I01 – JavaScript

Table des matières

[Présentation 3](#_Toc31363393)

[Références : 3](#_Toc31363394)

[Syntaxe 3](#_Toc31363395)

[Annexe 4](#_Toc31363396)

[Puissance 4 4](#_Toc31363397)

[Références : 4](#_Toc31363398)

# Présentation

Le langage a été créé en 1995. A la base, il servait « juste » à rendre les pages Web dynamiques et interactives mais c’est aujourd’hui bien plus que cela (par exemple : on peut aujourd’hui faire des applications IOS / Android avec du JavaScripts).

Un site Web ce n’est rien d’autre qu’un serveur et un client qui communique. Quand un client demande au serveur une page web (front-end – back-end), il lui renvoi des informations comme par exemple la page HTML à afficher. Celle-ci est alors censé rester statique jusqu’à ce que l’utilisateur demande le chargement d’une autre page via un hyperlien. Le JavaScript permet lui de modifier ce fichier HTML, par exemple en fonction des évènements que produit l’utilisateur. Par exemple quand celui-ci passe sa souris au-dessus d’une vignette de vidéo et que celle-ci se lance automatiquement (sur Netflix par Exemple ou YouTube). Il permet également de créer des animations, et même des jeux vidéo 3D (par exemple : grâce à la bibliothèque Unity).

## Références

Tutoriel très complet sur JavaScript (français)

<https://developer.mozilla.org/fr/docs/Web/JavaScript>

Tutoriel complet et à destination des débutant (français)

<https://openclassrooms.com/fr/courses/2984401-apprenez-a-coder-avec-javascript>

Tutoriel qui résume rapidement les différents éléments du JavaScript (anglais facile)

<https://www.w3schools.com/Js/>

Un premier exemple et une excellente page pour tester des petits scripts :

<https://www.w3schools.com/Js/tryit.asp?filename=tryjs_whereto_head>

# Syntaxe et sémantique

La syntaxe d’un langage informatique est la manière dont les différents éléments du langage peuvent être assemblé pour former un programme fonctionnelle (ex : la manière dont une boucle (ex : while) peut être combinée avec une instruction conditionnelle (ex : if).

La sémantique est le sens des phrases utilisées dans le langage.

Exemple : a > b ? a : b => si a est plus grand que b renvoi a sinon renvoi b.

Nous allons donc voir le vocabulaire utilisé pour écrire du code JavaScript, la signification et la manière d’agencer les éléments. Notre objectif sera de créer des petits scripts permettant de réagir aux interactions de l’utilisateur avec le navigateur web et de changer la structure de la page HTML en conséquence.

L’ensemble de la syntaxe est résumé dans un code JavaScript afin de le tester plus facilement dans le fichier disponible sur le repository Git du cours : <https://github.com/martini224/EFP_I01/blob/master/code/syntaxe.html>

## Divers

Écrire un commentaire, c'est à dire du code qui ne sera ignoré par le navigateur.

/\*

Ceci est un bloc commentaire, on peut écrire sur plusieurs lignes.

Les commentaires sont utiles pour attirer l'attention de la personne lisant le code (soi-même ou d’autres personnes) sur un élément qui pourrait être difficile à comprendre, ou simplement pour expliquer le fonctionnement d'un bout de code.

\*/

### Debugger son code

Pour ouvrir la console, il suffit d'appuyer sur F12 sur votre navigateur préféré. Dans cette console sera affiché différents messages d'erreur ou de log. Par exemple si vous écrivez console.log("bonjour"); dans votre code JS, bonjour s'affichera dans la console.

Plus d'info sur la console et le débogage :

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Introduction#La_console_web>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/debugger>

## Les variables

var nomDeLaVariable; déclaration d'une variable. Une variable doit être déclarée avant d'être utilisée.

Le nom d'une variable ne doit pas commencer par un chiffre et ne doit pas contenir de caractères spéciaux (ex : é, @, ù, ...).

Il est également préférable que le nom commence par une minuscule et que chaque autre mot que compose le nom de la variable commence par une majuscule.

Il est également préférable que le nom de la variable soit explicite quant à son utilité.

