JS DOM

QU'EST-CE QUE LE DOM?

Le terme DOM (Document Object Model) est un terme standardisé et donc défini de manière officielle.

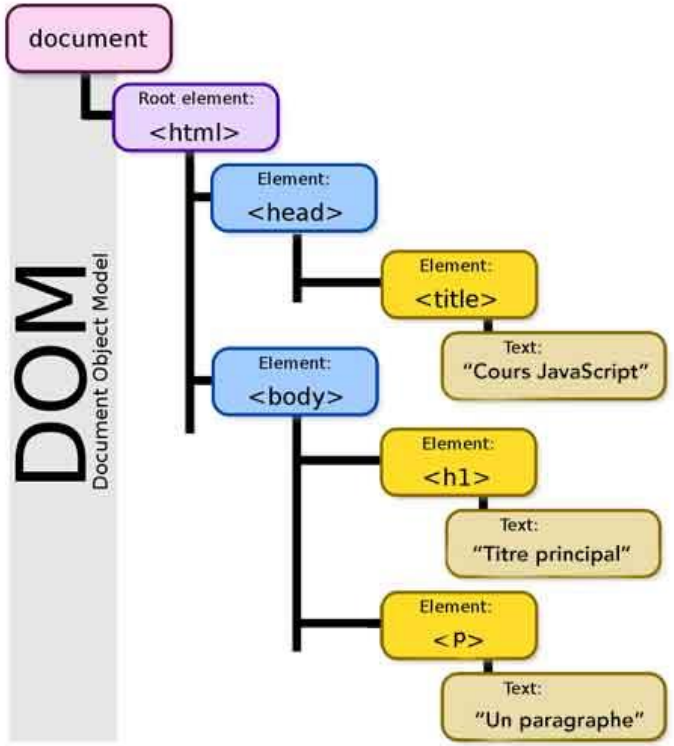
Le DOM est une interface de programmation pour des documents HTML ou XML qui représente le document sous une forme qui permet à JavaScript de modifier la structure, le contenu et les styles.

Le DOM est ainsi une représentation structurée du document sous forme **« d'arbre »** crée automatiquement par le navigateur.

Chaque branche de cet arbre se termine par ce qu'on appelle un nœud qui sont des objets disposant des méthodes et des propriétés pouvant être lus ou modifiés.

NovaoLearning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

QU'EST-CE QUE LE DOM?



