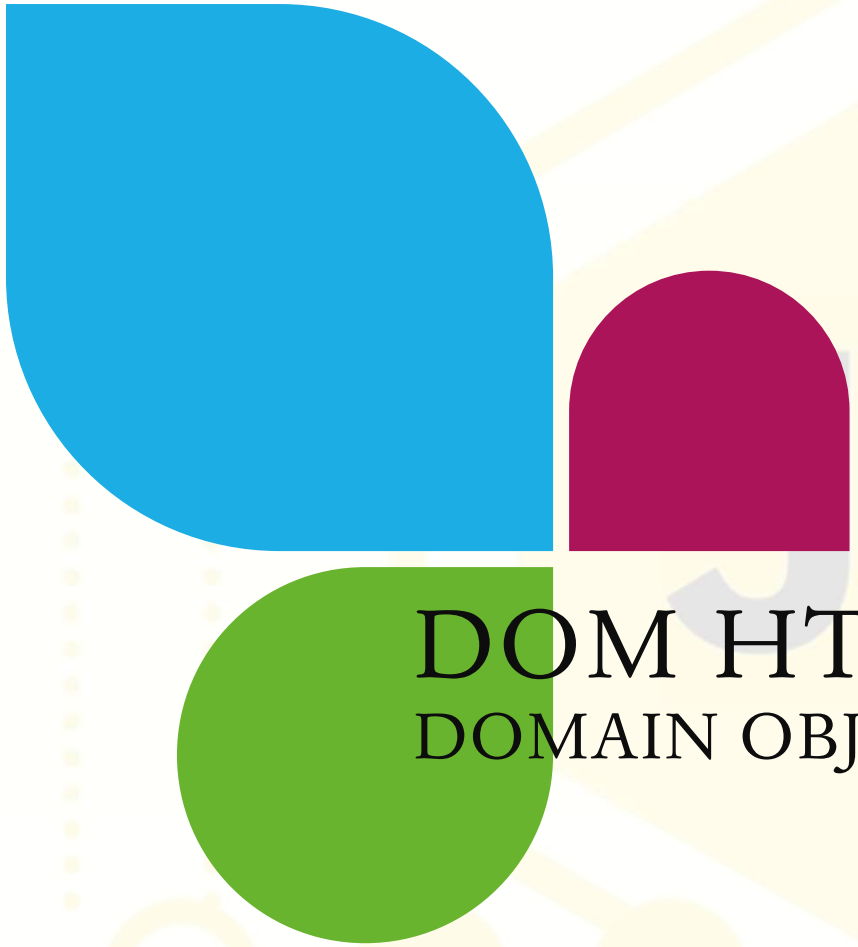


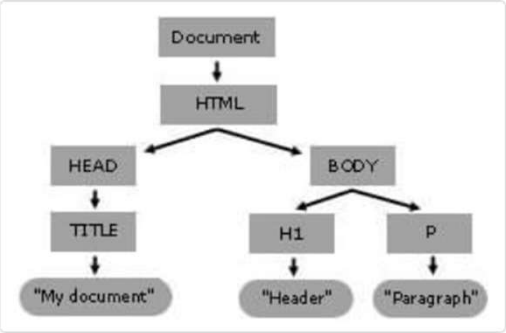
DOM HTML
DOMAIN OBJECT MODEL

DÉFINITION

Le DOM ou Document Object Model est une interface de programmation (ou API) pour les documents HTML

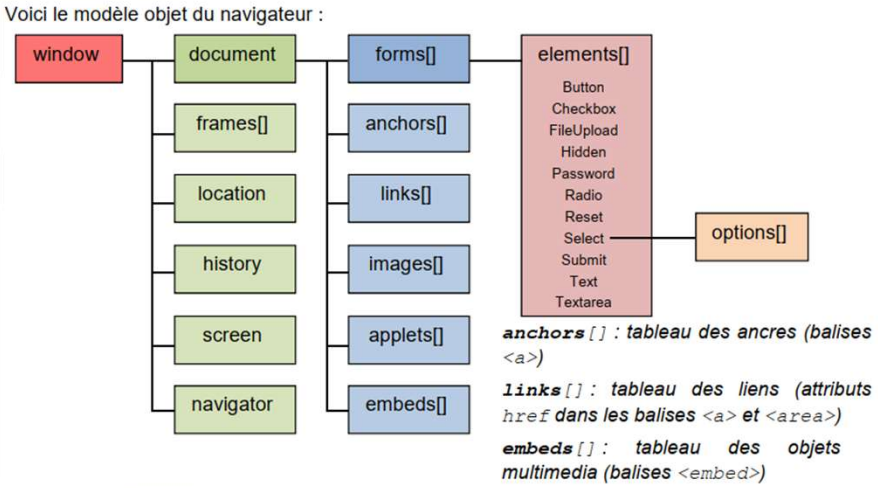
C'est donc un ensemble d'outils qui permettent de faire communiquer entre eux, dans le cas présent, les langages HTML et JavaScript.