### Opération mathématique

nomDeLaVariable = 3; on assigne la valeur entière 3 à la variable

nomDeLaVariable += 3; on ajoute 3 à la valeur de la variable (3+3, donc 6)

nomDeLaVariable = nomDeLaVariable + 3; écriture plus longue mais équivalente.

let n = 0.5; Une variable peut être déclarée avec le mot clé "var" ou "let". On préférera utilisé "let".

n = n + 3 - 3 + (2 \* (5 / 2)); un exemple d’opération mathématique écrite en JS.

\*\* = exponentiel

% modulo, c'est à dire le reste du résultat de la division. Exemple 11%2 = 1 car 11 = 5 x 2 + 1.

n++; on incrémente de 1 la valeur de la variable x, ce qui est équivalent à x += 1 ou x = x + 1.

n--; on décrémente de 1 la valeur de la variable x, ce qui est équivalent à x -= 1 ou x = x - 1.

### Les variables textuelles

let texte = "text"; on assigne une valeur textuelle à la variable texte. Une valeur ou une variable de type textuelle est appelée un string.

texte = 'text'; on réassigne une nouvelle valeur à cette variable.

Remarquez qu'une valeur textuelle (un string) peut s'écrire entre " " ou entre ' '.

texte = 'bonjour' + ' ' + "le monde"; permet de fusionner plusieurs string, on appelle cela la concaténation.

### Exercice

Quel sera le résultat de :

let x = 5 + 5;

let y = "5" + 5;

let z = "Hello" + 5;

texte.length; renvoie la longueur de la chaine de caractères

## Les comparateurs

let a = 1;

let b = 2;

let aText = '1';

### Égalité

a == b; false

a === b; false

a == aText; true

a === aText; false

### Inégalité

a != b; true

a !== b; true

a != aText; false

a !== aText; true

a < b; true

a <= b; true

a <= aText; true

### Ternaire

a > b ? "si true alors on affiche ceci" : "si false on affiche cela";

### OR, AND, NOT

a < b && a == aText; true, && = ET (AND) ce qui veut dire en français : si a est plus petit que b et que a égal aText. Les deux conditions doivent être vraies.

a < b || a == aText; true, || = OU (OR° ce qui veut dire en français : si a est plus petit que b ou que a égal aText. Seule une seule condition vraie suffit.

!(a < b); a < b = true et le ! (NOT) devant signifie l'inverse du résultat, donc false

### Conditionnel

if(a < b)

{

// la condition est vraie.

alert("Vrai");

}

else

{

// la condition est fausse.

alert("Faux");

}

### Switch

Un switch permet de soulager l'écriture des IF quand il y a beaucoup de cas possible.

new Date() permet d'obtenir la date du jour, et la fonction getDay() permet d'obtenir le numéro du jour de la semaine.

switch (new Date().getDay()) {

case 0: La semaine calendrier commence le dimanche en JS

day = "Dimanche";

break;

case 1:

day = "Lundi";

break;

case 2:

day = "Mardi";

break;

case 3:

day = "Mercredi";

break;

case 4:

day = "Thursday";

break;

case 5:

day = "Friday";

break;

case 6:

day = "Saturday";

break;

}

console.log(day); va afficher le nom du jour

Pour en savoir plus sur les dates : <https://www.w3schools.com/js/js_dates.asp>

## Type de données

Dans la plupart des langages, les variables sont typées. C'est à dire que leur valeur doit absolument être du type déclaré (en entier, un string, …). C’est le cas du JavaScript qui est dit faiblement typé, car il n’y a que 6 types différents de base. Le JS est dit à typage dynamique, car le type d’une variable déclarée peut changer en cours d’exécution d’un code JS.

En Java le typage est statique, une variable est déclarée avec son type (ex : int unEntier ; String uneChaineDeCaracteres ;) et la variable doit toujours avoir une valeur de ce type. En Java, une variable string déclarée comme ceci : String prenom = "Martin"; ne pourra par la suite pas être : prenom = 1; car la variable prenom est un string ce qui est incompatible avec un entier (1).

