***²JavaScript et le Web***:

Le DOM (Document Object Model).

<!DOCTYPE html>

<html lang="fr">

<head>

<meta charset="UTF-8">

<title>Tuto</title>

</head>

<body>

<h1 id="titre">Titre de mon DOM</h1>

<p class = "para">Du texte dans un paragraphe.</p>

<p>Du texte dans un paragraphe avec un <a href="#">lien</a>.</p>

<script type="text/javascript" src="Lecons js/TestDom.js"></script>

Pour les évènements :

<a href="#">Supprimer cet article</a>

<button>Passez au-dessus de moi</button>

</body>

</html>

* **Les méthodes de récupération** :

**getElementsByTagName()** : récupère toutes les balises du même type. Le résultat est donné en tant que tableau (HTMLCollection).

**getElementById()** : récupère les éléments de la balise avec l’identité (id).

**getElementsByClassName()** : récupère les éléments de de la classe.

- **Les nouvelles méthodes de récupération** :

**querySelector()** : récupère la balise , l’identité ou la classe selon le paramètre.("h1"),("#titre"),(".para").

**querySelectorAll()** : querySelector() récupère seulement un élément , avec **querySelectorAll()** on récupère tous les éléments du type demandé. Le résultat s’affiche dans la console comme une liste de nœuds (node liste).

Exemple de récupération :

let monH1 = document.getElementsByTagName("h1") ;

console.log(monH1) ;

S'il y a plusieurs élément h1 sur mon site et que getElementsByTagName retourne un tableau, il faut mettre son indice après le paramètre.

Ex :

let monH1 = document.getElementsByTagName("h1")[0] ;

Ou

monH1[0].textContent = "Hello World" ;

- **Modifier un élément avec des propriétés** :

**textContent**: Cette propriété modifie que le contenu textuel de ma sélection, ici mon h1.

monH1.**textContent** = "Hello World" ;

**innerHTML** : on peut modifier le CSS et le contenu de la balise.

monH1.**innerHTML** = "<div style= 'font-weight: normal'> Hello World ! </div>";

console.log(monH1) ;

- **Trouver un élément** :

On peut facilement naviguer dans le DOM grâce à plusieurs méthodes placées dans une variable.

ex :

La première recherche est par le nom de la balise :

**const** methode1 = document.body.getElementsByName("h1");

console.log(methode1); [h1#titre, titre: h1#titre] HTMLCollection

- Le résultat se trouve entre crochets et donc se comporte comme un tableau.

- Donc si dans le **body** il y a plusieurs **<P>** on peut le retrouver grâce à son indice :

**const** testDomTagName = document.body.getElementsByTagName("p")[1];

console.log(testDomTagName);

Attention, de mettre le lieu de la recherche (ici : body) dans la méthode et le s dans éléments.

- On peut faire une recherche par identité :

const methode2 = document.getElementById("titre");

console.log(methode2);

Ici, pas la peine de mettre l'emplacement (body) car il n'y a en général qu'un ID, et donc pas de s à élément.

- **Déplacer un élément** :

La méthode insertBefore(). mettre avant.

**const** titre = document.getElementById("titre");

**const** texte = document.body.getElementByTagName("p");

document.body.insertBefore(texte[1],titre); => (à déplacer, devant élément)

console.log(titre);

Attention, lorsque l'on déplace le second élément <p> tous en haut, il change alors d'indice, il n'est plus le 1 mais le 0.

Déplacer un élément tous en bas en utilisant la méthode appendChild() :

document.body.appendChild(titre);

console.log(titre); déplace le titre en bas du body.

Il est aussi possible de supprimer un élément avec la méthode removeChild() :

document.body.removeChild(titre);

- **Créer un nouveau texte** :

**const** titre = document.getElementById("titre");

**const** texte = document.body.getElementByTagName("p");

Création du nouveau texte avec la méthode createTextNode().