- Standard établit par W3C



A la racine, l'objet window représente l'instance du navigateur :

- l'objet location qui symbolise la barre d'adresse,
- l'objet history qui représente l'historique des pages visitées par l'utilisateur
- l'objet document qui représente la page Web en cours, le contenu du <body> Html, lui-même référençant tous ses éléments.



L'OBJET WINDOW

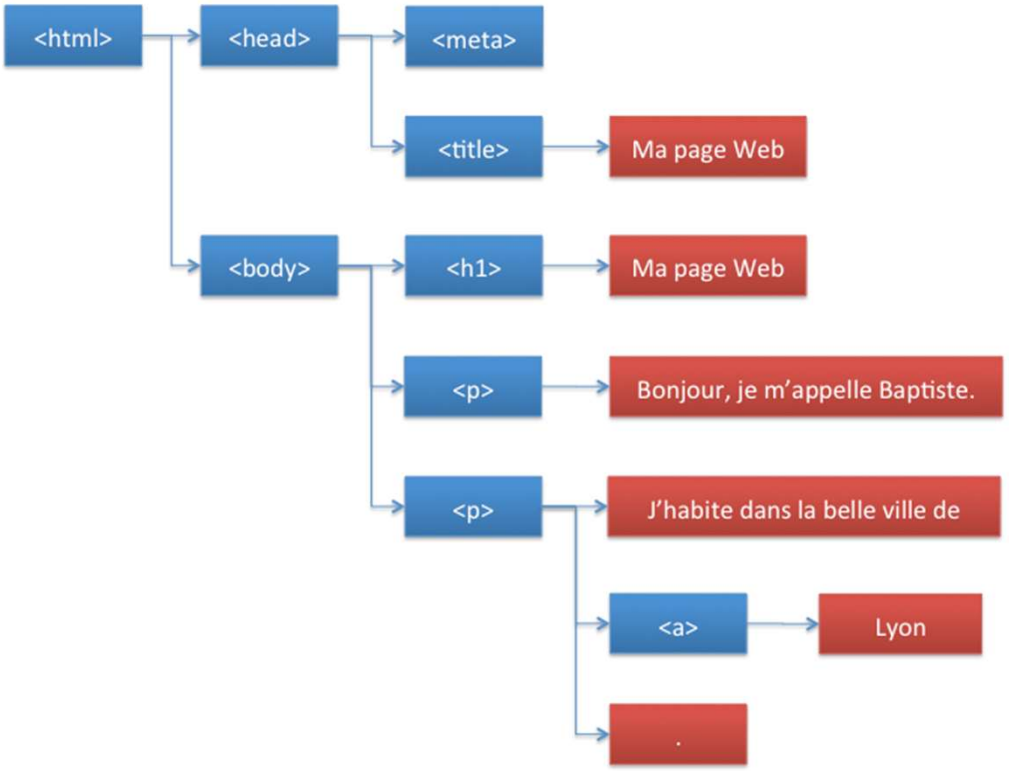
l'objet window qui représente une fenêtre contenant un document DOM

```
> console.log(window.location);
                                                                    VM162:1
  ▶ Location {ancestorOrigins: DOMStringList, href: 'http://tpform/template.html?lastName=b&firstName=b…DateStart=&periodDateEnd=&numberDay=&code01=01_01', origin: 'http://tpform', protocol: 'http:', host: 'tpform', …}
< undefined
> console.log(window.location.search);
  ?lastName=b&firstName=b&study=CDA&option=optionOne&awayDate=2022-03-07&awayTimeStart=10&awayTimeEnd=11&periodDateStart=&periodDateEnd=&numberDay=&code01=01_01   VM284:1
< undefined
> console.log(window.location.search.substring(1));
  lastName=b&firstName=b&study=CDA&option=optionOne&awayDate=2022-03-07&awayTimeStart=10&awayTimeEnd=11&periodDateStart=&periodDateEnd=&numberDay=&code01=01_01   VM367:1
```