En JavaScript, on n’indique pas de type au moment de la déclaration de variable. Ceci est du au faites que le type de la variable peut changer en fonction du type de sa valeur. Si let prénom = "Martin"; on pourra écrire après prénom = 1;

Voici les types de données en JavaScript :

**string** pour une chaine de caractères.

**number** un nombre 1, 1202015, -1, -5.21245.

**boolean** true, false (vrai ou faux).

**undefined** type non défini qui indique que la variable n'est pas définie.

**function** pour une fonction.

**object** pour un objet.

### Vérifier le type d'une variable (typeof)

let myVar;

typeof myVar; undefined

myVar = 1;

typeof myVar; number

myVar = function(i){

return i++;

}

typeof myVar; function

myVar = {nom: "Martin", prenom: "Rouffiange"};

typeof myVar; object

# Les fonctions

Une fonction est un bloc de code exécutant une tâche bien précise. Une fonction (et ce de manière générale en informatique) peut avoir des valeurs d'entrée (que l'on appel arguments ou paramètres de la fonction). Celle-ci peut également retourner un résultat (la valeur de retour). Les fonctions sont particulièrement utile pour découper le code en plusieurs sous tâches afin de le rendre plus lisble et de pouvoir réutiliser facilement le code afind'éviter la duplication de code (du code qui fait plusieurs fois la même chose.

Par exemple une fonction de conversion qui converit les degré fahrenheit en celcius :

function toCelsius(fahrenheit) {

return (5 / 9) \* (fahrenheit - 32);

}

toCelsius(10); renvoi -12.222222222222223

Il est en effet plus facile d'écrire le calcul une fois pour toute que de le répeter plusieurs fois.

A chaque fois que vous avec le sentiment que vous aller utiliser plusieurs un même bout de code, ou que vous avez déja écritun bout de code quelque part ou encore qu'un bout de code commence a devenir trop volumineux c'est qu'il faut découper en plus petites fonction. Seul l'expérience pourra vous donnez le bon feeling a ce niveau. Mais la règle est que l'informaticien est paresseux, ce qui signifie qu'il n'aime pas faire quelque chose qui a déjà été fait et qu'il peut réutiliser (que ce soit par lui même ou par quelqu'un d'autre).

Il existe un tas de fonctions déjà écrite pour nous. Par exemple Bootstrap qui permet de mettre en page plus facilement.

On peut alors récupérer ce code et l'ajouter au notre si la licence de celui-ci en donne la permission.

Il existe aussi un tas de fonctions de base en Js

## Opération sur les nombres

isNaN permet de vérifier si une variable n'est pas un nombre

isNaN(10); renvoit false car 10 est un nombre

isNaN("x"); renvoit true car x n'est pas un nombre

Number("10"); crée un nombre à partir d'un nombre ou d'un string

Number("10.33"); crée un nombre à partir d'un nombre ou d'un string qui peut contenir une virgule

parseInt("10"); crée un nombre à partir d'un nombre ou d'un string qui ne peut pas contenir une virgule

parseFloat("10.33"); crée un nombre à partir d'un nombre ou d'un string qui peut contenir une virgule

## Opérations mathématique de base via Math

arrondi supérieur

Math.round(4.7); renvoi 5

puissance : ici 8 puissance 2

Math.pow(8, 2); returns 64

racine carrée

Math.sqrt(64); returns 8

arrondi supérieur

Math.ceil(4.4); returns 5

arrondi inférieur

Math.floor(4.7); returns 4

génére un nombre réel pseudo-aléatoire entre 0 (inclus) et 1 (exclus)

Math.random();

génére un nombre entier (grâce à Math.floor()) pseudo-aléatoire entre 0 (inclus) et 1 (exclus)

Math.floor((Math.random() \* 10) + 1);