**const** nouveauTexte = document.body.createTextNode("Mon nouveau texte en JS");

document.body.appendChild(nouveauTexte);

Remplacement d'un texte avec la méthode replaceChild(). En paramètre on met le nouveau texte et ensuite le texte à remplacer :

document.body.replaceChild(nouveauTexte, texte[1]);

[**https://openclassrooms.com/fr/courses/3306901-creez-des-pages-web-interactives-avec-javascript/3501886-parcourez-le-dom#/id/r-3575069**](https://openclassrooms.com/fr/courses/3306901-creez-des-pages-web-interactives-avec-javascript/3501886-parcourez-le-dom#/id/r-3575069)

- **Ajouter des éléments dans le DOM** :(leçon 71)

2 méthodes à connaitre pour cela.

- **innerHTML()**; Met ce qu'il faut dans la balise.

- **createElement()**; Ajoute la balise voulue.

**function** ajoutTexte(pseudo, monTexte){

// creation d'un nouvel élément

**const** nouveauTexte = document.createElement("p");

// remplissage de la balise

nouveauTexte.innerHTML=`<strong>${pseudo}</strong> : ${monTexte}`;

// stockage des valeurs.

document.body.appendChild(nouveauTexte);

}

// L'élément ajouté se place en fin de DOM.

ajoutTexte("FooFun","hello there !");

ajoutTexte("Display", "Go back Home..")

- **Modifier des valeurs d'attributs dans les balises HTML** :

Il faut créer une variable qui va récupérer la balise de l’attribut.

**const** lien = document.body.getElementsByTagName("a");

Ensuite pour voir et modifier le fichier il faut faire appel à la méthode setAttribute().

lien.setAttribute("href","1234");

console.log(lien.getAttribute("herf")); 1234.

- **Modifier le style (CSS)** :

ex :

const *titre* = document.getElementById("titre");

*titre*.style.color = "red";

*titre*.style.fontFamily = "cursive";

*titre*.style.boxShadow = "2px 2px 10px rgba(3,5,55,0.3)";

Pour le - de font-family, on le remplace par le camelcase.

- **la sélection des éléments plus précise (ES6)** :

Le **querySelector()** permet de sélectionner le premier élément choisi, (une balise, un id, une classe), mais mettons que l'on choisisse une balise **<p>** avec la classe *maClass*, et qu'il y ai plusieurs balise **<p>** avec cette même classe. **querySelector()** sélectionnera seulement la première, si l'on veut toutes les sélectionné, alors il faut utiliser la méthode **querySelectorAll().**

ex :

**const** *titre* = document.getElementById("titre");

**const** *lien* = document.body.getElementsByTagName("a");

On peut remplacer cette **const** par :

**const** *titre* = document.querySelector("h1");

**const** *lien* = document.querySelector("a");

h1 ou autre chose, comme un id, une classe.

- **Ajouter et supprimer des classes en CSS** :

(Voir leçon 77 avec Antho.)

Agit sur le CSS écrit dans le DOM.

ex :

**const** *titre* = document.querySelector("h1"); le titre h1

**const** *liens* = document.querySelectorAll("a"); tous les liens a

*liens*[1].addEventListener("click", function(){

Utilise une méthode **addEventListener()** qui agit sur les liens **<a>** de la liste, et c'est pour celà qu'il y a un indice après le nom de la variable. "click" est une des propriétés qui agira sur le titre. Et enfin une fonction anonyme pour actionner le tous, ici on supprime la classe.

*titre*.classList.remove("couleurTitre"); //supprime une classe.

})

Récapitulatif sommaire (Louis-Nicolas) :

Accéder aux éléments

* **getElementsByTagName()** - Sélectionne tous les éléments avec la balise entre parenthèses
* **getElementById()** - Sélectionne un seul élément : le premier ayant l'ID entre parenthèses
* **getElementsByClassName()** - Sélectionne tous les éléments avec la classe entre parenthèses
* **querySelector()** - Sélectionne un seul élément : celui avec le sélecteur entre parenthèses
* **querySelectorAll()** - Sélectionne tous les éléments avec le sélecteur entre parenthèses

Modifier les éléments

* **textContent** - Modifie le texte d'un élément
* **innerHTML** - Modifie l'HTML d'un élément

Ajouter et supprimer des éléments

* **createElement()** - Crée un élément
* **prepend()** - Ajoute l'élément entre parenthèses devant l'élément cible
* **append()** - Ajouter l'élément entre parenthèses derrière l'élément cible (peut contenir du texte)

document.body.append(header, menu, p);

