# JavaScript – Algos

Il est sans doute possible de raccourcir/simplifier toutes les solutions proposées dans ce doc.

En annexe se trouveront les lignes de codes de chaque exos de manière à pouvoir être copiée/modifiée.

## 0.1 ALGOS-série 1 :

### *00 – Condition number.*

“Create a function that takes two arguments. Both arguments are integers, a and b. Return true if one of them is 10 or if their sum is 10.

Note :

- Don't forget to return the result.”

→ ***Méthodologie :***

- (Si) Vérifier que l'utilisateur entre un nombre (sinon faux).
- (Si) Vérifier que ce nombre est un entier (sinon faux).
- Si le a = 10 **OU** b = 10 **OU** a + b = 10 alors c'est vrai (sinon faux).
- Ne pas oublier d'afficher la réponse + faire des tests.

→ ***Code fonctionnel :***

```
function conditionNumber (a, b) {
if (typeof a === 'number' && typeof b === 'number'){
    if (Number.isInteger(a) && Number.isInteger(b)) {
        if (a === 10 || b === 10 || a + b === 10){
            return true;
        } else {
            return false;
        }
    } else {
        return false;
    }
} else {
    return false;
}
}
console.log(conditionNumber(6, 4)); // true
```

## ***01 – Age in days.***

“Create a function that takes the age in years and returns the age in days.

Note :

- Use 365 days as the length of a year for this challenge.

- Ignore leap years and days between last birthday and now.

- Expect only positive integer inputs.”

→ ***Méthodologie :***

- (Si) Vérifier que l'utilisateur entre un nombre, qu'il soit entier et plus grand que zéro (sinon faux).
- Multiplier le nombre de l'argument (l'âge) par 365.
- Ne pas oublier d'afficher la réponse + faire des tests.

→ ***Code fonctionnel :***

```
function convertAge (age){
    if (typeof age === 'number' && age >= 0 && Number.isInteger(age)){
        return age * 365;
    } else {
        return "Wrong input!";
    }
}
console.log(convertAge(10)); // 3650
```

## 02 – Special sum.

“Create a function that takes a number as an argument.

Add up all the numbers from 1 to the number you passed to the function.

For example, if the input is 4 then your function should return 10 because 1 + 2 + 3 + 4 = 10.

Note :

- Expect any positive number between 1 and 1000.”

### → Méthodologie :

- (Si) Vérifier que l'utilisateur entre un nombre, qu'il soit entier et plus grand que zéro (sinon faux). Ici j'ai utilisé la contraposée.
- Définir que la somme ne vaut rien atm.
- Utiliser une loop pour parcourir tout l'argument.
- Ajouter chaque incrément à la somme définie précédemment.
- Ne pas oublier d'afficher la réponse + faire des tests.

### → Code fonctionnel :

```
function randomNum (num) {
    if (typeof num !== 'number' || num < 0 || !Number.isInteger(num)){
        return "Error, try with a positiv number!";
    } else {
        let sum = 0;
        for (let i = 1; i <= num; i++){
        sum += i;
        }
        return sum;
    }
}
console.log(randomNum(15)); // 120
console.log(randomNum(-12)); // Error, try with a positiv number!
console.log(randomNum("abs")); // Error, try with a positiv number!
```

## 03 – Min & max - array.

“Create a function that takes an array of numbers and return both the minimum and maximum numbers, in that order.

Notes :

- All test arrays will have at least one element and are valid.”

### → *Méthodologie :*

- (Si) Vérifier que l'array ne soit pas vide (sinon invalide).
- (Si) Vérifier que l'array n'ai qu'une valeur, cette valeur est le minimum && le maximum de l'array.
- Sinon, préciser une variable qui représente le minimum parmis toutes les valeurs de l'array (...arr).
- Idem pour le maximum.
- Return l'array [minimum, maximum].
- Ne pas oublier d'afficher la réponse + faire des tests.