Voir plus : <https://www.w3schools.com/js/js_math.asp>

## Opération sur les chaines de caractères (string)

let str = "Apple, Banana, Kiwi";

substring : renvoi la chaine de caractères situé entre le 7ème caractère et le 13ème (inclus)

str.substring(7, 13); renvoi un nouveau string Banana

Remplacement du 1er élément correspondant au premier paramètre de la fonction

str.replace('Banana', 'Banane') renvoi un nouveau "Apple, Banane, Kiwi"

replace : Remplacement de tout les éléments correspondant au premier paramètre de la fonction

str = 'pomme pomme';

str.replace(/pomme/g, 'poire') renvoi un nouveau string "poire poire"

toUpperCase : Mettre la chaine en majuscule

str.toUpperCase(); renvoi "POMME POMME"

toLowerCase : Mettre la chaine en majuscule

str.toLowerCase(); renvoi "pomme pomme"

trim : Supprime les espaces du début de du string et de la fin

str = ' pomme ';

str.trim(); renvoi "pomme"

split : sépare le string en un tableau de string en séparant la chaine de caractère par la chaine envoyer en paramètre

str = 'pomme pomme pomme';

str.split(" "); renvoit un tableau ['pomme', 'pomme', 'pomme']

voir plus : <https://www.w3schools.com/js/js_string_methods.asp>

## Les tableaux (array)

Les tableaux permettent de stocker plusieurs données en même temps, typiquement une liste d'object (une liste de joueurs, liste d'items...)

Exemple de déclaration d'un tableau avec des valeurs string

let names = ["Martin", "Jérome", "Henri"];

Déclaration d'un tableau vide

names = [];

On peut également écrire, c'est une écriture équivalente :

names = new Array();

names = new Array("Martin", "Jérome", "Henri");

Acceder à un élément du tableau :

names[0]; renvoi Martin

Connaitre le nombre d'élements dans le tableau :

names.length; renvoi 3

Si on essaye d'accéder à un élément qui n'est pas dans le tableau :

names[3]; renvoi undefined

Il est donc toujours préférable de vérifier la longueur d'un tableau avant d'accéder à un élément :

if(names.length > 3){

names[3];

} else {

/\*

la taille du tableau est = 3, car il y a 3 élément. Or on essaye d'acceder au 4ème élément, donc on arrivera ici.

En effet, le premier élément du tableau est accessible via names[0]; Il faut donc faire -1 pour acceder à l'élément souhaité.

Ex : le 4ème élément : 4-1 = 3

\*/

}

## Fonction sur les tableaux :

Join : permet de fusionner les éléments du tableau dans un string.

let fruits = ["Banane", "Orange", "Pomme", "Mangue"];

fruits.join(" / "); renvoi un string : Banane / Orange / Pomme / Mangue"

shift : retire le premier élément du tableau

fruits.shift(); retire "Banane"

pop : retire le dernier élément du tableau

fruits.pop(); retire "Mangue"

unshift : ajoute un élément au début du tableau

fruits.unshift("Citron"); ajoute citron au tableau

push : ajoute un élément à la fin tableau

fruits.push("Citron"); ajoute orange au tableau

concat fusionne deux tableaux :

let names2 = ["Robin", "Morgan"];

names.concat(names); renvoi un nouveau tableau "Martin", "Jérome", "Henri", "Robin", "Morgan"

sort : trie le contenu du tableau dans l'ordre croissant

names2.sort(); le tableau names2 devient : "Morgan", "Robin"

reverse : trie le contenu du tableau dans l'ordre décroissant

names2.reverse(); le tableau names2 devient : "Robin", "Morgan"

Voir plus : <https://www.w3schools.com/Js/js_array_methods.asp>

## Les boucles

Une boucle permet d'exécuter le même code plusieurs fois, tant qu'une condition est vraie.