TYPE DE NŒUD (NODE)

Dans le DOM, 2 types de nœuds :

- Ceux en bleu qui correspondent à des éléments HTML comme body, p. Ces nœuds peuvent avoir des sous-nœuds, appelés fils ou enfants (children) (nodeType = ELEMENT_NODE)
- Ceux en rouge, qui correspondent au contenu textuel de la page. Ces nœuds ne peuvent avoir de fils. (nodeType = TEXT_NODE)
- Document étant l'élément <html>



PARCOURIR LES NŒUDS

```html
<body>
    <h1>Les sept merveilles du monde</h1>
    <p>Connaissez-vous les merveilles du monde ?</p>
    <div id="contenu">
        <h2>Merveilles du monde antique</h2>
        <p>Cette liste nous vient de l'Antiquité.</p>
        <ul class="merveilles" id="antiques">
            <li class="existe">La pyramide de Khéops</li>
            <li>Les jardins suspendus de Babylone</li>
            <li>La statue de Zeus</li>
            <li>Le temple d'Artémis</li>
            <li>Le mausolée d'Halicarnasse</li>
            <li>Le Colosse de Rhodes</li>
            <li>Le phare d'Alexandrie</li>
        </ul>
        <h2>Nouvelles merveilles du monde</h2>
        <p>Cette liste a été établie en 2009 à la suite d'un vote par Internet.
        </p>
        <ul class="merveilles" id="nouvelles">
            <li class="existe">La Grande Muraille de Chine</li>
            <li class="existe">Pétra</li>
            <Li class="existe">Le Christ du Corcovado</Li>
            <li class="existe">Machu Picchu</li>
            <li class="existe">Chichén Itzá</li>
            <li class="existe">Le Colisée</li>
            <li class="existe">Le Taj Mahal</li>
        </ul>
        <h2>Références</h2>
        <ul>
            <li><a href="https://fr.wikipedia.org/wiki/
            Sept_merveilles_du_monde">Merveilles antiques</a></li>
            <li><a href="https://fr.wikipedia.org/wiki/
            Sept_nouvelles_merveilles_du_monde">Nouvelles merveilles</a></li>
        </ul>
    </div>
```

```javascript
// etape 1
var h = document.head; // La variable h contient l'objet head du DOM
console.log(h);
// etape 2
var b = document.body; // La variable b contient l'objet body du DOM
console.log(b);
// etape 3
// type de noeud
if (document.body.nodeType === document.ELEMENT_NODE) {
    console.log("Body est un noeud élément");
} else {
    console.log("Body est un noeud textuel");
}
// etape 4
// Accéder aux enfants d'un nœud élément
// Accès au premier enfant du noeud body ??
console.log(document.body.childNodes[0]);
/*
les espaces entre les balises ainsi que les retours à la ligne dans le code
HTML sont considérés par le navigateur comme des nœuds textuels. Ici, le noeud
h1 n'est donc que le deuxième enfant du nœud body.
*/
console.log(document.body.childNodes[1]);
// parcours des noeuds enfants
// Affiche les noeuds enfant du noeud body
for (let i = 0; i < document.body.childNodes.length; i++) {
    console.log(document.body.childNodes[i]);
}
// Accéder au parent d'un nœud
var h1 = document.body.childNodes[1];
console.log(h1.parentNode); // Affiche le noeud body

console.log(document.parentNode); // Affiche null : document n'a aucun noeud
parent
```

ACCÉDER AU DOCUMENT

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

Tout commence avec le document qui représente la page entière.

```
let element = document.getElementById(id);

let elements = document.getElementsByClassName(names); // ou:
elements = rootElement.getElementsByClassName(names);

elements = document.getElementsByTagName(nom);

elements = document.getElementsByName(nom)

let el = document.querySelector(".maclasse");

let matches = myBox.querySelectorAll("p");
```

document.getElementById()

- Retrouver un élément précis par son id

document.getElementsByClassName()

- Retrouver tous les éléments par la classe

document.getElementsByTagName()

- Retrouver tous les éléments avec un nom de balise

document.getElementsByName()

- Retrouver tous les éléments portant un nom donné dans le document HTML

document.querySelector()

- Retrouver le 1er élément correspondant au sélecteur CSS ou groupe de sélecteurs CSS

document.querySelectorAll()

- Retrouver tous les éléments correspondant au sélecteur CSS ou groupe de sélecteurs CSS

RECHERCHE DEPUIS UN ÉLÉMENT

Il n'y a pas qu'avec document que vous pouvez rechercher des éléments.