* **appendChild()** - Ajouter l'élément entre parenthèses derrière l'élément cible (ne peut pas contenir du texte)
* **insertBefore()** - Insère un élément avant l'élément cible

Modifier le style d'un élément

* **style.propriété** - Modifie la propriété CSS spécifiée, par exemple : style.color = "orange"
* **className** - Modifie les classes d'un élément

Les évènements en JavaScript :

La base :

Il faut en premier lieu récupérer les éléments.

let a = document.querySelector("a");

let bouton = document.querySelector("button");

Ensuite, il suffit de mettre : variable.événement= fonction anonyme ou fléchée.

a.onclick = function(){ pour la fonction fléchée : a.onclick = () => {

if(confirm("Etes-vous sûr ?)){

location.href =<http://google.fr>

}

}

boutton.onmouseout = () => {

    document.body.style.backgroundColor="white";

}

boutton.onmouseover = () => {

    document.body.style.backgroundColor="red";

}

Le gestionnaire d'évènements :

Récupérer les éléments.

a.addEventListener('nom de l'évènement', () => {

if(confirm("Etes-vous sûr ?)){

location.href =<http://google.fr>

}

});

Il est possible de passer des paramètres dans la fonction fléchée, qui vont être en relation avec l'évènement, par convention c'est un e (event):

a.addEventListener('click', (e) => {

console.log(e);

});

Dans le navigateur, avec le débuggeur, on peut voir toutes les propriétés liées à e.

Gérer la propagation des évènements et stopPropagation :

<article style="background-color: silver">

<h1 style="display: inline">Titre de l'article</h1></article>

On récupère les éléments du DOM :

let article = document.querySelector("article");

let h1      = document.querySelector("h1");

 ajouter des évènements

article.addEventListener('click',()=>{

    alert('article cliqué');

});

h1.addEventListener('click',()=>{

    alert('h1 cliqué');

});

Le code affiche deux messages popup lors du clic sur le titre. Car le clic enfant (h1) se répercute sur le parent (article)."Problème", on ne veut qu'un popup. Pour cela il faut ajouter une propriété dans les parenthèses et ajouter la méthode stopPropagation.

h1.addEventListener('click',(e)=>{

    alert('h1 cliqué');

    e.stopPropagation();

});

Programmer un évènement :

setTimeout et setInterval.

setTimeout("se que doit faire la fonction", au bout de combien de temps (en milliseconde 1s = 1000ms));

setTimeout("alert('Bonjour')",3000 );

Idem pour setInterval, mais la le second paramètre fera des répétitions.

setInterval("alert('setInterval')",5000);

Là setinterval fera ici un message popup toutes les 5secondes.

Pour arrêter le processus, il y a deux méthodes, mais avant tous il faut mettre les 2 méthodes au-dessus dans une variable (time et interval).

clearTimeout(time);   pour setTimeout()

clearInterval(interval);    pour setInterval()

- **Le scroll d'une page par les liens** :

Mettons que l'on ait une 10ène de titres (10 h1) et de longs paragraphes, avec des liens qui doivent nous mener directement sur le titre voulu.

On commencera par mettre dans une variable tous les titres.

Ex :

const mesTitres = document.querySelectorAll("h1");

A partir de là, on va créer une fonction qui va prendre en compte le nombre pixels qu'il y a du haut de la page au premier titre.

function vaAuTitre (titre){

console.log(titre.offsetTop);

} // offsetTop donne une valeur en pixel du haut de la page à la valeur demandé, ici le titre 2.

console.log(titre[2]);

Pour garder cette valeur on va la stocker dans une variable.

function vaAuTitre (titre){

const distance = titre.offsetTop; !! pas de parenthèses à offsetTop

//console.log(titre.offsetTop);

window.scrollTo(0, distance);

}

window, c'est la page web, la méthode scrollTo (haut de la page en px , puis la valeur distance pour le lieu voulu.)

vaAuTitre(mesTitres[2]);

La page se mettra au niveau du titre 3(se manipule comme un tableau donc le 1er titre = 0).