### While

let i = 0;

while(i < 10) {

console.log(i); // affiche 0, 1, 2, .. jusqu’à 9 dans la console

i++;

}

### For

Une autre façon d'écrire une boucle plus condensée.

for(let j = 0; j < 10; j++) {

console.log(j); // affiche 0, 1, 2, .. jusqu'à 9 dans la console

}

Les boucles sont particulièrement utiles pour traiter le contenu d'un tableaux

fruits = ["Banane", "Orange", "Pomme", "Mangue"];

for(iFruit = 0; iFruit < fruits.length; iFruit++) {

console.log(fruits[iFruit]); // affiche chaque élément du tableau dans la console

}

Une autre façons de traiter un tableau plus condensée.

for(iFruit in fruits) {

console.log(fruits[iFruit]); // affiche chaque élément du tableau dans la console

}

### Les fonctions de boucle pour les tableaux

Les tableaux possèdent une fonction propre pour parcourir chaque élément :

fruits.forEach(fruit => console.log(fruit)); affiche chaque élément du tableau dans la console

Les tableaux possèdent également d’autres fonctions pour traiter leurs éléments :

map : permet de renvoyer une copie du tableau a laquelle on a appliqué un traitement

let numbers = [45, 4, 9, 16, 25];

numbers.map(number => number \* 2); renvoit une copie du tableau 90,8,18,32,50 (on a fait x2 à chaque élément)

filter : renvoi une copie du tableau correspondant a une condition

numbers = [45, 4, 9, 16, 25];

numbers.filter(number => number > 18); renvoit une copie du tableau 45,25 (les éléments plus grand que 18)

voir plus : <https://www.w3schools.com/Js/js_array_iteration.asp>

### Les fonctions asynchrone

## Les objets

Un objet permet de stocker plusieurs variables pouvant avoir des types différents, dont des fonctions.

Par exemple un objet représentant un joueur :

let object\_player = {nom: 'jouer', level: 3};

Les variables d'un objet sont appelé ses attributs, et les fonctions les méthodes.

Les informations sur les classes si dessous sont à titre informatif dans le cadre de la formation, vous y reviendrez plus en détails plus tard.

Une manière de représenter facilement ce a quoi doit ressembler un objet, c'est à dire ce que doit être les attributs et méthodes d'un objet et de créer une classe.

class Player {

constructor(name, level){

this.name = name;

this.level = level;

}

displayName(){

console.log(this.name);

}

}

ensuite pour créer un object, quand on crée un object d'une certaine classe, on dit qu'on instancie la classe.

object\_player = new Player('jouer', 3); si vous afficher la variable object\_player dans la console, cela ressemblera a {nom: 'jouer', level: 3}

object\_player.displayName(); fait appel à la méthode displayName qui affiche le nom du jouer dans la console

object\_player.name; accède à l'attribut name qui renvoi le nom du joueur

Pour en savoir plus sur les classes : <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Classes>

## Le DOM : la structure du code HTML en JS

Document Object Model ou DOM (pour modèle objet de document), c'est la représentation en arbre du document HTML.

Le DOM est composé de noeud, qui sonts des objets ayant des attributs et des méthodes permettant de connaitre et modifier le document HTML. La représentation en arbre permet de voyager dans le document, exemple : parent <- noeud -> enfants, freres. Tout noeud (sauf la racine = document) possède un parent, et peut avoir un ou plusieurs enfants et frères.

Un noeud peut être un élément HTML, un texte ou un commentaire

selecteurs : ils permettent de selectionner un élément dans le DOM

Renvoie l'élement possédant l'id envoyé en paramètre

document.getElementById('id');

Renvoie un tableau d'élements possédant la classe envoyé en paramètre

document.getElementsByClassName('classe');

Renvoie le premier élément correspond au selecteur css envoyé en paramètre

document.querySelector('#id');

Renvoie un tableau d'éléments correspond au selecteur css envoyé en paramètre

document.querySelectorAll('.classe');

Voir tous les selecteurs : <https://www.w3schools.com/cssref/css_selectors.asp>

let element = document.querySelector('#demo');

querySelector() et querySelectorAll() peuvent également être utilisé sur un élément précis, la recherche ce fera alors parmis toutes la descendance de l’élément.

element.querySelector('#id');

Afficher ou modifier le contenu textuel d'un élément

element.innerText; pour l'afficher

element.innerText = "test";

Afficher ou modifier le code HTML d'un élément

element.innerHTML; pour l'afficher

element.innerHTML = "<div><p></p></div>"; modifie le code HTML de l'élément

Acceder à un attribut

element.getAttribute('id'); renvoie l'id de l'élement, ici 'demo'