Comme nous l'avons vu, chaque élément est un objet JavaScript avec ses propriétés et ses fonctions.

Et parmi ces dernières, il en existe pour parcourir les enfants et le parent de chaque élément !

element.children

- Retourne une liste d'enfant de l'élément

element.parentElement

- Retourne l'élément parent

element.nextElementSibling et element.previousElementSibling

- Permet de naviguer vers l'élément suivant / précédent de même niveau

```
<div id="parent">
    <div id="previous">Précédent</div>
    <div id="main">
        <p>Paragraphe 1</p>
        <p>Paragraphe 2</p>
    </div>
    <div id="next">Suivant</div>
</div>
```

```
const elt = document.getElementById('main');
```

elt.children = les éléments de type p enfant de #main
elt.parentElement = div qui à l'id parent
elt.nextElementSibling = next
elt.previousElementSibling = previous

MODIFIER LE DOM

Deux propriétés principales :

- InnerHTML
  - Récupère ou définit la contenu HTML d'un élément du DOM
- textContent
  - Récupère son contenu textuel sans le balisage HTML du DOM

- Les attributs et les classes :
  - Modifier des classes d'un élément :
    - Possible d'accéder à la liste des classes d'un élément avec la propriété classList
      - add(string) : ajoute une classe
      - remove(string) : supprime la classe
      - contains(string) : vérifie si la classe existe
      - replace(old, new) : remplace l'ancienne classe par la nouvelle
  - Changer les styles d'un élément :
    - Possible d'accéder au style avec la propriété style
    - element.style.backgroundColor = '#000';
  - Modifier les attributs d'un élément :
    - Définir ou remplacer les attributs avec la fonction setAttribute, getAttribute, removeAttribute

MODIFIER LE DOM

On peut également :

1. Créer de nouveaux éléments.
2. Ajouter des enfants.
3. Supprimer et remplacer des éléments.

- La fonction createElement permet de créer de nouveaux éléments.
- Plusieurs façons d'ajouter un élément :
  - appendChild
- Les fonctions removeChild et replaceChild permettent de supprimer ou remplacer un élément

```
const newElt = document.createElement("div");
let elt = document.getElementById("main");
elt.appendChild(newElt);

elt.removeChild(newElt);    // Supprime l'élément newElt de l'élément elt
elt.replaceChild(document.createElement("article"), newElt);    // Remplace l'élément newElt
par un nouvel élément de type article
```

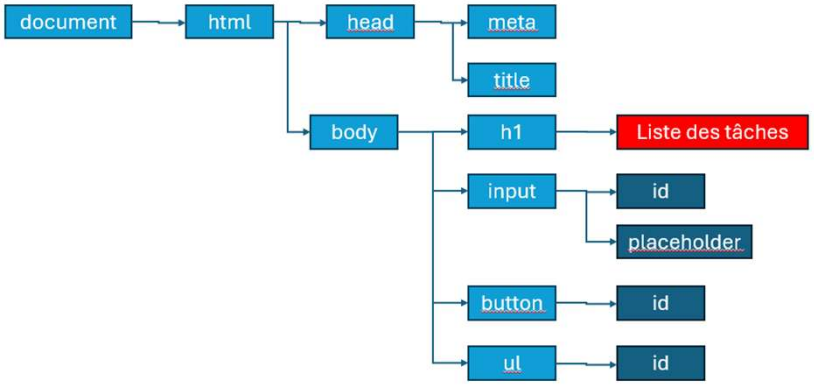
```
const newElt = document.createElement("div");
let elt = document.getElementById("main");

elt.appendChild(newElt);
```

```
const newElt = document.createElement("div");
```

EXEMPLE

Ci contre, un fichier HTML pour lister des tâches que l'utilisateur saisie.



dom.html > ...