Source: https://www.pierre-giraud.com

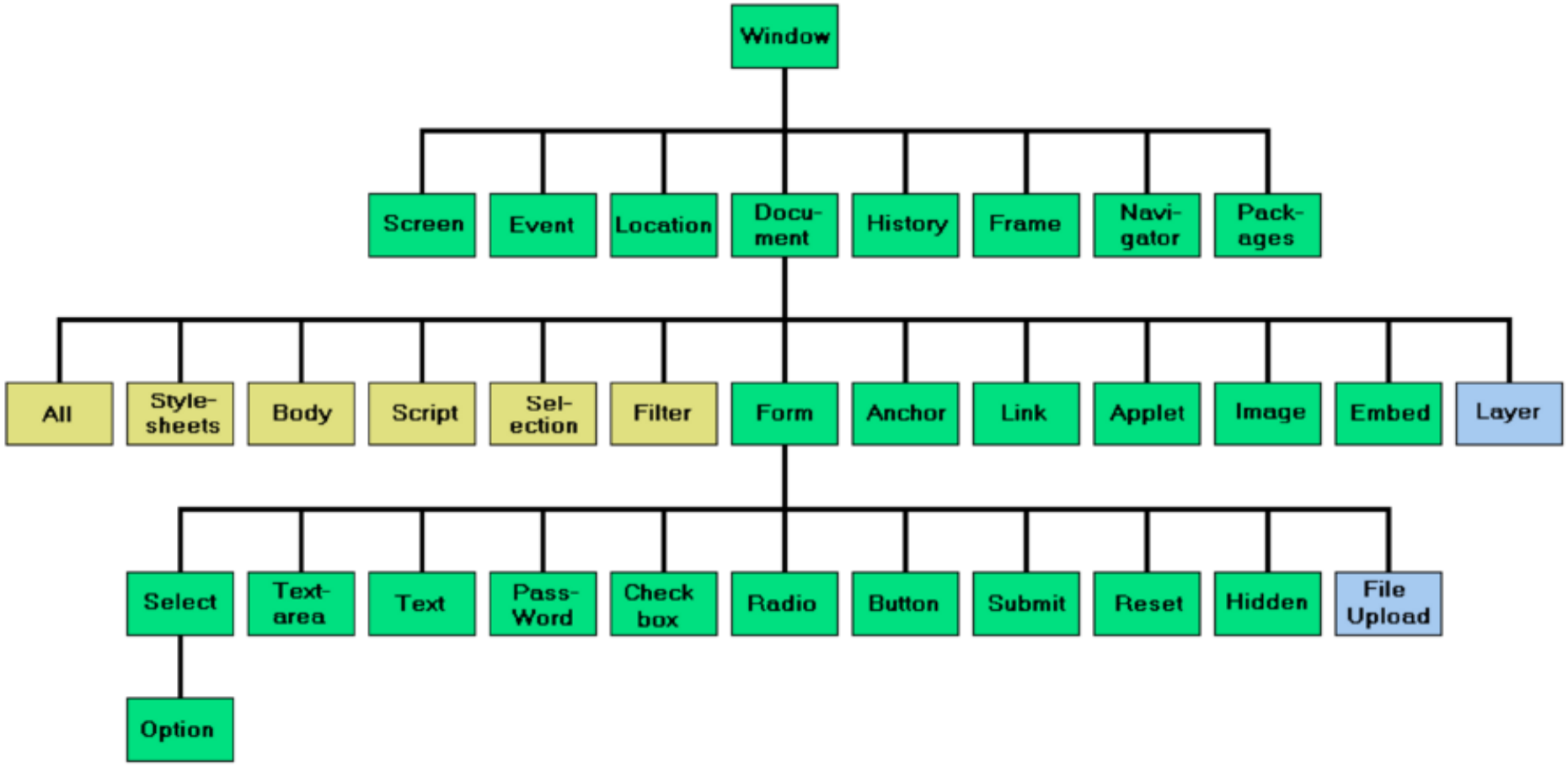
Dans le DOM, on commence toujours par un élément racine qui est le point de départ du document : la balise ***<html>*** .

Celle-ci a pour enfants les balises ***<head>*** et ***<body>.***

Vous trouverez ensuite le contenu de votre page dans la balise ***<body>*** sous forme de liens, boutons, blocs, etc.

QU'EST-CE QUE LE DOM?

