## JavaScript

Utiliser la console **ou** les fonctions natives suivantes :

* window.prompt();
* window.alert();
* window.confirm();

**Exercice : jeu du pendu**

Info : le nombre d'essais est illimité

* Créer un tableau pour contenir les lettres du mot : 'B','O','N','J','O','U','R',
* Créer un autre tableau pour mettre les lettres devinées : '*','*','*','*','*','*','\_'
* Écrivez une fonction appelée "guessLetter" cette fonction va
  + Itérer à travers les lettres
  + Voir si la lettre deviné se trouve dans le mot
  + Si oui changer le tableau des lettres deviné pour ajouter la nouvelle lettre
  + Afficher dans la console les lettre devinés
  + Le jeu doit savoir quand la partie est finie et féliciter le joueur
* Bonus: Se rapprocher du vrais pendu :
  + Gardez en mémoire toutes les lettres devinées (bonnes et mauvaises) et seulement laisser l'utilisateur deviner une lettre une fois.
  + Si une lettre est entrée deux fois, ne rien faire.
  + compter le nombre de tentatives ratés

## JavaScript

let table = new Object(); // ancienne manière de déclarer

let table = {}; // nouvelle manière de déclarer

let main\_color = "#FF0000";

let chair = {

color : main\_color,

feets : 4,

things : ["papier","telecommande","clé usb"],

knock : function(){

console.log("TAP TAP ! ");

console.log( chair.things ); // ["papier",..] si on utilise des références

console.log( this.things ); // ["papier",...] si on utilise d'autres instances

}

};

console.log( chair[feets] ); // pas bon

console.log( chair.feets ); // -> 4

console.log( chair["feets"] ); // -> 4

chair.knock(); // lance la méthode knock

for(let key in chair){ // boucle

console.log(typeof key); // affiche le type de key ici String

console.log( key ) // affiche chaque clé de mon objet ("color", "feets",...)

console.log( chair[key] ) // affiche chaque valeur ("#FF0000", 4,...)

}

voir : [w3schools json objects](https://www.w3schools.com/js/js_json_objects.asp)

**Exercice : PNJ (Personnage non-joueur)**

* Crée un objet "Character" qui contient les informations suivantes :
* name (string)
* age (numero)
* items\_to\_give (tableau)
* afficher chaque information sur une ligne séparés dans la console **(for in)**
* faire une fonction "giveItem" qui permet au PNJ de donner un objet aléatoirement

**Exercice : Shop**

Crée un tableau avec des *objets* à vendre (épée, hache, sceptre, etc.)

Caractéristique de chaque objet :

* title (string)
* physic (int)
* magic (int)
* minLevel (int)
* available (boolean)

Manipulation :

* faire une fonction pour afficher tout les objets
* faire une fonction pour afficher les objets disponibles
* faire une fonction pour afficher les objets dont le niveau minimum est de 10

**Exercice : Personnage**

* crée un objet "mainCharacter" qui possède le propriétés suivantes :
* name (string)
* level (int)
* life (int)
* weapon (object)
* attack (function)
* l'objet "weapon" possède les propriétés suivantes :
* name (string)
* damage (int)
* Appeler la fonction "attack" du personnage
  + la fonction doit retourner : (le nom du personnage) attaque avec l'arme (nom de l'arme) les dégâts sont (niveau du personnage multiplié par le damage de l'arme)

**BONUS :**

**Exercice : Adversaire**

* crée un objet "character" qui possèe le propriétés suivantes :
* name (string)
* level (int)
* life (int)
* weapon (object)
* attack (function)
* receiveDamage (function)
* l'objet "weapon" possèe les propriétés suivantes :
* name (string)
* damage (int)
* crée un objet "opponentCharacter" a partir de character
* crée un objet "mainCharacter" a partir de character
* Appeler la fonction "attack" de "mainCharacter" afin qu'il attaque "opponentCharacter"
* la fonction doit afficher dans la console (ligne par lige)
  + (le nom du personnage) attaque (nom de l'adversaire)
  + avec l'arme (nom de l'arme)
  + et lui inflige (niveau du personnage multiplié par le damage de l'arme) de dégats
  + (nom de l'adversaire) a maintenant (life de l'adversaire) de vie

## Exercice : Juste prix

Déclarez les variables suivantes :

