



مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle
et de la Promotion du Travail

Module 5 : PROGRAMMER EN JAVASCRIPT

Chapitre 3 : Manipuler Les Eléments D'une Page Avec Dom

Réalisé par :
Mme ZROURI Amira

Plan du chapitre

1. Comprendre l'arbre DOM, les nœuds parents et enfants.
2. Connaître les bases de la manipulation du DOM en JavaScript.
3. Manipuler les éléments HTML.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Arbre DOM

Notion de l'arbre DOM

- **DOM** (Document Object Model) est une interface de programmation (API) normalisée par le W3C.
- Son rôle est d'accéder au contenu du navigateur web, et le modifier, en utilisant des scripts.
- Le DOM représente un document HTML sous forme d'un **arbre d'objets** (un paragraphe, une image, un style, etc).
- La modification du document HTML à l'aide du DOM consiste alors à ajouter ou supprimer des nœuds de l'arbre.
- DOM offre un ensemble de méthodes pour accéder aux éléments HTML.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Arbre DOM

Notion de l'arbre DOM

Avec le modèle objet, JavaScript peut créer du contenu HTML dynamique en :

- Modifiant / Ajoutant / Supprimant les éléments HTML de la page ;
- Modifiant / Ajoutant / Supprimant les attributs des éléments HTML existants ;
- Modifiant les styles CSS de la page ;
- Réagissant aux événements HTML dans la page.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Arbre DOM

Notion de l'arbre DOM

Exemple: Soit le code HTML suivant correspondant à une page web. L'arbre DOM correspondant est représenté dans la figure ci-dessous :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Index</title>
</head>
<body>
  <h1>Ma page web</h1>
  <p>Bonjour, nous sommes les stagiaires de la filière développement
digital</p>
  <p>Nous étudions à l'<a href="http://www.ofppt.ma">OFPPT</a></p>
</body>
</html>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Arbre DOM

Notion de l'arbre DOM

Exemple:

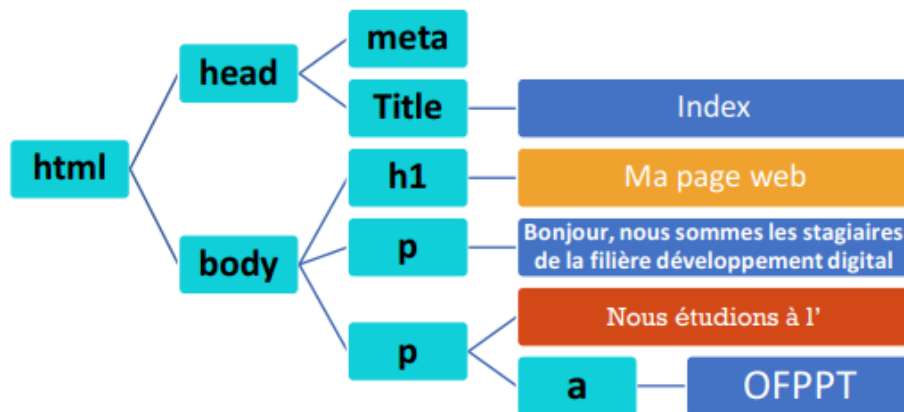


Figure 11: Arbre DOM d'une page HTML

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Objet Document

Objet « document »

- L'objet **document** correspond à l'élément <html> de la page Web.
- La variable document est la racine du DOM.
- Cette variable est un objet et dispose des propriétés **head** et **body** qui permettent d'accéder respectivement aux éléments <head> et <body> de la page.

```
var h = document.head; // La variable h contient l'objet head du DOM
console.log(h);
var b = document.body; // La variable b contient l'objet body du DOM
console.log(b);
```

- L'objet **document** dispose d'un ensemble de méthodes et de propriétés permettant d'accéder et de manipuler le code html.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Objet Document