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Liste de Tâches</title>
    <!-- css -->
    <link rel="stylesheet" href="./CSS/swapimeteo.css">
</head>
<body>
    <h1>Liste de Tâches</h1>
    <input type="text" id="nouvelleTache" placeholder="Ajouter une tâche">
    <button id="ajouter">Ajouter</button>
    <ul id="listeTaches"></ul>

    <script type="text/javascript" src="./scriptJS/dom.js"></script>
</body>
</html>
```

EXEMPLE

Le script associé au fichier précédemment va construire DOM en fonction de la saisie utilisateur en créant les nœuds li et les ajouter au nœud ul


```javascript
// script JS de manipulation du DOM
function ajouterTache() {

    // recuperation de la valeur de l'element dont l'id est nouvelleTache dans le DOM
    // utilisation de trim pour enlever les espaces vide
    let tache = document.getElementById("nouvelleTache").value.trim() ;
    // test si aucune saisie
    if (tache !== "") {
        // récuperation de l'element dont l'id est listeTaches dans le DOM
        let liste = document.getElementById("listeTaches");
        // création d'un element li
        let element = document.createElement("li");
        // ajout du texte du li
        element.textContent = tache;

        // ajout d'un evenement sur l'element li
        element.onclick = function () {
            // suppression de l'element
            liste.removeChild(element);
        };

        // ajout de l'element li au parent
        liste.appendChild(element);
        // reset du champs input
        document.getElementById("nouvelleTache").value = "";
    } else {
        // affichage d'une alerte de saisie
        alert("Merci de saisir une tâche !!");
    }
}

// au chargement de la page HTML
document.addEventListener('DOMContentLoaded', function () {
    // ajout d'un evenement sur le bouton sur click
    document.getElementById("ajouter").addEventListener('click', ajouterTache);
});
```

LES ÉVÈNEMENTS

ACTIONS !!

ÉCOUTER LES ÉVÈNEMENTS

- Réaction à une action émise par l'utilisateur comme le clic, la saisie etc...
- Un événement en JavaScript est représenté par un nom (click , mousemove ...) et une fonction que l'on nomme un callback
- Par défaut, un événement est propagé, transmis à l'élément parent jusqu'à la racine du document tant qu'on ne l'a pas traité.
- La fonction de callback est appelée à chaque fois que l'action est désirée

AddEventListener permet d'écouter tous type d'évenements

- PreventDefault() est une option permettant de supprimer le comportement par défaut de l'événement
- stopPropagation() permet d'empêcher la propagation de l'événement au parent

```
const elt = document.getElementById('mon-lien');     // On récupère l'élément sur lequel on veut détecter le clic
elt.addEventListener('click', function() {            // On écoute l'événement click
    elt.innerHTML = "C'est cliqué !";                 // On change le contenu de notre élément pour afficher "C'est cliqué !"
});
```

```
const elt = document.getElementById('mon-lien');     // On récupère l'élément sur lequel on veut détecter le clic
elt.addEventListener('click', function(event) {       // On écoute l'événement click, notre callback prend un paramètre que nous avons appelé event ici
    event.preventDefault();                           // On utilise la fonction preventDefault de notre objet event pour empêcher le comportement par défaut de cet élément lors du clic de la souris
});
```

LA PROPAGATION DES ÉVÉNEMENTS

Le DOM représente une page web sous la forme d'une hiérarchie de nœuds.

Les événements déclenchés sur un nœud enfant vont se déclencher ensuite sur son nœud parent, puis sur le parent de celui-ci, et ce jusqu'à la racine du DOM (la variable document).

C'est ce qu'on appelle la propagation des événements.

