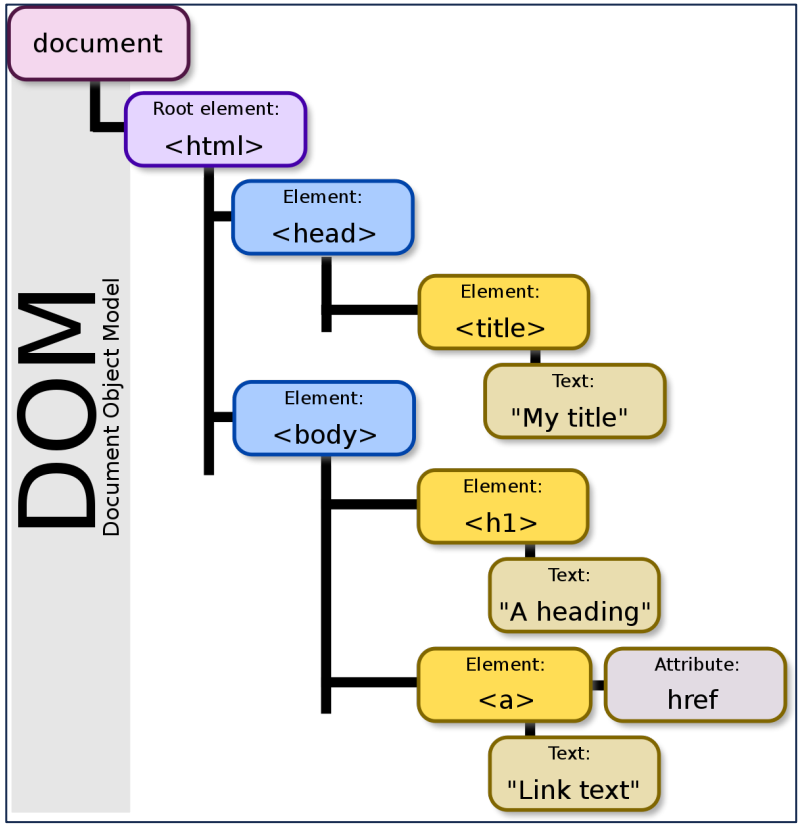
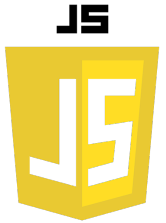
LE DOM (DOCUMENT OBJECT MODEL)




```
<!DOCTYPE html>
<html>
<head>
<title>My title</title>
<meta charset="utf-8">
</head>
<body>
<h1>A heading</h1>
<a href="http://newtonformationsnir.fr">Visit Schools</a>
</body>
</html>
```

Le terme "nœud" est un terme générique qui sert à désigner tous les objets contenus dans le DOM. A l'extrémité de chaque branche du DOM se trouve un nœud.
A partir du nœud HTML, 2 branches se forment : une première qui va aboutir au nœud HEAD et une deuxième qui aboutit à un nœud BODY.
De nouvelles branches se créent ensuite à partir des nœuds HEAD et BODY.

LE DOM (DOCUMENT OBJECT MODEL)



Manipulation du DOM :

La structure du DOM est dynamique. Cela signifie que vous pouvez utiliser JavaScript pour ajouter, modifier ou supprimer des nœuds, ce qui se reflète ensuite dans la page web affichée par le navigateur. Par exemple, vous pouvez utiliser JavaScript pour ajouter un nouvel élément à la page, modifier le texte d'un élément existant, changer le style CSS d'un élément, ou même supprimer un élément entièrement.

Importance du DOM :

Le DOM est un outil essentiel pour le développement web interactif. Sans lui, les pages web seraient essentiellement statiques. Avec le DOM, les développeurs peuvent créer des pages web qui répondent aux actions des utilisateurs, changent au fil du temps, et peuvent même être complètement reconfigurées à la volée par du code JavaScript.