Méthodes de recherche d'éléments html

Méthode	Description
<code>document.getElementById(id)</code>	Retourne un élément par la valeur de l'attribut ID
<code>document.getElementsByTagName(name)</code>	Retourne les éléments par nom de balise
<code>document.getElementsByClassName(name)</code>	Retourne les éléments par nom de classe CSS

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Objet Document

Méthodes de recherche d'éléments html

Exemple :

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <h1 id="p1" class="c1">cours DOM JS</h1>
  <p class="c1">1er paragraphe</p>
  <p class="c2">2ème paragraphe</p>
</body>
</html>
```

```
let e1=document.getElementById("p1");
console.log(e1);//<h1 id="p1" class="c1">cours DOM JS</h1>

let e2=document.getElementsByTagName("p");
console.log(e2);//[HTMLCollection(2) [p.c1, p]]

let e3=document.getElementsByClassName("c1");
console.log(e3);
//HTMLCollection(2) [h1#p1.c1, p.c1, p1: h1#p1.c1]
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Objet Document

Méthodes d'ajout et suppression d'éléments

Méthode	Description
<code>document.createElement(element)</code>	Créer un élément HTML
<code>document.removeChild(element)</code>	Supprimer un élément HTML
<code>document.appendChild(element)</code>	Ajouter un élément HTML enfant
<code>document.replaceChild(new, old)</code>	Remplacer un élément HTML
<code>document.write(text)</code>	Écrire dans un document HTML
<code>document.getElementById(id).onclick = function(){code}</code>	Ajouter un événement de clic à l'élément sélectionné

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Objet Document

Propriétés des éléments DOM

Méthode	Description
<code>element.innerHTML</code>	Permet de récupérer tout le contenu HTML d'un élément du DOM
<code>element.attribute</code>	Changer l'attribut d'un élément
<code>element.style.property</code>	Changer le style d'un élément
<code>element.textContent</code>	Renvoie tout le contenu textuel d'un élément du DOM, sans le balisage HTML
<code>element.classList</code>	Permet de récupérer la liste des classes d'un élément du DOM

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Relations entre les nœuds

Les éléments du DOM sont appelés des nœuds, qui sont en relation hiérarchique sous forme d'un arbre. Le nœud supérieur est appelé racine (ou nœud racine).

La relation entre les nœuds peut être qualifiée de relation :

- **Parent / Child:** des nœuds peuvent avoir des **ascendants** et des **descendants**
- Nœuds ascendants sont les nœuds qui sont parents d'un nœud (ou parents d'un nœud parent) ;
- Nœuds descendants sont les nœuds qui sont enfants d'un nœud (ou enfants d'un nœud enfant) ;
- Chaque nœud a exactement un parent, sauf la racine.
- Un nœud peut avoir plusieurs enfants.
- **Sibling** : correspondent aux frères d'un nœud, c'est-à-dire les nœuds avec le même parent.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Relations entre les nœuds

```
<ul>  
  <li>1er élément</li>  
  <li>2ème élément  
    <p>paragraphe</p>  
    <a>Lien</a>  
  </li>  
  <li>3ème élément</li>  
</ul>
```

