MEMENTO JAVASCRIPT

DECOUVRIR LE DOM	
document.head	Renvoie le contenu du head
document.body	Renvoie le contenu du body
elt.**nodeType** === **document.ELEMENT_NODE**	True si elt est un nœud de type élément
elt.**nodeType** === **document.TEXT_NODE**	True si elt est un nœud de type textuel
elt.**childNodes** ;	Fait références aux nœuds enfants d'un éléments. Attention ! childNodes n'est pas un tableau !
var nœuds = elt.childNodes ; var items = **[].slice.call(noeuds);**	**Astuce** pour transformer la liste de nœuds en tableau
var total = elt.**childNodes.length** ;	Renvoie le nombre de nœuds enfants d'un élément
var elt = document.body.**childNodes**[0] ;	Renvoie le premier enfant du nœud body **Attention !** les espaces et retours chariot comptent comme des nœuds (nœuds textuels)
var elt = document.getElementById("titre") ; console.log(elt.**parentNode**) ;	Renvoie le nœud parent d'un élément
PARCOURIR LE DOM	
var h2 = document.**getElementsByTagName**("h2"); var nb = h2.length ;	Récupère tous les titres h2 Récupère le nombre de titre h2
var merveilles = document.**getElementsByClassName**("merveille) ;	Récupère les éléments possédant la classe "merveille"
var elt = document.**getElementById**("nouvelles") ;	Récupère l'élément ayant l'identifiant "nouvelles"
var elts = document.**querySelectorAll**("p") ; var elts = document.**querySelectorAll**("#p .visible") ; var elts = document .**querySelectorAll**("#antiques > .existe") ;	Rechercher à partir d'un sélecteur
var elt = document.**querySelector**("h2) ;	Renvoie le premier titre h2
var html = elt.**innerHTML** ;	Récupère le contenu html d'un élément
elt.**innerHTML** = " mon texte. " ;	Affecte du contenu html à un élément
var texte = elt.**textContent** ;	Récupère le contenu textuel d'un éléments
elt.**textContent** = "mon texte" ; elt.**textContent +=** " à lire à tout prix." ;	Affecte du contenu textuel à un élément
var href = elt.**getAttribute**("href")) ;	Récupère le contenu de l'attribut href
elt.**setAttribute**("href", "http://google.com");	Affecte un lien à l'attribut href
var id = elt.**id** ; var href = elt.**href** ; var value = elt.**value** ; elt.**id** = "conteneur" ; elt.**href** = "http://google.com" elt.**value** = "5" ;	Certains attributs (id, href, value) sont accessibles/modifiables directement
var bool = elt.**hasAttribute**("target") ;	Renvoie true si l'élément possède l'attribut "target"
btn.**disabled** = true ;	Active et désactive le bouton
var classes = elt.**classList** ;	Renvoie, sous forme de tableau, la liste des classes d'un élément
var bool = document.getElementById("antiques") .classList.**contains**("merveille") ;	Renvoie true si l'élément "antiques" contient la classe "merveille"
MANIPULER LE DOM	
var classes = elt.**classList** ; elt.clIList.**add**("cordes") ; elt.clIList.**add**("cordes", "frottees") ;	classList permet d'accéder à la liste des classes Ajout d'une classe Ajout de plusieurs classes

elt.clllList.**remove**("cordes") ; elt.clllList.**remove**("cordes", "frottees") ;	Suppression d'une classe Suppression de plusieurs classes
var ul = document.**getElementById**("langages") ; var li = document.**createElement**("li"); li**.id** = "python"; li**.textContent** = "Python"; ul.**appendChild**(li);	Récupérer l'élément "ul" id = "langages" Créer un nouvel élément "li" Lui affecter un id Lui affecter un contenu textuel **Ajouter l'élément** li à la fin de l'élément ul