Afficher ou modifier le code CSS d'un élément (remplace tout le code)

element.style.cssText = "color: blue; border: 1px solid black";

Une autre manière de l'écrire

element.setAttribute("style", "color:red; border: 1px solid blue;");

Affiche, modifie ou ajoute un élément css précis, en laissant les autres intactes.

element.style.color = "blue";

Afficher, gérer les classes d'un élément

element.classList; renvoie les classes de l'élément, ici 'demo\_classe'

Ajouter une classe

element.classList.add('une\_classe');

Retirer une classe

element.classList.remove('une\_classe');

Si la classe est présente la retire, si elle est absente l'ajoute

element.classList.toggle('une\_classe');

Renvoie true si l'élément possède la classe

element.classList.contains('une\_classe');

Créer de nouveaux noeuds, les ajouter dans le DOM

Création d'un nouvel élément, ici un div

let div = document.createElement('div');

On ajoute cet élément à l'élément body

document.querySelector('body').appendChild(div);

div.innerText = 'Nouvel élément';

Parcourir les noeuds

element.parentElement; renvoi le parent de l'élement

element.previousSibling; renvoi le frere précédent l'élement (que ce soit un élément, texte ou commentaire)

element.previousElementSibling; renvoi le frere précédent l'élement (uniquement élément)

element.nextSibling; renvoi le frere suivant l'élement (que ce soit un élément, texte ou commentaire)

element.nextElementSibling; renvoi le frere suivant l'élement (uniquement élément)

element.childNodes; renvoi les enfants de l'élement (que ce soit un élément, texte ou commentaire)

element.children; renvoi les enfants de l'élement (uniquement élément)

## Les évènements

Les évènements permettent de réagir par rapport aux évènements produits par l'utilisateur, c'est à dire les actions qu'ils produit sur la page HTML. Par exemple cliquer sur un boutton, passer sa souris au dessus d'un élément,...

let boutton = document.querySelector('#btn');

boutton.addEventListener('click', reaction);

boutton.addEventListener('click', function(){

revient au même qu'au dessus

});

function reaction() {

element.innerText = "modification";

}

### Exemple de type évènements

|  |  |
| --- | --- |
| click | Un bouton d'un dispositif de pointage a été appuyé ou relaché sur un élément. |
| dblclick | Un bouton d'un dispositif de pointage a été double-cliqué sur un élément. |
| contextmenu | clique droit de la souris |
| focus | Un élément a acquis le focus. |
| focusout | Un élément perd le focus. |
| fullscreenchange | Un élément est passé en mode plein écran, ou est revenu au mode normal. |
| keydown | Une touche est relâchée. |
| keyup | Une touche est enfoncée. |
| keypress | Une touche est pressée et cette touche produit normalement un caractère (utilisez input à la place). |
| change | Un élément perd le focus et sa valeur a changé depuis l'acquisition du focus. |
| chargingchange | La batterie commence ou finit de charger. |

Voir tous les types d'évènements : <https://developer.mozilla.org/fr/docs/Web/Events>

## Les API JavaScript

Les API sont des fonctionnalités que vous pouvez utiliser pour accéder à des éléments de votre navigateur par exemple. Elles permettent entre autres de gérer les cookies, utiliser la webcam, géolocaliser l’utilisateur, traduire des textes, …

Utiliser la webcam <https://developer.mozilla.org/fr/docs/WebRTC/Prendre_des_photos_avec_la_webcam>

Géolocalisation

<https://developer.mozilla.org/fr/docs/Web/API/Geolocation_API>

Liste des API JS

<https://developer.mozilla.org/fr/docs/Web/API>

## JSON

Exercice

## Plus ou moins

Nous allons créer un jeu qui fait deviner un nombre à l’utilisateur. Le jeu génère un nombre aléatoire entre 1 et 100, et l’utilisateur doit le deviner en 10 tours maximum. Pour chaque tentative, le jeu dit si l’estimation de l’utilisateur est trop ou pas assez élevée.