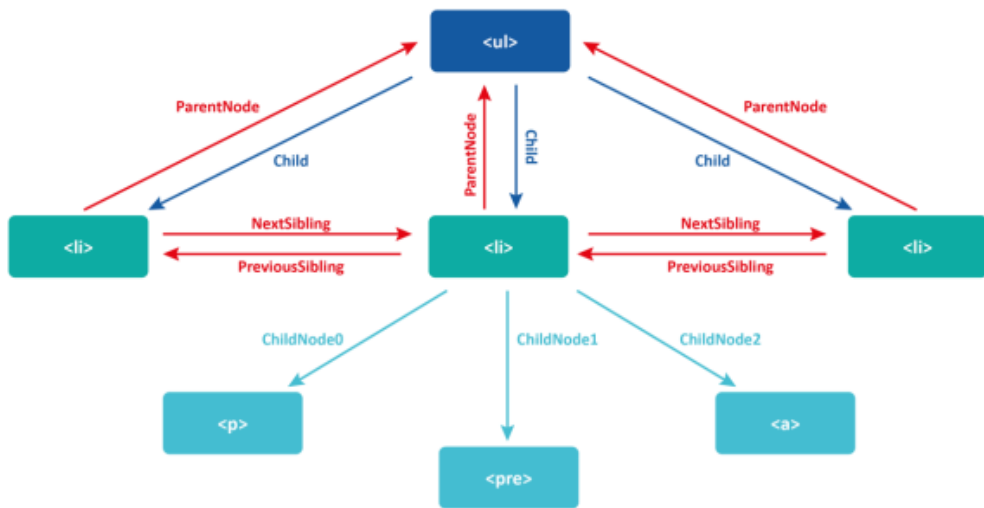


Figure 12 : Relation entre les nœuds

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Types de nœuds du DOM

Chaque objet du DOM a une propriété **nodeType** qui indique son type. La valeur de cette propriété est :

- document.**ELEMENT_NODE** pour un nœud "élément" (balise HTML).
- document.**TEXT_NODE** pour un nœud textuel.

Exemple :

```
if (document.body.nodeType === document.ELEMENT_NODE)
{
    console.log("Body est un nœud élément");
}
else
{
    console.log("Body est un nœud textuel");
}
//Body est un nœud élément
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Types de nœuds du DOM

Chaque objet du **DOM** de type **ELEMENT_NODE** possède une propriété **childNodes** qui correspond à une collection de ses différents enfants :

- On peut connaître la taille de la collection avec la propriété **length** ;
- On peut accéder aux éléments grâce à leur indice ;
- On peut parcourir la collection avec une boucle **for**.



Remarque

- les retours à la ligne et les espaces entre les balises dans le code HTML sont considérés par le navigateur comme des nœuds textuels


1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Types de nœuds du DOM

Exemple :

```
console.log(document.body.childNodes[1]);  
//Affiche <h1 id="p1" class="c1">cours DOM JS</h1>  
  
// Afficher les noeuds enfant du noeud body  
for (var i = 0; i < document.body.childNodes.length; i++)  
    console.log(document.body.childNodes[i]);  
  
for(let i=0; i < document.body.childNodes[1].childNodes.length; i++)  
    console.log(`${i} contient  
${document.body.childNodes[1].childNodes[i]}`);  
// 0 contient [object Text]
```



```
► #text  
  <h1 id="p1" class="c1">cours DOM JS</h1>  
► #text  
  <p class="c1">1er paragraphe</p>  
► #text  
  <p class="c2">2ème paragraphe</p>  
► #text
```

Figure 13 : Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Types de nœuds du DOM

Chaque objet du **DOM** possède une propriété **parentNode** qui renvoie son nœud parent sous la forme d'un objet DOM.

Le parent de l'élément document est **null**.

Exemple :

```
console.log(document.parentNode); // Affiche null

var h1 = document.body.childNodes[1];

console.log(h1.parentNode); // Affiche le noeud body
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM

Navigation dans les nœuds enfants

- **firstChild** : Retourne le premier enfant de l'élément.
- **firstElementChild** : Retourne le premier élément enfant du parent.
- **lastChild** : Retourne le dernier enfant de l'élément.
- **lastElementChild** : Retourne le dernier élément enfant du parent.
- **childNodes** : Retourne tous les enfants de l'élément sous forme d'une collection.
- **children** : Renvoie tous les enfants qui sont des éléments sous forme d'une collection.