* Variable qui sera utilisée pour afficher un message
* Variable qui va compter le nombre d'essais
* Variable de la valeur **minimum** qu'on peut entrer (ici 20)
* Variable de la valeur **maximale** qu'on peut entrer (ici 80)

Déclarez les fonctions suivantes :

* crée une fonction qui retourne un numéro aléatoire arondi entre la variable minimale et la variable maximale
* crée une fonction
  + qui prends un argument pour tester si le numéro qu'on a entré est le bon
  + la fonction retourne un de ces 3 messages : C'est plus, C'est moins,C'est juste tu as trouvé en X coups

Utiliser la console **ou** les fonctions natives suivantes :

* window.prompt();
* window.alert();
* window.confirm();

## JavaScript

### Le DOM

Le Document Object Model est un moyen de manipuler la structure et le style d'une page HTML. Il représente les entrailles de la page que le navigateur voit. Il permet aux développeurs de le modifier avec du JavaScript. Si vous souhaitez avoir un regard sur le DOM d'une page, ouvrez les outils de développement dans votre navigateur et recherchez "l’inspecteur d‘éléments". Dans la plupart des navigateurs, vous pouvez supprimer et modifier directement les éléments du DOM.

#### Les arbres et les branches

HTML est une structure de type XML. Les éléments qui le forment ont une structure de nœuds avec des parents et des enfants. Comme le tronc et les branches d'un arbre. Il y a un élément de racine (html) avec des branches comme le head et body. Et chaque branche a ses propres branches. Pour cette raison, le DOM est aussi appelé l'arbre DOM. La modification du DOM se fait en choisissant un élément et en changeant quelque chose à son sujet. C’est une action que l’on fait souvent en JavaScript. Pour accéder au DOM à partir de JavaScript, l'objet document est utilisé. Il est fourni par le navigateur et permet au code sur la page d'interagir avec le contenu.

#### Obtenir un élément

Tout d'abord, nous devons savoir comment obtenir un élément. Il y a un certain nombre de façons de le faire.

#### Par l’ID

document.getElementById est une méthode pour obtenir un élément par son ID.

var pageHeader = document.getElementById('pageHeader');

L'élément de pageHeader peut alors être manipulé. Sa taille et la couleur peuvent être modifiées par exemple. Ou un autre code peut être déclaré pour gérer l'élément quand on clique dessus ou quand il es survolé. Notez que getElementById est une méthode de l'objet document. La plupart des méthodes utilisées pour accéder à la page se trouvent sur l'objet document. Par le tag (nom de balise) document.getElementsByTagName fonctionne de la même manière que getElementById, sauf qu'elle prend un nom de tag (div, ul, li, etc.) au lieu d'un ID. Elle renvoie une NodeList, qui est essentiellement un tableau des éléments DOM qu’elle a trouvé.

#### Par le nom de classe

document.getElementsByClassName renvoie le même genre de NodeList comme getElementsByTagName, sauf que vous passez un nom de classe et non un nom de tag.

#### En sélecteur CSS

De nouvelles méthodes sont disponibles dans les navigateurs modernes. Elles font des sélections d’éléments plus faciles en permettant l'utilisation de sélecteurs CSS. Il s'agit de : document.querySelector et document.querySelectorAll.

var pageHeader = document.querySelector('#header');

var bouton = document.querySelectorAll('.btn');

querySelector, comme getElementById, retourne un seul élément alors querySelectorAll renvoie une NodeList. Si plusieurs éléments correspondent au même sélecteur et que vous utilisez querySelector, seul le premier élément sera retourné.

## DOM

* Télécharger fichier "file.zip"

# #EXERCICE 1 : Manipulation de classes

* sélectionner le body
* lui retirer la classe "bg-aqua"
* lui ajouter la classe "bg-olive"
* sélectioner first-paragraph
* lui retirer les classes "bg-lime" et "gray"
* lui ajouter la class "aqua"
* sélectionner tous les éléments qui ont la classe 'bg-silver'
* modifier tout ces éléments en leurs ajoutant la classe "bg-teal"
* modifier tout ces éléments en leurs supprimant la classe "bg-silver"
* sélectionne tous les éléments de type 'blockquote'
* modifier tout ces éléments en leurs ajoutant la classe "bg-white"