ACCÉDER AU DOM

Accéder au Document Object Model (DOM) a plusieurs utilités importantes dans le développement web. Voici quelques-unes des principales raisons pour lesquelles vous voudriez interagir avec le DOM :

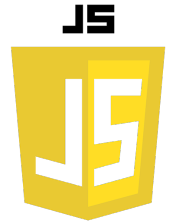
- Manipulation de contenu : Une fois que vous avez accès à un élément du DOM, vous pouvez obtenir ou modifier son contenu. Par exemple, vous pouvez changer le texte dans un élément <p>, ou ajouter une nouvelle image à une .

- Modification des attributs et des classes : Avec le DOM, vous pouvez lire ou changer les attributs d'un élément, tels que l'URL dans un lien <a href="">, ou ajouter/supprimer des classes pour changer les styles appliqués à un élément.

- Modification du style : Vous pouvez modifier les styles CSS d'un élément directement via le DOM, ce qui permet d'ajuster l'apparence d'une page web en fonction des actions de l'utilisateur ou d'autres événements.

- Création et suppression d'éléments : Le DOM vous permet de créer de nouveaux nœuds et de les ajouter à la page, ou de supprimer des nœuds existants. C'est la base de nombreuses techniques dynamiques de manipulation de pages, comme l'ajout, la modification ou la suppression de lignes dans un tableau.

ACCÉDER AU DOM



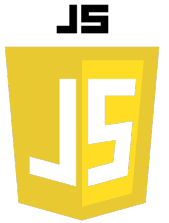
Dans JavaScript, vous avez plusieurs moyens d'**accéder** aux éléments du DOM, c'est-à-dire de sélectionner un ou plusieurs nœuds à partir de leur nom de balise, de leur classe, de leur identifiant ou même en fonction de leur relation avec d'autres nœuds. Voici quelques exemples :

document.getElementById(id)
C'est probablement la méthode la plus connue pour accéder à un élément du DOM. Elle retourne l'élément qui a l'attribut ID correspondant, ou null s'il n'y a pas d'élément avec cet ID. Par exemple :

```
let titleElement = document.getElementById('title’);
```

Dans cet exemple, titleElement est une référence à l'élément du DOM qui a l'ID "title". Vous pouvez maintenant utiliser cet élément pour lire ou modifier ses propriétés.

ACCÉDER AU DOM



document.getElementsByClassName(name)
Cette méthode retourne une HTMLCollection de tous les éléments qui ont une certaine classe. Par exemple :

```
let highlightedElements = document.getElementsByClassName('highlight');
```

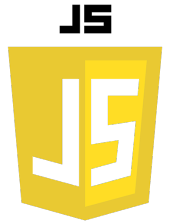
Dans cet exemple, highlightedElements est une collection des éléments qui ont la classe "highlight". Vous pouvez parcourir cette collection comme un tableau pour accéder à chaque élément.

document.getElementsByTagName(name)
Cette méthode retourne une HTMLCollection de tous les éléments qui ont un certain nom de balise. Par exemple :

```
let paragraphElements = document.getElementsByTagName('p');
```

Dans cet exemple, paragraphElements est une collection des éléments qui sont des paragraphes (<p>).

ACCÉDER AU DOM



document.querySelector(selector) et document.querySelectorAll(selector) :
Ces méthodes permettent de sélectionner des éléments en utilisant n'importe quel sélecteur CSS, ce qui les rend très puissantes et flexibles. querySelector retourne le premier élément qui correspond au sélecteur, tandis que querySelectorAll retourne une NodeList de tous les éléments qui correspondent.

Par exemple :

```
let firstHighlightedElement = document.querySelector('.highlight’);
let allHighlightedElements = document.querySelectorAll('.highlight’);
```

Dans cet exemple, firstHighlightedElement est le premier élément qui a la classe "highlight", et allHighlightedElements est une liste de tous ces éléments.