Les objets existants dans le navigateur

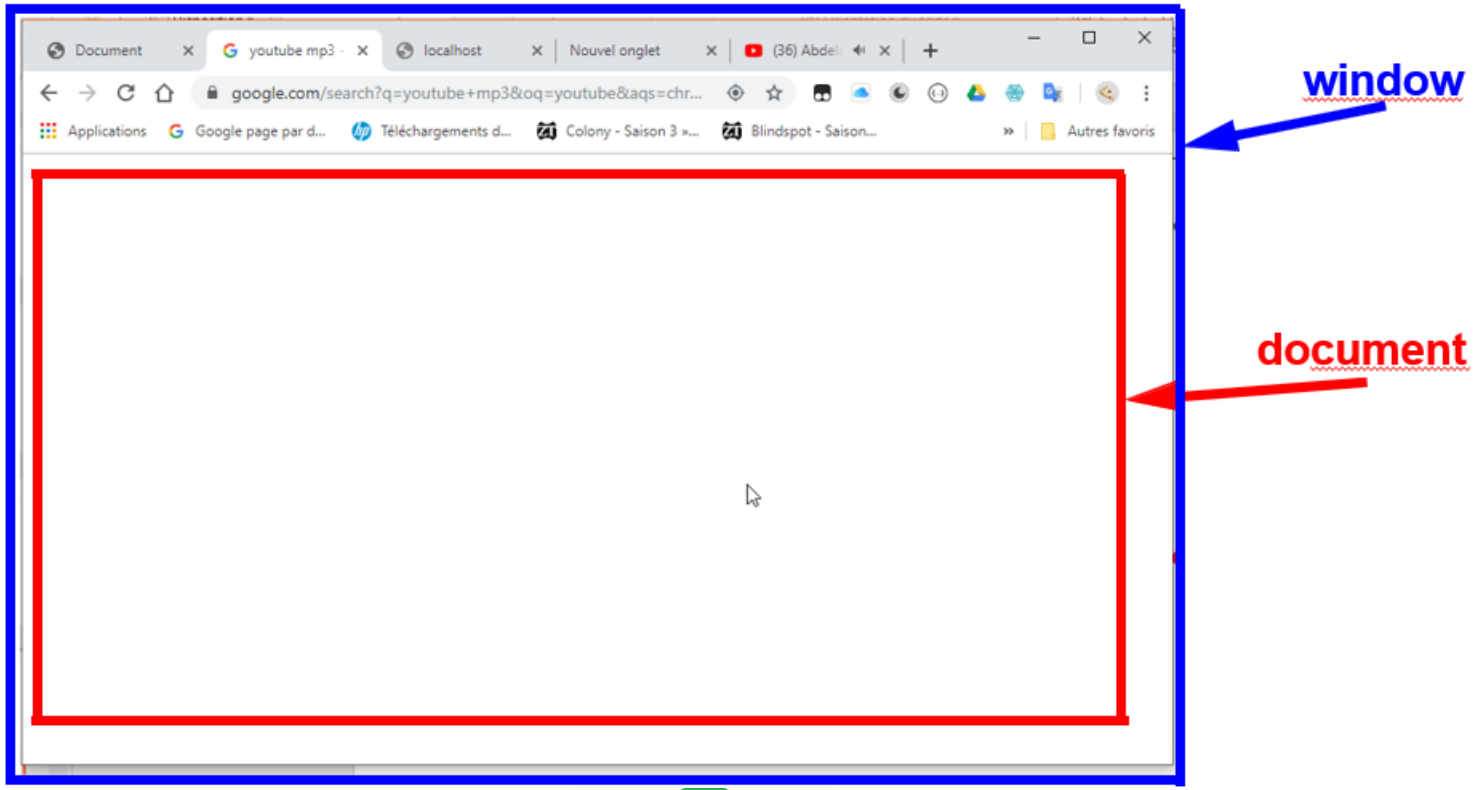


Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

SURVOL DES OBJETS DOM

Il existe quelques variables système qui contiennent les objets les plus importants:

- **document**: accéder et manipuler les contenus et styles d'une page
- **window**: informations sur les fenêtre et création de nouvelles fenêtres
- ***navigator***: informations sur le navigateur (browser).



QU’EST-CE QUE LE DOM?

Avec une interface de programmation nous permettant de parcourir le DOM, nous allons pouvoir interagir avec lui. Ces interactions comprennent :

- La modification du contenu d'un élément précis ;
- La modification du style d'un élément ;
- La création ou la suppression d'éléments ;
- L'interaction avec les utilisateurs, afin de repérer des clics sur un élément ou encore de récupérer leur nom dans un formulaire ;
- Etc.

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ACCÉDER AUX ÉLÉMENTS DU DOM

Chaque élément du DOM est un **objet** JavaScript avec ses propriétés et ses fonctions pour le manipuler.

Tout commence avec le ***document*** .

Cet **objet**, auquel vous avez directement accès dans votre code JavaScript, est le point de départ du DOM.

Il représente votre page entière (votre document) .

C'est donc lui qui contient les **fonctions** dont vous aurez besoin pour retrouver les éléments que vous cherchez.

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ACCÉDER AUX ÉLÉMENTS DU DOM

- `document.getElementById();`

C’est sans doute la méthode la plus utilisée pour retrouver un **élément précis**.

Comme son nom l'indique, elle va rechercher un élément grâce à son ***id*** , or, rappelez-vous qu'il ne doit y avoir qu'un seul élément avec un ***id*** donné dans le html, cette méthode est donc une candidate parfaite pour retrouver un élément particulier.

getElementById("valeur de l’id ") prend en paramètre l' ***id*** de l'élément que vous recherchez et vous retournera cet élément s'il a été trouvé.