Navigation dans les nœuds parents

- **parentNode** : Renvoie le nœud parent de l'élément.
- **parentElement** : Renvoie le nœud de l'élément parent de l'élément.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM

Navigation dans les nœuds frères

- **nextSibling** : Renvoie le nœud frère correspondant au prochain enfant du parent.
- **nextElementSibling** : Renvoie l'élément frère correspondant au prochain enfant de son parent.
- **previousSibling** : Renvoie le nœud frère qui est un enfant précédant de son parent.
- **previousElementSibling** : Renvoie l'élément frère qui est un enfant précédant de son parent.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **firstChild**

La propriété **firstChild**, appelée sur un nœud, retourne le premier nœud de l'élément. Ce nœud n'est pas nécessairement un élément, il peut également contenir du texte ou un commentaire.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

Exemple :

```
<div id="parent">
  <h1>Un titre</h1>
  <p>Un paragraphe</p>
</div>

<script>
  let elt      = document.getElementById("parent");
  let premElt  = elt.firstChild;
  console.log(premElt); // text node
  console.log(premElt.nodeName); // #text
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)



Remarque

- Re-exécuter ce code en supprimant l'espace entre l'élément « div » et la balise « h1 »

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **firstElementChild**

La propriété **firstElementChild** retourne le premier enfant, de type élément, du parent.

Exemple :

```
<div id="parent">
  <h1>Un titre</h1>
  <p>Un paragraphe</p>
</div>

<script>
  let element = document.getElementById("parent");
  let premElt = element.firstElementChild.nodeName;
  console.log(premElt); // h1
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **lastChild**

La propriété **lastChild** permet de sélectionner le dernier enfant de l'élément parent. Elle renvoie **null** s'il n'y a pas d'enfant.

Exemple :

```
<div id="parent">
  <h1>Un titre</h1>
  <p>Un paragraphe</p>
</div>

<script>
  let element = document.getElementById("parent");
  let DernElt = element.lastChild.nodeName;
  console.log(DernElt); // #text
</script>
```


1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **lastElementChild**

La propriété **lastElementChild** retourne le dernier enfant, de type élément, du parent.

Exemple :

```
<div id="parent">
  <h1>Un titre</h1>
  <p>Un paragraphe</p>
</div>

<script>
  let element = document.getElementById("parent");
  let DernElt = element.lastElementChild.nodeName;
  console.log(DernElt); // P
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **childNodes**

La propriété **childNodes** d'un nœud retourne une liste de nœuds enfants d'un élément donné. Les indices des éléments enfants commencent à 0.

Exemple :

```
<div id="parent">
  <h1>Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let element = document.getElementById("parent");
  for (let i = 0; i < element.childNodes.length; i++) {
    console.log(element.childNodes[i]);
  }
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `childNodes`

Exemple :

```
▶ #text
```

```
<h1>Un titre</h1>
```

```
▶ #text
```

```
<p>Un paragraphe</p>
```

```
▶ #text
```

```
<a href="#">link</a>
```

```
▶ #text
```

La liste « `childNodes` » comprend les nœuds '#text' et 'element'.

Figure 14 : Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **children**

La propriété children, appelée sur l'élément parent, permet d'obtenir uniquement les nœuds de type élément.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- children

Exemple :

```
<div id="parent">
  <h1>Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let parent = document.getElementById("parent");
  for (let i = 0; i < parent.children.length; i++)
  {
    console.log(parent.children[i].nodeName + " - " +
parent.children[i].textContent);
  }
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- children

Exemple :

H1 - Un titre

P - Un paragraphe

A - link

Figure 15 : Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **parentNode**

La propriété **parentNode** retourne l'élément parent de l'élément appelant ou **null** (si le parent n'existe pas).

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- parentNode

Exemple :

```
<div id="parent">
  <h1 id="id1">Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let element = document.getElementById("id1");
  let parent = element.parentNode;
  console.log("élément parent : " +
parent.nodeName);
  console.log(parent);
</script>
```


1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- parentNode

Exemple :

élément parent : BODY

```
▶ <body>...</body>
```

Figure 16 : Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **parentElement**

La propriété **parentElement** renvoie le parent de l'élément. La différence entre **parentNode** et **parentElement** est montrée dans l'exemple suivant :

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `parentElement`

```
<div id="parent">
  <h1 id="id1">Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let element = document.getElementById("id1");
  let parent = element.parentElement;
  console.log("Élément parent - " + parent.nodeName);
  // parentNode vs parentElement
  console.log(document.documentElement.parentNode); // document
  console.log(document.documentElement.parentElement); // null
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `parentElement`

```
Elément parent - BODY
```

```
▶ #document
```

```
null
```