### → *Code fonctionnel :*

```
function numberArr(arr) {
    if (arr.length <= 0) {
        return "Invalid array"
    } else if (arr.length === 1){
        return [arr[0], arr[0]];
    } else {
        let min = Math.min(...arr);
        let max = Math.max(...arr);
        return [min, max];
    }
}

console.log(numberArr([1, 9, -7])); // [-7, 9]
console.log(numberArr([2])); // [2, 2]
console.log(numberArr([])); // Invalid array
```

## ***04 – Word is on the loose.***

“A word is on the loose and now has tried to hide amongst a crowd of tall letters!

Help write a function to detect what the word is, knowing the following rules:

- The wanted word is in lowercase.

- The crowd of letters is all in uppercase.

- Note that the word will be spread out amongst the random letters, but their letters remain in the same order.”

→ ***Méthodologie :***

- Créer une variable qui servira de stockage pour le mot recherché.
- Parcourir l'ensemble de lettre à l'aide d'une boucle for..of qui va récolter toutes les lettres minuscules contenue dans l'ensemble de lettre (de 'a' jusque 'z').
- Ajouter ces caractères à la variable de stockage.
- Return la variable de stockage.

→ ***Code fonctionnel :***

```
function findWord (letters){
    let wantedWord = '';

    for (let character of letters){
        if (character >= 'a' && character <= 'z') {
            wantedWord += character;
        }
    }
    return wantedWord;
}
console.log(findWord("UcUNFYGaFYFYGtNUH"));
console.log(findWord("bEEFGBuFBRrHgUHlNFYaYr"));
console.log(findWord("YFemHUFBbezFBYzFBYLleGBYEFGBMENTment"));
```

## *05 – Sort by price.*

“You will be given an array of drinks, with each drink being an object with two properties : name and price.

Create a function that has the drinks array as an argument and

return the drinks objects sorted by price in ascending order.”

### → *Méthodologie :*

- La première fonction prend un tableau d'objets *drinks* comme argument.
- Chaque objet a deux propriétés : *name* & *price*.
- Trier (.sort) les objets : comparer les *prix* (donnée qui détermine le tri dans la réponse).
- Si **a < b** alors return **-1** pour que **a reste devant le b**.
- Si **a > b** alors return **1** pour que **b passe devant le a**.
- Si les deux sont égaux, return 0 et ils ne bougent pas.
- Ne pas oublier d'afficher la réponse + faire des tests.

### → *Code fonctionnel :*

```
function sortedDrink (drinks){
    drinks.sort((first, second) => {
        if (first.price < second.price){
            return -1;
        } else if (first.price > second.price) {
            return 1;
        } else {
            return 0;
        }
    });
    return drinks;
}
const drinks = [
    {name: "lemonade", price: 10},
    {name: "lime", price: 10},
    {name: "coke", price: 8},
    {name: "vodka", price: 17},
    {name: "water", price: 3},
]
    console.log(sortedDrink(drinks));
```

Return :

```
▼ (5) [{…}, {…}, {…}, {…}, {…}]
  ▶ 0: {name: 'water', price: 3}
  ▶ 1: {name: 'coke', price: 8}
  ▶ 2: {name: 'lemonade', price: 10}
  ▶ 3: {name: 'lime', price: 10}
  ▶ 4: {name: 'vodka', price: 17}
```

## ***06 – Chickens, cows & pigs.***

“In this challenge, a farmer is asking you to tell him how many legs can be counted among all his animals.

The farmer breeds three species: chickens = 2 legs, cows = 4 legs & pigs = 4 legs.

The farmer has counted his animals and he gives you a subtotal for each species. You have to implement a function that returns the total number of legs of all the animals.

Notes :

- Don't forget to return the result.

- The order of animals passed is animals(chickens, cows, pigs).

- Remember that the farmer wants to know the total number of legs and not the total number of animals.”

### → *Méthodologie :*

- Vérifier que la valeur des trois arguments soit bien un nombre entier positif (sinon faux).
- Créer une variable pour le totale de patte de chaque animaux comme ceci : nombre de poule * 2 = nombre total de pattes de toutes les poules (faire cela pour les vaches et les cochons).
- Créer une variable (ou return directement) la somme de nos trois variables "total de pattes".