```html
<body>
    <button id="bouton">Cliquez-moi !</button>

    <p id="para">Un paragraphe avec un
        <button id="propa">bouton</button> à l'intérieur
    </p>


    <script src="ressources/event.js"></script>
</body>
```

```js
// Gestion du clic sur le document
document.addEventListener("click", function () {
    console.log("Gestionnaire document");
});
// Gestion du clic sur le paragraphe
document.getElementById("para").addEventListener("click", function () {
    console.log("Gestionnaire paragraphe");
});
// Gestion du clic sur le bouton
document.getElementById("propa").addEventListener("click", function (e) {
    console.log("Gestionnaire bouton");
});
```

LA PROPAGATION DES ÉVÉNEMENTS

Stopper la propagation

```
// Gestion du clic sur le bouton
document.getElementById("propa").addEventListener("click", function (e) {
    console.log("Gestionnaire bouton");
    e.stopPropagation(); // Arrêt de la propagation de l'événement
});
```

Modifier le comportement par défaut

```
<p>Du temps à perdre ? <a id="interdit" href="https://9gag.com/">Cliqu
ici</a></p>
```

```
// Gestion du clic sur le lien interdit
document.getElementById("interdit").addEventListener("click", function (e)
    console.log("Continuez plutôt à lire le cours ;)");
    e.preventDefault(); // Annulation de la navigation vers la cible du lie
});
```

TYPE D'ÉVÉNEMENTS

Les événements que les éléments du DOM peuvent déclencher sont très nombreux.

Le tableau ci-contre présente les principales catégories d'événements.

Quel que soit le type d'événement, son déclenchement s'accompagne de la création d'un objet Event

Catégorie	Exemples
Événements clavier	Appui ou relâchement d'une touche du clavier
Événements souris	Clic avec les différents boutons, appui ou relâchement d'un bouton de la souris, survol d'une zone avec la souris
Événements fenêtre	Chargement ou fermeture de la page, redimensionnement, défilement *(scrolling)*
Événements formulaire	Changement de cible de saisie (focus), envoi d'un formulaire

```
// Ajout d'un gestionnaire qui affiche le type et la cible de l'événement
document.getElementById("bouton").addEventListener("click", function (e) {
    console.log("Evènement : " + e.type +
        ", texte de la cible : " + e.target.textContent);
});
```

keyup - keydow - keypress

clic - mousemove

submit - reset

load - beforeunload

LES ÉVÈNEMENTS

La souris

Avec mousemove, on peut détecter le mouvement de la souris.

Cet évènement fournit un objet MouseEvent qui permet de récupérer

- clientX / clientY : position de la souris
- offsetX / offsetY : position par rapport à l'élément
- pageX / pageY position par rapport au document
- screenX / screenY position par rapport à la fenêtre
- movementX / movementY position par rapport à la position lors du dernier évènement mousemove

Lire un champ texte

L'évènement change pour les éléments input, select, textarea, checkbox, radio est déclenché lorsque le champ perd le focus

L'évènement input permet de connaitre les modifications d'un élément en cours par l'utilisateur.

```
<label>Choose an ice cream flavor:
    <select id="ice-cream" name="ice-cream">
        <option value="">Select One …</option>
        <option value="chocolate">Chocolate</option>
        <option value="strawberry">Strawberry</option>
        <option value="vanilla">Vanilla</option>
    </select>
</label>
```

```
document.addEventListener('DOMContentLoaded',function() {
    document.querySelector('select[name="ice-cream"]').onchange=changeEventHandler;
},false);

function changeEventHandler(event) {
    // You can use “this” to refer to the selected element.
    if(!event.target.value) alert('Please Select One');
    else alert('You like ' + event.target.value + ' ice cream.');
}
```

EXEMPLE

On peut aussi appliquer l'événement directement dans le html

```html
<!DOCTYPE html>
<html lang="fr">

    <head>

        <!-- ENCODAGE -->
        <meta charset="UTF-8">
        <!-- description du site -->
        <meta name="description" content="Cours HTML">
        <!-- prise en charge du contexte mobile -->
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- titre de la page -->
        <title>Document</title>
        <link rel="stylesheet" type="text/css" href="ressources/style/style.
        css">

    <body>

        <h1>TEST JS</h1>

        <input type="button" value="Change ce document." onclick="change()">
        <h2>H2</h2>
        <p>Paragraphe</p>


        <script StyleSheet="text/javascript" src="/ressources/script/script.
        js"></script>