Figure 17 : Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **nextSibling**

La propriété **nextSibling** permet d'accéder à l'élément frère d'un élément. L'élément retourné n'est pas nécessairement un nœud d'élément.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- nextSibling

Exemple :

```
<div id="parent">
  <h1 id="id1">Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let element = document.getElementById("id1");
  let next = element.nextSibling;
  console.log("Élément frère suivant - " + next.nodeName);
  console.log(next); </script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- nextSibling

Exemple :

```
Élément frère suivant - #text
```

```
▶ #text
```

Figure 18: Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **nextElementSibling**

La propriété **nextElementSibling** permet d'obtenir le nœud d'élément immédiatement suivant de l'élément appelant.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `nextElementSibling`

Exemple :

```
<div id="parent">
  <h1 id="id1">Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let element = document.getElementById("id1");
  let EltSuiv = element.nextElementSibling;
  console.log("Élément frère suivant - " + EltSuiv.nodeName);
  console.log(EltSuiv);
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- nextElementSibling

Exemple :

Elément frère suivant - P

```
<p>Un paragraphe</p>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **previousSibling**

La propriété **previousSibling** appelée sur un élément, permet d'obtenir le nœud précédent.

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- previousSibling

Exemple :

```
<div id="parent">
  <h1 id="id1">Un titre</h1>
  <p>Un paragraphe</p>
  <a href="#">Un lien</a>
</div>

<script>
  let element = document.getElementById("id1");
  let eltPrec = element.previousSibling;
  console.log("Element frère précédant - " + eltPrec.nodeName);
  console.log(eltPrec);
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `previousSibling`

Exemple :

```
Element frère précédant - #text  
▶ #text
```

Figure 20 : Résultat du code

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- **previousElementSibling**

La propriété **previousElementSibling** appelée sur un élément, permet d'obtenir le nœud précédent (de type élément).

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `previousElementSibling`

Exemple :

```
<div id="parent">
  <h1 id="id1">Un titre</h1>
  <p id="id2">Un paragraphe</p>
  <a href="#">Un lien</a>
</div>
<script>
  let element = document.getElementById("id2");
  let eltPrec = element.previousElementSibling;
  console.log("Element frère précédant - " + eltPrec.nodeName);
  console.log(eltPrec);
</script>
```

1. Comprendre l'arbre DOM, les nœuds parents et enfants

Navigation dans le DOM

Navigation entre les nœuds de l'arborescence DOM (Exemples)

- `previousElementSibling`

Exemple :

```
Element frère précédant - H1
```

```
<h1 id="id1">Un titre</h1>
```

Figure 21 : Résultat du code

2. Connaître les bases de la manipulation du DOM en JavaScript

Sélecteurs (simples, multiples...)

Sélecteurs CSS

En JavaScript, on peut chercher les éléments par leur sélecteur CSS :

- La méthode **querySelector()** renvoie le premier élément qui correspond à un ou plusieurs sélecteurs CSS spécifiés.
- La méthode **querySelectorAll()** renvoie tous les éléments correspondants.

Syntaxe :

```
document.querySelector(sélecteur CSS)  
document.querySelectorAll(sélecteur CSS)
```

2. Connaître les bases de la manipulation du DOM en JavaScript

Sélecteurs (simples, multiples...)

La méthode `querySelector()`

Exemples :

- //Obtenir le premier élément <p> dans le document :
`document.querySelector("p");`
- //Obtenir le premier élément <p> du document qui a class="par":
`document.querySelector("p.par");`
- //Modifier le texte de l'élément dont l'attribut id="id1":
`document.querySelector("#id1").innerHTML = "Bonjour!";`
- //Obtenir le premier élément <p> dans le document où le parent est un élément <div> :
`document.querySelector("div > p");`
- //Obtenir le premier élément <a> dans le document qui a un attribut "target":
`document.querySelector("a[target]");`

2. Connaître les bases de la manipulation du DOM en JavaScript

Sélecteurs (simples, multiples...)