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ACCÉDER AUX ÉLÉMENTS DU DOM

Exemple : Prenons ce contenu HTML

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no" />
    <title>Cours JS</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <h1 class="titre">Cours JS</h1>
    <p class="para1">
      Lorem ipsum dolor sit, amet consectetur
adipisicing elit. Illo maxime
      explicabo cupiditate ipsam autem.
Estnecessitatibus, architecto esse
      quaerat hic repellat libero nisi ad eos ex vero
at in velit.
    </p>
```

```
<section class="container">
      <form id="my-form">
        <h1>Add User</h1>
        <div class="msg"></div>
        <div>
          <label for="name">Name:</label>
          <input type="text" id="name" name="name" />
        </div>
        <div>
          <label for="email">Email:</label>
          <input type="text" id="email" name="email" />
        </div>
        <input class="btn" type="submit" value="Submit"
/>
      </form>
      <ul id="users"></ul>
    </section>
    <script src="main.js"></script>
  </body>
</html>
```

ACCÉDER AUX ÉLÉMENTS DU DOM

Créons maintenant un fichier main.js et ajoutons le code suivant :

```
// Sélectionner un élément de formulaire par son identifiant (id)
const form = document.getElementById("my-form");

// Afficher l'élément de formulaire dans la console
console.log(form);
```

En observant la console de votre navigateur, vous constaterez que celle-ci a été récupérer l'élément du DOM qui contient l'attribut ***id*** avec la valeur « ***my-form*** » pour l'afficher.

D'autres exemples : https://developer.mozilla.org/fr/docs/Web/API/Document/getElementById

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ACCÉDER AUX ÉLÉMENTS DU DOM

- `document.getElementsByClassName();`

Cette méthode fonctionne de la même manière que la précédente, mais fera sa recherche sur la ***class*** des éléments et retournera la liste des éléments qui correspondent.

Exemple :

```
// Sélectionner tous les éléments avec la classe "container"
const container = document.getElementsByClassName("container");

// Afficher les éléments dans la console
console.log(container);
```

Ceci récupère tous les éléments qui ont l'attribut ***class*** avec la valeur « ***container*** »

D'autres exemples :
https://developer.mozilla.org/fr/docs/Web/API/Document/getElementsByClassName

ACCÉDER AUX ÉLÉMENTS DU DOM

- `document.getElementsByName();`

Avec cette méthode, vous rechercherez tous les éléments avec l'attribut ***name*** ayant la valeur précisée dans la parenthèse.

Exemple :

```
// Sélectionner tous les éléments avec le nom "email"
const inputMail = document.getElementsByName("email");

// Afficher les éléments dans la console
console.log(inputMail);
```

D'autres exemples :
https://developer.mozilla.org/fr/docs/Web/API/Document/getElementsByName

ACCÉDER AUX ÉLÉMENTS DU DOM

- `document.getElementsByTagName();`

Pour cette méthode, vous rechercherez tous les éléments avec un nom de balise bien précis (par exemple tous les liens (***a***), tous les boutons (***button***)...).

De la même manière que la méthode précédente, vous récupérerez la liste des éléments correspondants.

Exemple :

```
// Sélectionner tous les éléments <ul>
const ul = document.getElementsByTagName("ul");

// Afficher les éléments dans la console
console.log(ul);
```

D'autres exemples :
https://developer.mozilla.org/fr/docs/Web/API/Document/getElementsByTagName

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ACCÉDER AUX ÉLÉMENTS DU DOM

- `document.querySelector();`

Cette méthode est plus complexe, mais aussi beaucoup plus puissante.

Elle renvoie le premier élément qui correspond à un ou plusieurs sélecteurs CSS spécifiés dans le document.

Exemple :

```
// Sélectionner l'élément <label> avec l'attribut 'for' égal à 'email'
const labelFor = document.querySelector("label[for=email]");

// Afficher l'élément dans la console
console.log(labelFor);
```

La valeur dans la parenthèse est un sélecteur CSS, celle de l'exemple permet d'aller récupérer l'élément dont la balise est ***label***, dont l'attribut est ***for*** et dont la valeur de ***for*** est ***email***.

ACCÉDER AUX ÉLÉMENTS DU DOM

- `document.querySelectorAll();`

La méthode querySelector() ne renvoie que le premier élément qui correspond aux sélecteurs spécifiés.

Pour renvoyer toutes les correspondances, utilisez plutôt la méthode querySelectorAll().

Exemple :

```
// Sélectionner tous les éléments <input>
const allInputs = document.querySelectorAll("input");

// Afficher les éléments dans la console
console.log(allInputs);
```

D'autres exemples : https://developer.mozilla.org/fr/docs/Web/API/Document/querySelector

https://developer.mozilla.org/fr/docs/Web/API/Document/querySelectorAll

MODIFIER LES ÉLÉMENTS DU DOM

➢Modifiez le contenu d'un élément

Pour commencer, voyons déjà les propriétés permettant de modifier directement le contenu de notre élément.

Les deux principales sont : ***innerHTML*** et ***textContent*** .

innerHTML demande à ce que vous entriez du texte représentant un contenu HTML.

Exemple :