Penser comme un programmeur

Une des choses les plus difficiles à apprendre en programmation n'est pas la syntaxe, mais comment l'appliquer afin de résoudre un problème réel. Vous devez commencer à penser comme un programmeur — ce qui implique généralement d'examiner les tâches que votre programme doit effectuer, de déterminer les fonctionnalités du code nécessaires à leurs réalisations et comment les faire fonctionner ensemble.

Cela requiert un mélange de travail acharné, d'expérience avec la syntaxe de programmation (de manière générale) et surtout de la pratique — ainsi qu'un peu de créativité. Plus vous allez coder, plus vous aller vous améliorer. On ne peut pas garantir que vous aurez un « cerveau de développeur » en 5 minutes.

Voici comment on peut découper le problème :

1. Générer un nombre aléatoire entre 1 et 100.
2. Stocker le nombre d’essais restant. Commencer par 10.
3. Fournir au joueur le moyen de saisir un nombre.
4. Stocker l'ensemble des propositions de nombres pour que le joueur puisse les consulter.
5. Vérifier si le nombre saisi par le joueur est correct.
6. S'il est correct :
   * 1. Afficher un message de félicitations.
     2. Empêcher que le joueur saisisse de nouveau un nombre.
     3. Afficher un contrôle pour que le joueur puisse rejouer.
7. S'il est faux et que le joueur a encore des tours à jouer :
   * 1. Informer le joueur que sa proposition de nombre est fausse.
     2. Lui permettre d'entrer une nouvelle proposition de nombre.
     3. Décrémenter le nombre d’essais de 1.
8. S'il est faux et que le joueur n'a plus de tours à jouer :
   * 1. Informer le joueur qu'il a perdu et que la partie est finie.
     2. Empêcher que le joueur saisisse de nouveau un nombre.
     3. Afficher un contrôle pour que le joueur puisse rejouer.
9. Une fois le jeu redémarré, s'assurer que la logique du jeu et l'interface utilisateur sont complètement réinitialisées, puis revenir à l'étape 1.

### Références

Cet exercice est repris du site mozzila.org

<https://developer.mozilla.org/fr/docs/Learn/JavaScript/First_steps/A_first_splash>