La méthode `querySelectorAll()`

Exemples :

- //Obtenir tous les éléments <p> du document et définir la couleur d'arrière-plan du premier élément <p> (index 0) :

```
let x = document.querySelectorAll("p");  
x[0].style.backgroundColor = "red";
```
- //Obtenir tous les éléments <p> du document qui ont l'attribut class="par", et définir l'arrière-plan du premier élément <p> :

```
let x = document.querySelectorAll("p.par");  
x[0].style.backgroundColor = "red";
```
- // Calculer le nombre d'éléments qui ont l'attribut class="par" (en utilisant la propriété length de l'objet NodeList) :

```
var x = document.querySelectorAll(".par").length;  
// Définir la couleur d'arrière-plan de tous les éléments du document qui ont l'attribut class="par":  
let x = document.querySelectorAll(".par");  
for (let i = 0; i < x.length; i++) { x[i].style.backgroundColor = "red"; }
```
- //Définir la bordure de tous les éléments <a> du document qui ont un attribut "target":

```
let x = document.querySelectorAll("a[target]");  
for (let i = 0; i < x.length; i++) { x[i].style.border = "10px solid red"; }
```

2. Connaître les bases de la manipulation du DOM en JavaScript

Sélecteurs (simples, multiples...)

La méthode `querySelectorAll()`

Exemples :

- //Sélectionner le premier paragraphe du document et modifier son texte avec la propriété `textContent` */
`document.querySelector('p').textContent = '1er paragraphe du document';`
`let documentDiv = document.querySelector('div'); //1er div du document`
- //Sélectionner le premier paragraphe du premier div du document et modifier son texte
`documentDiv.querySelector('p').textContent = '1er paragraphe du premier div';`
- //Sélectionner le premier paragraphe du document avec un attribut `class='bleu'` et changer sa couleur en bleu avec la propriété `style` */
`document.querySelector('p.bleu').style.color = 'blue';`
- //Sélectionne tous les paragraphes du premier div
`let divParas = documentDiv.querySelectorAll('p');`

2. Connaître les bases de la manipulation du DOM en JavaScript

Modes d'Accès aux éléments

Accéder à un élément du DOM

On peut Chercher les éléments directement par nom en utilisant les méthodes suivantes :

- `document.getElementsByTagName()`
- `document.getElementsByClassName ()`
- `document.getElementById ()`
- `document.getElementsByName ()`

Ou bien en utilisant un sélecteur CSS associé :

- `document.querySelector()`
- `document.querySelectorAll()`

2. Connaître les bases de la manipulation du DOM en JavaScript

Modes d'Accès aux éléments

Accéder à un élément en fonction de la valeur de son attribut id

- La méthode **getElementById()** renvoie un objet qui représente l'élément dont la valeur de l'attribut id correspond à la valeur spécifiée en argument.

```
//Sélectionner l'élément avec un id = 'p1' et modifie la couleur du texte  
document.getElementById('p1').style.color = 'blue';
```

2. Connaître les bases de la manipulation du DOM en JavaScript

Modes d'Accès aux éléments

Accéder à un élément en fonction de la valeur de son attribut class

- La méthode **getElementsByClassName()** renvoie une liste des éléments possédant un attribut **class** avec la valeur spécifiée en argument.

```
//Sélectionner les éléments avec une class = 'bleu'  
let bleu = document.getElementsByClassName('bleu');  
for(valeur of bleu){      valeur.style.color = 'blue'; }
```

2. Connaître les bases de la manipulation du DOM en JavaScript

Modes d'Accès aux éléments

Accéder à un élément en fonction de son identité (Nom de la balise)

- La méthode **getElementsByTagName()** permet de sélectionner des éléments en fonction de leur nom.

```
//Sélectionner tous les éléments p du document  
let paras = document.getElementsByTagName('p');  
for(valeur of paras){          valeur.style.color = 'blue'; }
```