```
// Sélectionner le premier élément avec la classe "para1"
const para1Element = document.querySelector(".para1");

// Remplacer le contenu HTML de l'élément
para1Element.innerHTML = "<span class='p2'>para de classe p2 </span>";
```

MODIFIER LES ÉLÉMENTS DU DOM

La propriété ***textContent*** , quant à elle, demande un simple texte qui ne sera pas interprété comme étant du HTML.

Exemple :

```
// Modifier le contenu textuel de l'élément
para1Element.textContent = "para modifié";
```

Pour en savoir plus :

innerHTML : https://developer.mozilla.org/fr/docs/Web/API/Element/innerHTML

textContent : https://developer.mozilla.org/fr/docs/Web/API/Node/textContent

MODIFIER LES ÉLÉMENTS DU DOM

➢Modifiez des classes

Il est aussi possible d'accéder directement à la liste des classes d'un élément avec la propriété ***classList*** .

Cette propriété ***classList*** fournit aussi une série de fonctions permettant de modifier cette liste de classes.

En voici quelques-unes :

- ***add*** => Ajoute les classes spécifiées
- ***remove*** => Supprime les classes spécifiées
- ***toggle*** => change la présence d'une classe dans la liste. Ajoute la classe si elle n'est pas présente, et la supprime si elle est déjà présente. Cette méthode est utile pour basculer dynamiquement l'état d'une classe.

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

MODIFIER LES ÉLÉMENTS DU DOM

Exemples :

```
// Ajouter la classe "paragraphe" à la balise <p>
document.querySelector("p").classList.add("paragraphe");

// Supprimer la classe "paragraphe" de la balise <p>
document.querySelector("p").classList.remove("paragraphe");

// Toggle (ajoute ou supprime) la classe "paragraphe" à la balise <p>
document.querySelector("p").classList.toggle("paragraphe");
```

Pour en savoir plus : https://developer.mozilla.org/fr/docs/Web/API/Element/classList#Méthodes

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

MODIFIER LES ÉLÉMENTS DU DOM

➢Changez les styles d'un élément

Avec la propriété ***style*** , vous pouvez récupérer et modifier les différents styles d'un élément.

style est un objet qui a une propriété pour chaque style existant.

Elle reprend les même propriétés qu'en "CSS", mais écrit en camelCase :

background-color → backgroundColor

```
// Sélectionner l'élément par sa classe
const para1Element = document.querySelector(".para1");

// Modifier les styles avec la propriété style

// Changer la couleur du texte en rouge
para1Element.style.color = "red";

// Changer la taille de la police à 20 pixels
para1Element.style.fontSize = "20px";

// Changer la couleur de fond en bleu clair
para1Element.style.backgroundColor = "lightblue";
```

Pour en savoir plus : https://developer.mozilla.org/fr/docs/Web/API/HTMLElement/style

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

MODIFIER LES ÉLÉMENTS DU DOM

➢Modifiez les attributs

Pour définir ou remplacer les attributs d'un élément, vous pouvez utiliser la fonction ***setAttribute*** .

Exemples :

```
// Change le type de l'input en un type password
document.getElementById("name").setAttribute("type", "password");

// Change le nom de l'input en "my-password"
document.getElementById("name").setAttribute("name", "my-password");
```

Pour en savoir plus : https://developer.mozilla.org/fr/docs/Web/API/Element/setAttribute

CRÉER DES ÉLÉMENTS DU DOM

➢**Créez de nouveaux éléments et les ajouter en tant qu'enfants**

La fonction ***document.createElement*** permet de créer un nouvel élément du type spécifié que nous pourrons insérer dans notre DOM.

Exemples :

Avec le code à droite, nous venons de créer un nouvel élément de type ***div*** , mais qui n'est pas encore rattaché au DOM.
Nous allons ensuite récupéré l'élément ayant pour id ***my-form*** .

```
// Créer un nouvel élément <div>
const newDiv = document.createElement("div");

// Ajouter du texte au nouvel élément
newDiv.textContent = "Nouveau contenu dans la div";

// Récupérer l'élément existant avec l'ID "my-form"
const myForm = document.getElementById("my-form");
```

CRÉER DES ÉLÉMENTS DU DOM

Enfin, nous allons ajouter notre nouvel élément dans les enfants de l'élément ***#my-form*** .