var ul = document.**getElementById**("langages") ;
var li = document.**createElement**("li");
li.**id** = "C++";
li.appendChild(document.**createTextNode**("C++"));
ul.**appendChild**(li);
	Idem que précédemment en insérant en plus un nœud textuel

var ul = document.**getElementById**("langages") ; var phpLi = document.**getElementById**("php") ; var perlLi = document.**createElement**("li"); perlLi.**id** = "perl"; perlLi.**textContent** = "Perl"; ul.**insertBefore**(perlLi, phpLi);	**Insérer** un nœud (Perl) avant un autre (PHP)
var ul = document.**getElementById**("langages") ; var perlLi = document.**getElementById**("perl") ; var javaLi = document.**createElement**("li"); javaLi.**id** = "java"; javaLi.**textContent** = "Java" ; ul.**replaceChild**(javaLi, perlLI) ;	**Remplacer** un noeud par un autre
var ul = document.**getElementById**("langages") ; var perLi = document.**getElementById**("perl") ; ul.**removeChild**(perlLi);	**Supprimer** un noeud
var ul = document.**getElementById**("langages") ; var contenu = '<li id="javascript">JavaScript ' ; ul.**insertAdjacentHTML**("**afterbegin**", contenu);	InsertAdjacent insère du contenu HTML à une position exacte : **beforebegin** : avant l'élément lui-même. **afterbegin** : juste à l'intérieur de l'élément, avant son premier enfant. **beforeend** : juste à l'intérieur de l'élément, après son dernier enfant. **afterend** : après l'élément lui-même.
GERER LES STYLES	
var p = document.querySelector("p") ; p.**style.color** = "red" ; p.**style.marginLeft** = "50px" ;	Récupérer le premier paragraphe Changer la couleur Changer margin-left (on supprime le tiret)
var p = document.querySelector("p") ; var color = p.**style.color** ;	Récupérer la valeur d'une propriété de style **IMPORTANT** ! Seules les propriétés définies à même la ligne, ou par programmation, sont récupérables de cette façon. **getComputedStyle**(elt) est préférable pour récupérer l'intégralité du style d'un élément.
var p = document.querySelector("p") ; var style = **getComputedStyle**(p); var color = style.color ;	Permet de récupérer un objet représentant le style effectif de l'élément.
GERER LES EVENEMENTS	

var btn = document.getElementById("bouton");
btn.**addEventListener**("click", function () {
  console.log("Clic !");
});
	Ajoute un événement "click" sur le bouton "#bouton" et l'oriente vers une **fonction anonyme.**

function clicDuBouton() {
  console.log("Clic !");
}
 	
var btn = document.getElementById("bouton");
btn.**addEventListener**("click", clicDuBouton) ;
	Ajoute un événement "click" sur le bouton "#bouton" et l'oriente vers une **fonction prédéfinie**.

document.**querySelector("button")** .addEventListener("click", function(e) {	Définit un événement "click" pour tous les boutons

}) ;	
btn.**removeEventListener**("click", nomFonction) ;	Supprime la gestion de l'événement. Ne marche pas avec les fonctions anonymes.

var btn = document.getElementById("bouton") ;
btn.addEventListener("click", function (**e**) {
  var type : **e.type** ;
  var target = **e.target** ;
  var texteBtn = target.textContent ;
});
	Récupère le type de l'événement et la cible (élément déclencheur).

var btn = document.getElementById("bouton") ;
btn.addEventListener("click", function (**e**) {
  var type : **e.type** ;
  var target = **e.target** ;
  var bouton = **e.button** ;
  var posx = **e.clientX** ;
  var posx = **e.clientY** ;
});
	Propriétés de l'objet renvoyé par l'événement "click" :
	
	Type d'événement (ici "click")
	Cible (le bouton)
	Bouton de la souris (0=gauche, 1=milieu, 2=droit)
	Position X
	Position Y

Principales catégories d'événements :