2. Connaître les bases de la manipulation du DOM en JavaScript

Modes d'Accès aux éléments

Accéder directement à des éléments particuliers avec les propriétés de Document

L'API DOM fournit également des propriétés permettant d'accéder directement à certains éléments du document. Parmi ces propriétés on trouve :

- La propriété **body** qui retourne le nœud représentant l'élément body ;
- La propriété **head** qui retourne le nœud représentant l'élément head ;
- La propriété **links** qui retourne une liste de tous les éléments « a » ou « area » possédant un attribut href avec une valeur ;
- La propriété **title** qui retourne le titre (le contenu de l'élément title) du document ;
- La propriété **url** qui renvoie l'URL du document sous forme d'une chaîne de caractères ;
- La propriété **scripts** qui retourne une liste de tous les éléments script du document ;
- La propriété **cookie** qui retourne la liste de tous les cookies associés au document sous forme d'une chaîne de caractères.

2. Connaître les bases de la manipulation du DOM en JavaScript

Modes d'Accès aux éléments

Exemple :

```
//Sélectionner l'élément body et appliquer une couleur bleu
document.body.style.color = 'blue';

//Modifier le texte de l'élément title
document.title= 'Le DOM';
```

3. Manipuler les éléments html

Manipulation des éléments

Créer un élément en JavaScript

La méthode **createElement** permet de créer de nouveaux éléments dans le document.
La variable **element** renvoie la référence de l'élément créé.



Remarque

- L'élément créé par la méthode `createElement()` ne s'attache pas automatiquement au document

3. Manipuler les éléments html

Manipulation des éléments

Créer un élément en JavaScript

Exemples:

```
let element1 = document.createElement('p');  
console.log(element1); // <p></p>
```

```
let element2 = document.createElement('div');  
console.log(element2); // <div></div>
```

La méthode createElement convertit le nom de l'élément en minuscule

```
let element3 = document.createElement('DIV');  
console.log(element3); // <div></div>
```

3. Manipuler les éléments html

Manipulation des éléments

Ajouter un élément en JavaScript

Pour ajouter un élément à l'arborescence du DOM (après l'avoir créé), il faut l'attacher à un élément parent.

La méthode **append()** insère un objet en tant que dernier enfant d'un élément parent.

Exemple:

```
let parent = document.getElementById("parent"); // sélectionner un élément parent
let enfant = document.createElement("p"); // Créer un élément enfant
enfant.innerHTML = "C'est un nouveau élément"; // Ajouter un texte à l'élément créée
parent.append(enfant); // Attacher l'enfant à l'élément parent
```

3. Manipuler les éléments html

Manipulation des éléments

Supprimer un élément en JavaScript

La méthode **removeChild()** supprime un élément de la structure du DOM. Le nœud à supprimer est passé en argument à la méthode. Une référence vers le nœud supprimé est retournée à la fin.

Exemple:

```
let parent = document.getElementById("parent"); // sélectionner un élément parent
let enfant = document.getElementById("eltSupp"); // Sélectionner un élément enfant
parent.removeChild(enfant);
```

3. Manipuler les éléments html

Manipulation des éléments

Modifier un élément en JavaScript

La méthode **replaceChild()** remplace un nœud par un autre nœud dans le DOM. Une référence vers le nœud remplacé est retournée à la fin.

Syntaxe:

```
parent.replaceChild(nouveauElement, ancienElement)
```

Exemple:

```
let parent = document.getElementById("parent"); // sélectionner un élément parent
let AncienElement = document.getElementById("id1"); // sélectionner l'ancien élément
let nouvElement = document.createElement("h2"); // Créer un nouveau élément de type <h2>
nouvElement.innerHTML = "C'est le nouveau élément."
parent.replaceChild(nouvElement, AncienElement);
```

3. Manipuler les éléments html

Mise à jour des styles, attributs et classes

Mettre à jour le style

Les propriétés **.style** ou **.className** appliquées sur un élément permettent de changer les styles CSS.