MODIFIER LE DOM

Une fois que vous avez accès à un élément du DOM, vous pouvez modifier son contenu en utilisant diverses propriétés et méthodes disponibles. Voici quelques-unes des plus couramment utilisées :

innerHTML : Cette propriété vous permet d'obtenir ou de définir le contenu HTML à l'intérieur d'un élément. Par exemple, si vous avez une référence à un élément de paragraphe, vous pouvez changer son contenu texte ainsi :

```
let p = document.getElementById('monParagraphe');
p.innerHTML = 'Ceci est un nouveau texte pour le paragraphe.';
```

Cette modification va changer le contenu de l'élément avec l'ID monParagraphe sur la page web.

MODIFIER LE DOM

textContent : Cette propriété vous permet d'obtenir ou de définir le contenu textuel d'un élément. Contrairement à innerHTML, elle n'interprète pas les chaînes comme du HTML. Par exemple :

```
let p = document.getElementById('monParagraphe');
p.textContent = 'Ceci est un <em>nouveau</em> texte pour le paragraphe.';
```

Avec textContent, les balises HTML incluses dans la chaîne seront affichées telles quelles et non interprétées comme du HTML.

innerText : Semblable à textContent, mais innerText prend en compte la visibilité du texte. Par exemple, si le texte est caché par CSS (display: none ou visibility: hidden), innerText ne l'inclura pas, tandis que textContent le fera.

AJOUTER DES ÉLÉMENTS AU DOM

Pour ajouter des éléments au DOM, vous devez d'abord créer l'élément, puis l'attacher à un élément existant dans le DOM. Voici comment vous pouvez le faire en JavaScript :

Créer un nouvel élément : Vous pouvez utiliser la méthode document.createElement() pour créer un nouvel élément. Par exemple :

```
let newElement = document.createElement('div'); //créera un nouvel élément div.
```

Ajouter le nouvel élément à un élément existant : Vous pouvez utiliser la méthode appendChild() pour ajouter le nouvel élément à un élément existant. Par exemple :

```
document.body.appendChild(newElement); //ajoutera le nouvel élément div à la fin de l'élément body du document.
```

Définir le contenu ou les attributs du nouvel élément : Vous pouvez utiliser les propriétés et méthodes comme textContent, innerHTML, et setAttribute() pour définir le contenu ou les attributs du nouvel élément. Par exemple :

```
newElement.textContent = 'Bonjour le monde !'; //définira le contenu textuel du nouvel élément div.
```

SUPPRIMER DES ÉLÉMENTS AU DOM

Pour supprimer des éléments du DOM, vous devez d'abord obtenir une référence à l'élément, puis utiliser la méthode removeChild() ou remove() sur l'élément parent ou sur l'élément lui-même, respectivement.

Obtenir une référence à l'élément à supprimer : Vous pouvez utiliser des méthodes comme document.getElementById(), document.getElementsByClassName(), ou document.querySelector() pour obtenir une référence à l'élément à supprimer. Par exemple :

```
let elementASupprimer = document.getElementById('id-de-lelement'); //obtiendra une référence à l'élément avec l'ID 'id-de-lelement'.
```

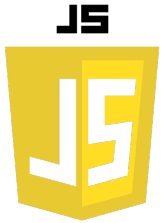
Supprimer l'élément : Vous pouvez utiliser la méthode removeChild() sur l'élément parent pour supprimer l'élément. Par exemple :

```
elementASupprimer.parentNode.removeChild(elementASupprimer); //supprimera l'élément du DOM.
```

Alternativement, si le navigateur le supporte, vous pouvez simplement utiliser :

```
elementASupprimer.remove(); //pour supprimer l'élément.
```

MODIFIER LES STYLES CSS



En JavaScript, vous pouvez utiliser l'objet style pour lire ou modifier les styles en ligne d'un élément. Chaque élément du DOM a une propriété style qui est un objet représentant les styles en ligne de l'élément.

Voici comment vous pouvez utiliser la propriété style pour modifier le style d'un élément :

```
// Obtenez une référence à l'élément
let element = document.getElementById('monElement');

// Modifiez le style de l'élément
element.style.color = 'blue';
element.style.backgroundColor = 'yellow';
element.style.border = '1px solid black';
```