Catégorie	Exemples
Evénements 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

document.addEventListener ("**keypress**", function (**e**) {
  var touche = String.fromCharCode(**e.charCode**) ;
  console.log("Touche " + touche) ;
});
	Gestion de l'appui sur une touche du clavier produisant un caractère
	
	Ordre de déclenchement des événements clavier :
	keydown → **keypress** → **keyup**

window.addEventListener("**load**", function () { console.log("Page entièrement chargée"); });	Événement déclenché lorsque la page est totalement chargée.

window.addEventListener("**beforeunload**", function (**e**) {
  var message = "On est bien ici !";
  e.returnValue = message;
  return message;
});
	Evénement déclenché avant la fermeture de la fenêtre.
	
	Note : le message de confirmation ne s'affiche que sur certains navigateurs.

var btn document.getElementById("propa") ;
btn.addEventListener("click", function (**e**) {
  console.log("Gestionnaire bouton");
  e.stopPropagation();
});
	Empêche l'événement de se propager aux éléments parents. : div,…, document.

var a = document.getElementById("interdit") ;
a.addEventListener("click", function (**e**) {
  console.log("Continuez plutôt à lire le cours ;)");
  e.preventDefault() ;
});
	Empêche l'événement associé par défaut à un élément (ici un lien <a>) de se déclencher

MANIPULER LES FORMULAIRES	
var pseudoElt = document.getElementById("pseudo"); pseudoElt.**value** = "MonPseudo";	Affecte une valeur à un champ
var valeur = document.getElementById("pseudo").**value** ;	Récupère la valeur d'un champ
pseudoElt.**focus**() ; pseudoElt.**blur**() ;	Donne le focus à pseudoElt Enlève le focus à pseudoElt
pseudoElt.addEventListener("**focus**", maFonction) ; pseudoElt.addEventListener("**blur**", maFonction) ;	Gestion des événements "focus" et "blur"
caseChk.addEventListener("**change**", maFonction) ;	Gestion de l'événement "change"

pseudoElt..addEventListener("**input**", function(e) { … }) ;	Gestion de l'événement "input" (insertion de texte)
form.addEventListener("**submit**", function(e) { … }) ;	Gestion de l'événement "submit"
var form = document.querySelector("form"); var nbChamps = form.elements.length ; var nom = form.elements[0].name ; var type = form.elements.nomChamp.type	 Nombre de champs du formulaire Nom de l'élément [0] Type de l'élément ayant pour name nomChamp
ANIMER LES PAGES	
var intervalID = setInterval(maFonction, 1000);	Appelle une fonction à intervalle régulier (millisecondes)
clearInterval(intervalID) ;	Stoppe setInterval()
setTimeout(maFonction, 2000) ;	Exécute une fonction après un délai.
var animeID = requestAnimationFrame(monAnimation);	Demande au navigation d'optimiser le lancement de l'animation afin de la rendre fluide.
cancelAnimationFrame(animeID);	Annule l'animation
INTERROGER UN SERVEUR	

Requête simple en mode synchrone :

```
// Création d'une requête http
var req = new XMLHttpRequest();
// Configurer la requête : Méthode (GET/POST/PUT), Url cible,  Asynchrone (true/false)
req.open("GET", "http://localhost/javascript-web-srv/data/langages.txt", false);
// Envoi (paramètre = null si méthode GET)
req.send(null) ;
// Réponse reçue sous forme textuelle
var reponse = req.responseText ;
```

Fonction générique d'une requête asynchrone en GET avec gestion des erreurs :

```
// Exécute un appel AJAX GET
// Prend en paramètres l'URL cible et la fonction callback appelée en cas de succès
function ajaxGet(url, callback) {
    var req = new XMLHttpRequest();
    req.open("GET", url);
    req.addEventListener("load", function () {
        if (req.status >= 200 && req.status < 400) {
            // Appelle la fonction callback en lui passant la réponse de la requête
            callback(req.responseText);
        }
        else {
            console.error(req.status + " " + req.statusText + " " + url);
        }
    });
    req.addEventListener("error", function () {
        console.error("Erreur réseau avec l'URL " + url);
    });
    req.send(null);
}
```