    </body>


</html>
```

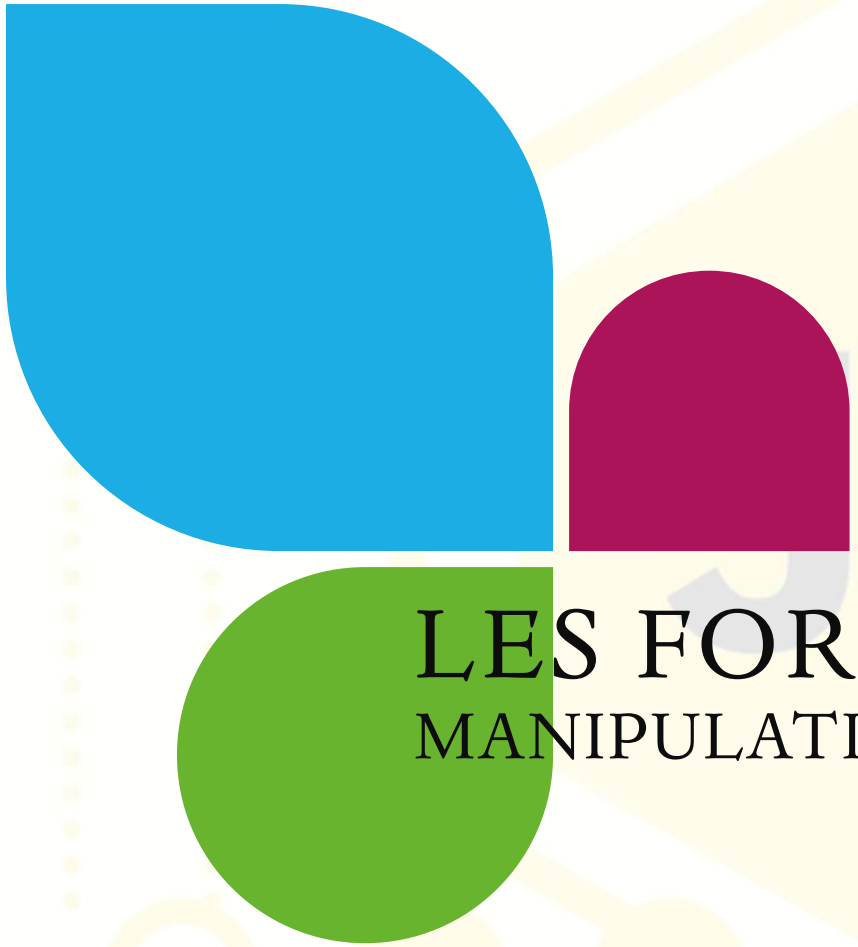
```javascript
function change() {

    // document.getElementsByTagName ("H2") renvoie un NodeList du <h2>
    // éléments dans le document, et le premier est le nombre 0:
    let header = document.getElementsByTagName("H2").item(0);
    // le firstChild de l'en-tête est un noeud texte::
    header.firstChild.data = "A dynamic document";
    // maintenant l'en-tête est "Un document dynamique".

    let para = document.getElementsByTagName("P").item(0);
    para.firstChild.data = "This is the first paragraph.";

    // crée un nouveau noeud texte pour le second paragraphe
    let newText = document.createTextNode("This is the second paragraph.");
    // crée un nouvel Element pour le second paragraphe
    let newElement = document.createElement("P");
    // pose le texte dans le paragraphe
    newElement.appendChild(newText);
    // et pose le paragraphe à la fin du document en l'ajoutant
    // au BODY (qui est le parent de para)
    para.parentNode.appendChild(newElement);
}
```

Afpa



LES FORMULAIRES
MANIPULATION ET INTERACTION

LES FORMULAIRES

Zone de texte - input et textarea

- required : oblige la saisie
- value : valeur de la zone de texte
- gestion du focus :
  - focus : saisie en cours dans la zone de texte
  - blur : changement de cible provoque un blur sur l'ancienne zone de texte qui avait le focus

```
// Affichage d'un message contextuel pour la saisie du pseudo
pseudoElt.addEventListener("focus", function () {
    document.getElementById("aidePseudo").textContent = "Entrez votre pseud
});
// Suppression du message contextuel pour la saisie du pseudo
pseudoElt.addEventListener("blur", function (e) {
    document.getElementById("aidePseudo").textContent = "";
});
```

Les cases à cocher et boutons radios

- change indique lorsque l'utilisateur modifie son choix.
- Case à cocher :
  - Event dispose d'une propriété checked (cochée / décochée)
- Boutons radio :
  - Balise input de type radio avec l'attribut name et value