## EXERCICE 2 : Selecteurs CSS

* avec querySelector sélectioner l'élément 'my-table'
* lui ajouter la classe "bg-purple"
* avec querySelectorAll sélectioner tous les paragraphes dans 'container'
* modifier tout ces éléments en leurs ajoutant la classe 'shadow'

## EXERCICE 3

* sélectionne tous les éléments de type 'pre'
* avec la propriété style change la couleur de texte
* avec la propriété style change la couleur de fond grâce à backgroundColor
* avec la propriété style ajouter un border-top de 3px solid red (n'oublie pas que tu peux acceder au propriété d'un objet grâce comme ceci : ["prop-name"] )
* avec la propriété style ajouter un border-bottom de 3px solid red
* sélectionne le premier élément de type 'h3'
* ramplacer son contenu HTML en <em>Itelic title ! yeah !</em>
* sélectionne le premier élément de type 'h2'
* ramplacer son contenu texte par en <strong>HTML doens't work !</strong>

## EXERCICE 4 : Création d'éléments

* sélectioner le premier 'ul'
* crée un élément de type 'li'
* ajouter ce texte Mon meilleur ami est <a href='http://www.google.com'>Google</a> dans le 'li'
* ajouter ce 'li' l'élément dans le ul
* sélectionner le premier lien dans le 'li' précedement crée
* avec la propriété style change la couleur de texte de ce lien

## EXERCICE 4 : Création et suppression de plusieurs éléments

* sélectionner le premier élément 'ol'
* faire une boucle sur tous les enfants de ce dernier grâce à sa propriété children
* supprimer chaque enfant de 'ol' grâce à removeChild()
* déclare dans un tableau les valuers suivantes : ["Silent Teacher","Code Monkey", "CodeCombat"]
* faire une boucle sur tous les enfants du tableau précédement crée
* pour chaque valeur du tableau crée un élément 'li' crâce à document.createElement()
* ajouter à chaque 'li' la valeur courrante tableau
* ensuite ajouter chaque 'li' à l'élément 'ol' précédement sélectionné

### Reminder

Selecteurs

* document.getElementById()
* document.getElementsByClassName()
* document.getElementsByTagName()
* document.querySelector()
* document.querySelectorAll()

Création d'éléments

* document.createElement()
* element.appendChild()

Manipulation de classes CSS

* element.className
* element.classList.remove()
* element.classList.add()
* element.classList.toggle()

Manipulation CSS

* element.style ( ex :*element.style.backgroundColor = "blue"*)

Autres

* element.id
* element.innerHTML
* element.innerText

## JavaScript

### AJAX

Dans les premières années, les pages web étaient simples. Les pages contenaient du texte, avec quelque fois un style CSS. Elles comportaient aussi des liens vers d'autres pages pour obtenir de nouveau contenu que vous aviez déplacé sur une page. Mais des développeurs, plus ambitieux, ont décidé de construire des applications plus interactives. Il était dans leurs objectifs de trouver un moyen pour charger du nouveau contenu dans une page sans rechargement complet de cette dernière. Pour ajouter du contenu dans une page, comme de nouveaux articles sur un site a défilement infinie ou pour vous informer de nouveaux e­mails par exemple, on fait une requête HTTP XML (XHR). Les applications Web qui font cela sont aussi appelés “applications AJAX”. AJAX correspond à Asynchronous JavaScript and XML. Presque tous les sites qui ajoutent du nouveau contenu sans rechargement de la page (comme Facebook, Gmail, Google Maps, etc.) utilisent la même technique. Ce sont les développeurs de Microsoft Outlook Web Access qui ont initialement créé le XMLHttpRequest.

### Requête HTTP XML

Alors a quoi ressemble une XMLHttpRequest ?

var req = new XMLHttpRequest();

req.onload = function (event) {...};

req.open('get', 'myfile.txt', true);

req.send();

La première chose à faire est de créer une nouvelle demande de XMLHttpRequest, en l'appelant comme une fonction. Elle retourne un objet qu’on enregistre dans une variable. Ensuite, nous spécifions une fonction de callback, à appeler lorsque les données sont chargées. Les informations de l'événement sont transmises comme premier argument de la fonction. Par après, nous spécifions comment obtenir les données que nous voulons, en utilisant req.open. Le premier argument est la méthode HTTP (GET, POST, PUT, etc). Vient ensuite l'URL pour aller chercher le fichier.