```
/* Attacher le nouvel élément en tant qu'enfant
de l'élément avec l'ID "my-form" */
myForm.appendChild(newDiv);
```

Pour en savoir plus : https://developer.mozilla.org/fr/docs/Web/API/Node/appendChild

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

SUPPRIMER OU REMPLACER DES ÉLÉMENTS DU DOM

➢Supprimez et remplacez des éléments

Il existe les fonctions ***removeChild*** et ***replaceChild*** , afin de respectivement supprimer et remplacer un élément.

Exemple :

Supprimer un élément :

```
// Retirer le newDiv de son élément parent
myForm.removeChild(newDiv);
```

modifier un élément par un autre :

```
// Créer un bouton, qui va servir d'élément de remplacement
const newButton = document.createElement("button");
newButton.textContent = "Cliquez-moi";


// Remplacer 'newDiv' par le bouton
myForm.replaceChild(newButton, newDiv);
```

Pour en savoir plus :

https://developer.mozilla.org/fr/docs/Web/API/Node/removeChild

https://developer.mozilla.org/fr/docs/Web/API/Node/replaceChild

ÉCOUTER DES EVENTS (ÉVÉNEMENTS)

➢Qu'est-ce qu'un événement ?

Un **événement** ou **event** est une réaction à une action émise par l'utilisateur, comme le clic sur un bouton ou la saisie d'un texte dans un formulaire.

Un **événement** en JavaScript est représenté par un nom (***click*** , ***mousemove*** ...) et une fonction que l'on nomme une **callback** .

Un événement est par défaut propagé, c'est-à-dire que si nous n'indiquons pas à l'événement que nous le traitons, il sera transmis à l'élément parent, et ainsi de suite jusqu'à l'élément racine.

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

LISTE DES HTML EVENTS

Event Handler	Event	element	window	Utilisateur
onmouseout	mouseout	x	x	on déplace la souris ailleurs
onmouseover	mouseover	x	x	on déplace la souris sur l'objet
onmouseup	mouseup	x	x	on lache un bout de souris
onresize	resize	x	x	on diminue une fenetre
onload	load		x	on charge une fenêtre dans les navi.
onunload	unload			contenu part (on charge un autre document)
onclose	close		x	on ferme une fenêtre
ondragdrop	dragdrop		x	
onscroll	scroll		x	on scroll le contenu d'une fenêtre

Event Handler	Event	element	window	Utilisateur
onblur	blur	x	x	on focalise sur un autre objet (bouger, cliquer, etc ailleurs)
onclick	click	x	x	on clique sur un objet
ondblclick	dblclick	x		
onfocus	focus	x	x	on focalise sur l'objet
onkeydown	keydown	x	x	une touche reste enfoncée
onkeypress	keypress	x	x	une touche est enfoncée
onkeyup	keyup	x	x	
onmousedown	mousedown	x	x	clique avec la source
onmousemove	mousemove	x	x	on bouche la souris

ÉCOUTER DES EVENTS (ÉVÉNEMENTS)

Cette fonction **callback** , c'est nous qui allons la spécifier.

Elle sera appelée à chaque fois que l'action que l'on désire suivre est exécutée.

Cela signifie que si l'on désire suivre le clic sur un élément, notre fonction sera appelée à chaque fois que l'utilisateur cliquera sur cet élément.

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ÉCOUTER DES EVENTS (ÉVÉNEMENTS)

➢ **Événement lors d'un clic sur un élément**

Afin de réagir lors d'un **clic** sur un élément, il faut *écouter* cet **événement**.

Pour cela, nous avons à notre disposition la fonction ***addEventListener()*** .

Cette fonction nous permet d'écouter tous types d'événements (pas que le clic).

Réagir à un **événement**, c'est faire une action lorsque celui-ci se déclenche.
Écouter, c'est vouloir être averti quand **l'événement** se déclenche.

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

ÉCOUTER DES EVENTS (ÉVÉNEMENTS)

```
// Sélectionner le premier élément avec la classe "btn"
const element = document.querySelector(".btn");

// Ajouter un écouteur d'événements pour le clic sur l'élément avec la classe "btn"
element.addEventListener("click", function (e) {
    // Empêcher le comportement par défaut de l'élément
    e.preventDefault();