Utilisation de la fonction :

```
ajaxGet("http://monserveur.fr/fichier.txt", function(reponse) {
    // Gestion de la réponse
})
```

var avion = {
marque: "Airbus",
couleur: "A320"
};
var texteAvion = **JSON.stringify(**avion);
	Transforme un tableau Javascript en chaîne de caractères JSON

var avion = **JSON.parse**(texteAvion);	Transforme une chaîne de caractères JSON en tableau Javascript

Fonction générique d'une requête asynchrone en POST avec FormData :

```
// Exécute un appel AJAX POST
// Prend en paramètres l'URL cible, la donnée à envoyer et la fonction callback appelée en cas de succès
```

```
function ajaxPost(url, data, callback) {
    var req = new XMLHttpRequest();
    req.open("POST", url);
    req.addEventListener("load", function () {
        if (req.status >= 200 && req.status < 400) {
            // Appelle la fonction callback en lui passant la réponse de la requête
            callback(req.responseText);
        } else {
            console.error(req.status + " " + req.statusText + " " + url);
        }
    });
    req.addEventListener("error", function () {
        console.error("Erreur réseau avec l'URL " + url);
    });
    req.send(data);
}
```

Utilisation de la fonction:

```
var datas = new FormData() ;
datas.append("couleur", "rouge");
datas.append("pointure", "43");
// Envoi de l'objet FormData au serveur
ajaxPost("http://localhost/javascript-web-srv/post_form.php", datas,
    function (reponse) {
        // Affichage dans la console en cas de succès
        console.log("Commande envoyée au serveur");
    }
);
```

Envoi des données d'un formulaire :

```
var form = dorcument.querySelector("form");
// Gestion de la soumission du formulaire
form.addEventListener("submit", function (e) {
    e.preventDefault();
    // Récupération des champs du formulaire dans l'objet FormData
    var datas = new FormData(form);
    // Envoi des données du formulaire au serveur
    // La fonction callback est ici vide
    ajaxPost("http://localhost/javascript-web-srv/post_form.php", datas, function () {});
});
```

Fonction générique d'une requête asynchrone en POST adressant des données FormData ou Json :

```
// Exécute un appel AJAX POST
// Prend en paramètres l'URL cible, la donnée à envoyer et la fonction callback appelée en cas de succès
// Le paramètre isJson (optionnel) permet d'indiquer si l'envoi concerne des données JSON
function ajaxPost(url, data, callback, isJson) {
    var req = new XMLHttpRequest();
    req.open("POST", url);
    req.addEventListener("load", function () {
        if (req.status >= 200 && req.status < 400) {
            // Appelle la fonction callback en lui passant la réponse de la requête
            callback(req.responseText);
        } else {
            console.error(req.status + " " + req.statusText + " " + url);
        }
    });
    req.addEventListener("error", function () {
        console.error("Erreur réseau avec l'URL " + url);
    });
    if (isJson) {
        // Définit le contenu de la requête comme étant du JSON
        req.setRequestHeader("Content-Type", "application/json");
        // Transforme la donnée du format JSON vers le format texte avant l'envoi
        data = JSON.stringify(data);
    }
    req.send(data);
}
```

Utilisation de la fonction :

```
// Création d'un objet représentant un film
var film = {
    titre: "Zootopie",
    annee: "2016",
    realisateur: "Byron Howard et Rich Moore"
};

// Envoi de l'objet au serveur
ajaxPost("http://localhost/javascript-web-srv/post_json.php", film,
    function (reponse) {
        // Le film est affiché dans la console en cas de succès
        console.log("Le film " + JSON.stringify(film) + " a été envoyé au serveur");
    },
    true // Valeur du paramètre isJson
);
```