Le troisième argument est un booléen indiquant si la demande est asynchrone. Le paramètre par défaut est false. Dans le cas ou la requête n’est pas asynchrone, l'exécution du code s’arrête à cette ligne jusqu'à ce que les données soient récupérées. La requête est alors appelée synchrone. Les requête synchrones ne sont pas souvent utilisées étant donné qu’une demande à un serveur peut prendre une éternité. Le navigateur ne fera donc rien durant une longue période.

Sur la dernière ligne nous envoyons la requête au navigateur pour enclencher la demande de données. L’utilisation d'unXMLHttpRequest permet de charger HTML, JSON, XML et du texte brut sur HTTP et HTTPS. Il prend également en charge d'autres protocoles tels que FTP. Ces requêtes sont très utiles dans le développement d'applications JavaScript. Cette méthode a été fortement développée à tel point que toutes les applications l’utilisent actuellement. Des bibliothèques et des frameworks ont aussi été construits pour aider à faciliter leur utilisation.

### JSON

Le JavaScript Object Notation n’est pas du JavaScript. Officiellement c’est un language totalement différent avec ses propres spécificités. Il est néanmoins une partie importante du JavaScript et il est important d’aborder le sujet. JSON est un ensemble de règles de formatage de texte pour stocker et transférer des données dans une machine et de façon lisible. Il ressemble beaucoup à la syntaxe d’un objet en JavaScript. Et c’est de là que provient son nom. Voici quelques JSON :

{

"name": "Yoda",

"age": 894,

"lightsaber": {"color": "vert"}

}

Comme dans le JavaScript, la syntaxe avec les accolades est utilisée. L'exemple ci­dessus est valide en JavaScript. JSON est utilisé pour transférer des informations entre votre navigateur et un serveur. Les informations sont enregistrées dans des fichiers textes et peuvent être récupérées plus tard. Car il s'agit tout simplement de texte. Cela signifie que vous ne pouvez pas stocker des données complexes comme une fonction. Mais vous pouvez stocker des tableaux, des objets contenant des données simples, des chaînes de caractères et des nombres. JSON est en train de prendre la place du XML comme format de transfert de données. De nombreuses nouvelles API Web sont écrites exclusivement pour utiliser du JSON. Vous utilisez donc la technologie AJAX pour charger du JSON.

### Utilisation de JSON

En JavaScript l’objet JSON est disponible dans à peu près tous les navigateurs modernes. Il existe des moyens de l'ajouter aux navigateurs qui ne l'ont pas.

var jsonString = JSON.stringify ({

faire: "McLaren",

modèle: "MP4­12C",

miles: 5023

});

JSON.stringify convertit un objet en une chaîne JSON. Dans cet exemple, jsonString devient : {"faire": "McLaren", "modèle": "MP4­12C", "miles": 5023}.

var car = JSON.parse(jsonString);

La chaîne de caractère JSON peut être reconvertie en un objet JavaScript. En utilisant JSON.parse. car est maintenant utilisable comme un objet JavaScript normale. Nous pouvons maintenant redéfinir ses propriétés :

car.model = "P1";

Ressources :

* [generate data](http://www.generatedata.com/)
* [Array methods](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array)

**/!\ Utilise Firefox pour cet exercice !**

**avec les fonctions des tableaux**

1. charger le fichier "data.json" tip : xmlHttpRequest
2. ajouter un score aléatoire (0 -> 1000) à toutes ces personnes tip: arr.map()
3. classer l'odre du tableau selon les scores tip : arr.sort()
4. mettre personnes dans 3 categories (a > 750, b > 500, c < 500 ) tip : forEach() les catgories sont des tableaux
5. compter combien de personne viennent du "Bahrain" tip : filter()
6. afficher le plus grand score dans la console tip : sort()
7. afficher le plus petit score tip : sort()

Ressources :

* [generate data](http://www.generatedata.com/)
* [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest)
* [you might not need jquery](http://youmightnotneedjquery.com/)
* [Array methods](https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array)

# Mouse

### Évènements et callbacks

Dans le navigateur le code peut être exécuté selon des évements. L'écriture d’applications interactives en JavaScript consiste souvent à attendre pour réagir aux événements et pour modifier le comportement du navigateur. Les événements se produisent lorsque la page se charge, quand l'utilisateur interagit (clics, survols, fait des changements) et lors d'une myriade d'autres événements. Ces évènements peuvent aussi être déclenchés manuellement. Pour réagir à un événement, vous devez fournir une fonction qui sera appelée par le navigateur lorsque l'événement se produit. Cette fonction est appelée “callback”.

var handleClick = function (event) {

// Faire quelque chose!