Exemple:

```
<div>
  <div>
    <label>Nom : </label><br>
    <input type="text" class="style1" id="b1">
  </div>
</div>

<script>
// Modifier le style de l'élément qui a l'attribut class=style1
  document.getElementsByClassName("style1").style.borderColor = "red";
</script>
```


3. Manipuler les éléments html

Mise à jour des styles, attributs et classes

Définir le style à l'aide de `element.className`

La propriété **`element.className`** permet de changer les paramètres de style d'un élément HTML en lui attribuant une nouvelle classe dont le nom est passé à l'élément sélectionné.

```
<div>
  <div>
    <label>Nom : </label><br>
    <input type="text" class="style1" id="b1">
  </div>
</div>

<script>
// Modifier le style de l'élément qui a l'attribut class=style1 en lui associant une classe nommée
styleErreur
  document.getElementsByClassName("style1").className = "styleErreur";
</script>
```

3. Manipuler les éléments html

Mise à jour des styles, attributs et classes

Mise à jour d'un attribut avec `setAttribute`

La méthode **`setAttribute()`** est utilisée pour définir un attribut à l'élément spécifié. Si l'attribut existe déjà, sa valeur est mise à jour. Sinon, un nouvel attribut est ajouté avec le nom et la valeur spécifiés.

Exemple 1 : Ajouter les attributs **`class`** et **`disabled`** à l'élément `<button>`

```
<button type="button" id="Btn">Click</button>

<script>
  // sélectionner l'élément
  let btn = document.getElementById("Btn");

  // Ajouter les attributs
  btn.setAttribute("class", "style1");
  btn.setAttribute("disabled", "");
</script>
```

3. Manipuler les éléments html

Mise à jour des styles, attributs et classes

Mise à jour d'un attribut avec `setAttribute`

Exemple 2 : Mettre à jour la valeur de l'attribut href de l'élément `<a>`.

```
<a href="#" id="lien">OFPPT</a>

<script>
  // sélectionner l'élément
  let lien = document.getElementById("lien");

  // Modifier la valeur de l'attribut href
  lien.setAttribute("href", "https://www.ofppt.ma");
</script>
```

3. Manipuler les éléments html

Mise à jour des styles, attributs et classes

Suppression d'attributs d'éléments

La méthode **removeAttribute()** est utilisée pour supprimer un attribut d'un élément spécifié.

Exemple 1 : Supprimer l'attribut href d'un lien.

```
<a href="https://www.ofppt.com/" id="lien">OFPPT</a>

<script>
  // sélectionner l'élément
  let lien = document.getElementById("lien");

  // Supprimer la valeur de l'attribut href
  lien.removeAttribute("href");

</script>
```

3. Manipuler les éléments html

Création DOMMenu Object

Création d'un DOMMenu Objetc

L'objet **DOMMenu** en HTML représente l' élément <menu>.

Syntaxe:

```
var menuObject = document.createElement("MENU")
```

Les attributs :

- **Label** : prend une valeur textuelle, spécifie le label du menu.
- **Type** : prend l'une des valeurs (list, toolbar, contex). Son rôle est de spécifier le type du menu à afficher.



Remarque

- Cet élément n'est plus supporté par les principaux navigateurs

3. Manipuler les éléments html

Création DOMMenu Object

Création d'un DOMMenu Objetc

Exemple:

```
<menu type="context" id="monMenu">
  <menuitem label="Actualiser" onclick="window.location.reload();" icon="ico_reload.png"></menuitem>
  <menu label="Partager sur...">
    <menuitem label="Twitter" icon="ico_twitter.png" onclick="window.open('//twitter.com/intent/tweet?text=
'+window.location.href);">
    </menuitem>
    <menuitem label="Facebook" icon="ico_facebook.png" onclick="window.open('//facebook.com/sharer/sharer.
php?u='+window.location.href);">
    </menuitem>
  </menu>
  <menuitem label="Envoyer cette page" onclick="window.location='mailto:?body='+window.location.href;">
  </menuitem>
</menu>
```