```
// Affichage de la demande de confirmation d'inscription
document.getElementById("confirmation").addEventListener("change", function (e)
{
    console.log("Demande de confirmation : " + e.target.checked);
});
```

LES FORMULAIRES

Liste déroulante

- balise select + option
- change est déclenché sur les modifications apportées à la liste.

Comme pour les boutons radio, la propriété e.target.value de l'événement change contient la valeur de l'attribut value de la balise option associé au nouveau choix, et non pas le texte affiché dans la liste déroulante.

Formulaire et DOM

La balise form est l'élément à cibler pour accéder au contenu du formulaire par l'attribut elements rassemblant les champs de saisie

- soumission du formulaire
  - input de type submit et input de type reset
  - C'est à partir de cette soumission que nous allons tester et contrôler la saisie de l'utilisateur et en cas d'erreur utiliser preventDefault

CONTRÔLE DE LA SAISIE

CONTRÔLER L'UTILISATEUR

SUR LES ÉLÉMENTS DU FORMULAIRE

Vous avez aussi la possibilité de mettre en place des patterns sur la balise input afin de mettre en place un contrôle sur la saisie à la validation.

L'attribut pattern peut être utilisé pour les champs de type text, tel, email, url, password, search.

Les REGEX vont nous aider à mettre en place des contrôles sur la saisie.

```
<p>
    <label for="mdp">Mot de passe</label> :
    <input type="password" name="mdp" id="mdp"
    pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
    title="doit contenir au moins 1 chiffre, 1 majuscule, 1 minuscule
    et au moins 8 caractères" required>
    <span id="aideMdp"></span>
</p>
```

SUR LES DIFFÉRENTES PHASES DU FORMULAIRE

Le contrôle de validité peut se faire de plusieurs manières, éventuellement combinables :

- Soit au fur et à mesure de la saisie d'une donnée
  - input
- Soit à la fin de la saisie d'une donnée
  - blur
- Soit au moment où l'utilisateur soumet le formulaire.
  - submit

```
// Vérification de la longueur du mot de passe saisi
document.getElementById("mdp").addEventListener("input", function (e) {
    var mdp = e.target.value; // Valeur saisie dans le champ mdp
    var longueurMdp = "faible";
    var couleurMsg = "red"; // Longueur faible => couleur rouge
    if (mdp.length >= 8) {
        longueurMdp = "suffisante";
        couleurMsg = "green"; // Longueur suffisante => couleur verte
    } else if (mdp.length >= 4) {
        longueurMdp = "moyenne";
        couleurMsg = "orange"; // Longueur moyenne => couleur orange
    }
    var aideMdpElt = document.getElementById("aideMdp");
    aideMdpElt.textContent = "Longueur : " + longueurMdp; // Texte de l'aide
    aideMdpElt.style.color = couleurMsg; // Couleur du texte de l'aide
});

// Contrôle du courriel en fin de saisie
document.getElementById("courriel").addEventListener("blur", function (e) {
    var validiteCourriel = "";
    if (e.target.value.indexOf("@") === -1) {
        // Le courriel saisi ne contient pas le caractère @
        validiteCourriel = "Adresse invalide";
    }
    document.getElementById("aideCourriel").textContent = validiteCourriel;
});
```

EXEMPLE AVEC JAVASCRIPT

On peut également écrire nos fonctions de contrôle à la validation afin d'être plus précis sur nos contrôles.

```
// méthode sans REGEX
// Contrôle du courriel en fin de saisie
document.getElementById("courriel").addEventListener("blur", function (e) {
    var validiteCourriel = "";
    if (e.target.value.indexOf("@") === -1) {
        // Le courriel saisi ne contient pas le caractère @
        validiteCourriel = "Adresse invalide";
    }
    document.getElementById("aideCourriel").textContent = validiteCourriel;
});

// méthode avec REGEX
// Contrôle du courriel en fin de saisie
document.getElementById("courriel").addEventListener("blur", function (e) {
    // Correspond à une chaîne de la forme xxx@yyy.zzz
    var regexCourriel = /.+@.+\..+/;
    var validiteCourriel = "";
    if (!regexCourriel.test(e.target.value)) {
        validiteCourriel = "Adresse invalide";
    }
    document.getElementById("aideCourriel").textContent = validiteCourriel;
});
```