};

var bouton = document.querySelector('#big­button');

button.addEventListener ('click', handleClick);

addEventListener est une méthode qui agit sur tous les éléments DOM. Ici, il est appelé sur un élément enregistré dans la variable button. Le premier argument est une chaîne de caractères : le nom de l'événement a écouter. Le clic correspond à un clic de souris ou à une pression du doigt. Le second argument est la fonction de callback, ici c’est handleClick. Malheureusement, Internet Explorer ne supporte pas addEventListener dans les versions antérieures a Internet Explorer 9. Au lieu de addEventListener on doit utiliser attachEvent.

button.attachEvent("onclick", handleClick);

Notez que attachEvent demande onclick, pas seulement click. Il faut faire de même pour les autres éléments. C’est ce genre de différences qui ont permis la création de bibliothèques comme jQuery. Elle vous aide à ne pas vous occuper des différentes manières d'écouter les événements entre les navigateurs.

Les données concernant l’événement sont transmis à la fonction callback. Jetez un oeil à handleClick. Vous pouvez voir son argument : event. C’est un objet dont les propriétés décrivent ce qu’il s’est passé. Voici un événement exemple. Vous pourriez voir dans un callback de l'événement click comme handleClick. Il y a beaucoup de propriétés qui vous donne une idée de l'endroit où l'événement s’est produit sur la page (comme pageX et offsetY). Ils sont légèrement différents car ils dépendent du point de référence utilisé pour mesurer. Vous pouvez également voir la cible qui est une référence au nœud qui a été cliqué.

{

offsetX: 74,

offsetY: 10,

pageX: 154,

pageY: 576,

screenX: 154,

screenY: 489,

target: H2,

timeStamp: 1363131952985,

type: "click",

x: 154,

y: 395

}

### Les évènements de la souris

* click
* ctextmenu
* dblclick
* mousedown
* mouseenter
* mouseleave
* mousemove
* mouseover
* mouseout
* mouseup

### Timing

voir :

setInterval(abc, tmp)

setTimeout(abc, tmp)

clearInterval(fn);

clearTiemout(fn);

# Mouse

## Évènements de la souris

click, ctextmenu, dblclick, mousedown, mouseenter, mouseleave, mousemove, mouseover, mouseout, mouseup

### Exercices 1

* Crée (en HTML) 3 DIV possedant la même classe 'hoverMe'
* Faire disparaitre les divs lorsque l'on passe la souris dessus

Tip : utiliser le mouseenter ou mouseover

### Exercices 2

* Crée (en HTML) une DIV avec une l'id "reset"
* Faire de sorte que quand on clique dessus elle fasse réaparaitre les 3 divs premièrement créé

Tip : avec une animation ce serait super cool

### Exercices 3

* Crée (en HTML) 2 DIV avec une id chacune 'axe-x' et 'axe-y'
* Faire de sorte que lorsque l'on bouge la souris dans la fenêtre, la position sois affiché dans les divs associés

Tip : Utiliser l'evenement reçu en argument Tip : Utiliser innerHTML ou innerText

# KEYBOARD

### Les évènements du clavier

Voir les évènements de la souris

var handleKey = function (event) {

// Faire quelque chose!

};

window.addEventListener ('keydown', handleKey);

* keydown
* keypress
* keyup

# KEYBOARD

### Exercices 1

* Crée (en HTML) une DIV avec l'id "character"
* Faire changer de couleur de fond a cette DIVà chaque fois qu'on appuie sur une numéro de 0 à 9 sur le clavier

Tip : utiliser le switch

### Exercices 2

* Crée (en HTML) 4 divs avec les id suivantes : "up, down, left, right"
* crée une class 'highlight' dans le CSS qui réduit l'opacité de l'élément
* Pour chaque div ajouter la classe 'highlight' quand une des touches directionel est enfoncée
* Pour chaque div retier la classe 'highlight' quand on relâche la touche

Tip : Faire la structure HTML sans JavaScript