    // Modifier le contenu de l'élément avec l'ID "users" pour afficher « Hello World »
    document.getElementById("users").innerHTML = "<li>Hello World</li>";
});
```

*En appelant ici la fonction **preventDefault()** dans notre **callback**, nous demandons au gestionnaire des événements de ne pas exécuter le comportement par défaut de notre élément (qui est la redirection vers une autre page pour un lien).*

ÉCOUTER DES EVENTS (ÉVÉNEMENTS)

Pour en savoir plus :

https://developer.mozilla.org/fr/docs/Web/API/EventTarget/addEventListener

https://developer.mozilla.org/fr/docs/Web/Events

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

WINDOW

Jusque-là nous avons vu les différents attributs et méthodes pour sélectionner, modifier et écouter des évènements dans les éléments du DOM grâce à document

document est une partie de 'window', ce dernier représente une entité spéciale appelée "l'objet global" dans le contexte du navigateur web. L'objet 'window' est essentiellement la fenêtre du navigateur et agit comme le conteneur global pour tout le contenu de la page.

En termes simples, le window est le point d'entrée global pour JavaScript dans un navigateur. Il contient des propriétés et des méthodes qui permettent d'interagir avec le navigateur et la fenêtre du document. Certaines de ces propriétés incluent document (qui représente le DOM du document), location (qui représente l'URL du document), console (pour les messages de débogage), etc.

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

WINDOW

Outre document, window possède d'autres propriétés et méthodes pouvant être utile :

Propriétés de base :

window.location: Cette propriété fournit des informations sur l'URL de la page actuelle et permet également de rediriger la page.

```
// Récupérer l'URL actuelle de la fenêtre du navigateur
let currentURL = window.location.href;
```

window.navigator: Donne accès aux informations sur le navigateur.

```
// Afficher la langue par défaut dans la console
console.log("Langue par défaut du navigateur :", defaultLanguage);
```

WINDOW

Méthodes utiles :

window.alert(): Affiche une boîte de dialogue d'alerte avec un message.

```
window.alert("Hello, World!");
```

window.confirm(): Affiche une boîte de dialogue de confirmation avec un message et des boutons "OK" et "Annuler".

```
// Afficher une boîte de dialogue de confirmation
// avec le message "Voulez-vous continuer ?"
let userResponse = window.confirm("Voulez-vous continuer ?");
```

window.prompt(): Affiche une boîte de dialogue demandant à l'utilisateur de saisir du texte.

```
// Afficher une boîte de dialogue de saisie avec le message "Entrez votre nom :"
let userInput = window.prompt("Entrez votre nom :");
```

WINDOW

Gestionnaires d'événements :

window peut être utilisé pour attacher des gestionnaires d'événements globaux.

```
// Ajouter un écouteur d'événements pour l'événement de redimensionnement de la fenêtre
window.addEventListener('resize', function(event) {
    // Cette fonction sera appelée chaque fois que la fenêtre est redimensionnée
    console.log('La fenêtre a été redimensionnée.');
});
```

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

WINDOW

Intervalles et temporisateurs :

window.setTimeout(): Exécute une fonction une fois après une attente spécifiée.

```
// Utiliser setTimeout pour exécuter une fonction après un délai
// de 2000 millisecondes (2 secondes)
window.setTimeout(function() {
    // Cette fonction sera exécutée après le délai spécifié
    console.log("Cette fonction s'exécute après 2000 millisecondes.");
}, 2000);
```

window.setInterval(): Exécute une fonction à intervalles réguliers.

```
// Utiliser setInterval pour exécuter une fonction
// toutes les 1000 millisecondes (1 seconde)
let intervalID = window.setInterval(function() {
    // Cette fonction sera exécutée à intervalles réguliers
    console.log("Cette fonction s'exécute chaque seconde.");
}, 1000);
```

Novao Learning est un service de formation dans le domaine du numérique.de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille

WINDOW

Intervalles et temporisateurs :

window.clearInterval(): Arrête l'exécution d'une fonction qui a été définie avec setInterval().

```
// Arrêter l'exécution de la fonction associée
// à l'intervalle avec l'identifiant intervalID
window.clearInterval(intervalID);
```

Novao Learning est un service de formation dans le domaine du numérique de © INSY2S – La manufacture, 3 Rue du Dr Huart, 59260 Lille