Le correctif de l’exercice sur le repository du cours

<https://github.com/martini224/EFP_I01/blob/master/code/deviner_un_nombre.html>

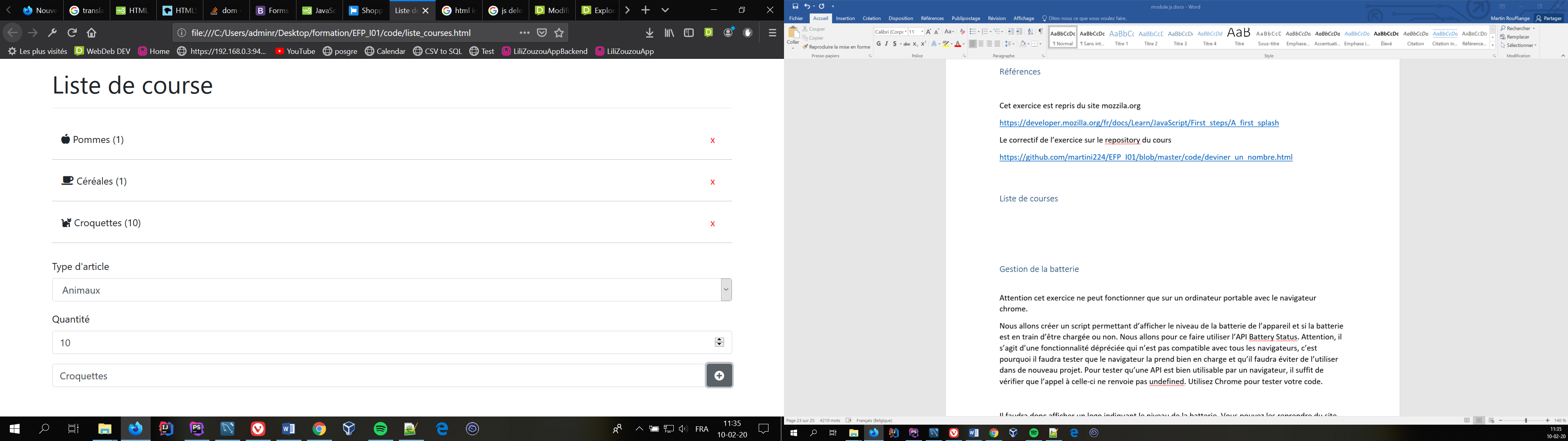
### Liste de courses

Le but de cet exercice sera la création d’une liste des courses à partir d’un formulaire où l’on pourra indiquer le type de l’article à acheter, la quantité et l’intitulé de celui-ci. L’utilisateur se verra proposé une liste de proposition d’intitulé d’article quand il commencera a tapé dans le champ de l’intitulé.

Cette liste est fournie au format JSON, pour chaque catégorie est renseigné un id technique, son intitulé et une liste d’intitulé d’articles faisant partie de la catégorie. Quand aucun type d’article n’est sélectionné, il doit être proposé à l’utilisateur les propositions contenant les lettres qu’il a tapé dans le champ intitulé, et ce pour n’importe quelle catégorie. Quand une catégorie est indiquée, il ne doit alors être proposé que les articles de cette catégorie.

L’utilisateur doit avoir la possibilité d’ajouté un article à la liste via un bouton, et de supprimé un article de la liste via un autre bouton.

Le résultat final pourrait ressembler à la figure ci-dessous :



### Gestion de la batterie

Attention cet exercice ne peut fonctionner que sur un ordinateur portable avec le navigateur chrome.

Nous allons créer un script permettant d’afficher le niveau de la batterie de l’appareil et si la batterie est en train d’être chargée ou non. Nous allons pour ce faire utiliser l’API Battery Status. Attention, il s’agit d’une fonctionnalité dépréciée qui n’est pas compatible avec tous les navigateurs, c’est pourquoi il faudra tester que le navigateur la prend bien en charge et qu’il faudra éviter de l’utiliser dans de nouveau projet. Pour tester qu’une API est bien utilisable par un navigateur, il suffit de vérifier que l’appel à celle-ci ne renvoie pas undefined. Utilisez Chrome pour tester votre code.

Il faudra donc afficher un logo indiquant le niveau de la batterie. Vous pouvez les reprendre du site fontawesome : <https://fontawesome.com/icons?d=gallery&q=battery&m=free>. Si le niveau de batterie est < 25% on affiche le logo battery-empty, si < 50 : battery-quarter, si < 75 battery-half, si < 100 battery-three-quarter et si = 100 battery-full. Vous pouvez également attribuer une couleur différente en fonction du niveau de batterie.

En plus de cela, il faudra écrire le niveau de batterie en pourcent et afficher un logo (par exemple : <https://fontawesome.com/icons/bolt?style=solid> ) quand la batterie est en charge.

Ces données devront être affichée dès le lancement de la page HTML sur lequel votre script va fonctionner, mais également quand l’évènement chargingchange se produit (c’est-à-dire que le chargeur vient d’être branché ou débranché) et l’évènement levelchange quand le niveau de la batterie change.

### Références

L’API Battery Status

<https://developer.mozilla.org/fr/docs/Web/API/Battery_Status_API>

Le correctif

<https://github.com/martini224/EFP_I01/blob/master/code/batterie.html>

# Annexe

## Puissance 4

L’objectif sera de réaliser un puissance 4.

L’énoncé ne reprendra pas à pas la marche à suivre pour y arriver. Vous pouvez trouver la solution sur le repository Git du cours : <https://github.com/martini224/EFP_I01/tree/master/code> . Télécharger le code et ouvrez le fichier puissance\_4.html dans votre navigateur afin de voir le jeu en action.

### Références

Les règles du puissance 4 :

<https://fr.wikipedia.org/wiki/Puissance_4>

Documentation bootstrap (mise en page, secondaire dans ce cours)

<https://getbootstrap.com/docs/4.4/getting-started/introduction/>

Liste des icônes avec fontawesome (de nouveau secondaire dans ce cours)

<https://fontawesome.com/icons?d=gallery&m=free>