### → *Code fonctionnel :*

```
function legsTot (chickens, cows, pigs){
    if (typeof chickens !== 'number' || !Number.isInteger(chickens) || chickens < 0
    || typeof cows !== 'number' || !Number.isInteger(cows) || cows < 0
    || typeof pigs !== 'number' || !Number.isInteger(pigs) || pigs < 0){
    return "Oops mistake, legs are counted with positive integers!";
} else {
    let totalChickenLegs = chickens * 2;
    let totalCowsLegs = cows * 4;
    let totalPigsLegs = pigs * 4;
    let totalLegs = totalChickenLegs + totalCowsLegs + totalPigsLegs;
    return totalLegs;
}
}
console.log(legsTot(1, 2, 3)); // 22
```

## ***07 – Gamble.***

“Create a function that takes three arguments probability, prize, pay and returns true if *probability * prize > pay,* otherwise return *false*.

To illustrate:

profitableGamble(0.2, 50, 9)

... should yield true, since the net profit is 1 (0.2 * 50 - 9), and 1 > 0.

Notes :

- A profitable gamble is a game that yields a positive net profit

where net profit is calculated in the following manner:

net_outcome = probability_of_winning * prize - cost_of_playing.”

→ ***Méthodologie :***

- Vérifier que la valeur des trois arguments soit bien un nombre (sinon faux).
- Vérifier que la formule donnée soit respectée (sinon faux).
- Ne pas oublier d'afficher la réponse + faire des tests.

→ ***Code fonctionnel :***

```
function gamble(proba, prize, pay) {
    if (isNaN(proba) || isNaN(prize) || isNaN(pay)) {
        return "You have to use number to use this equation!"
    } else {
        if (proba * prize > pay){
            return true;
        } else {
            return false;
        }
    }
}
console.log(gamble(0.7, 40, 28)); // false
```

## *08 – Frames per second (FPS).*

“Create a function that returns the number of frames shown in a given number of minutes for a certain FPS.

Notes :

- FPS stands for "frames per second" and it's the number of frames a computer screen shows every second.”

### → *Méthodologie :*

- Comprendre la formule donnée dans l'exemple de solution attendue :

  arg1 * arg2 * 60 = la valeur des FPS.

- Vérifier que les arguments soient bien des nombres positifs (sinon faux).

- Si ce sont des nombres positifs, il suffit d'appliquer la formule et d'afficher la solution (en faisant toujours plusieurs tests).

### → *Code fonctionnel :*

```
function fps(factor, res){
    if (isNaN(factor) || isNaN(res) || factor < 0 || res < 0){
        return 'Sorry bro, you should use positive numbers to get your FPS'
    } else {
        return factor * res * 60;
    }
}
console.log(fps(1, 1)); // 60
console.log(fps(10, 1)); // 600
console.log(fps(10, 25)); // 15000
```

## ***09 – Fuel's quantity.***

"A vehicle needs 10 times the amount of fuel than the distance it travels. However, it must always carry a minimum of 100 fuel before setting off. Create a function which calculates the amount of fuel it needs, given the distance.

Notes :

- Distance will be a number greater than zero.

- Return 100 if the calculated fuel turns out to be less than 100."

***→ Méthodologie :***

- Vérifier que le nombre de km introduit soit bien un nombre positif.
- Vérifier les différents cas possible :

  - Si les km sont compris entre 0 et 10 (inclus), on return 100.

  - Si les km sont strictement supérieurs à 10, on les multiplie par 10 pour connaitre la quantité de fuel nécessaire.
- Ne pas oublier d'afficher la réponse + faire des tests.

***→ Code fonctionnel :***

```
function qtyFuel (km) {
    if (isNaN(km) || km < 0){
        return "To use this function you have to use a positive integer!"
    } else {
        if (km > 0 && km <= 10){
            return 100;
        } else {
            return km * 10;
        }
    }
}
console.log(qtyFuel(15)); // 150
console.log(qtyFuel(23.5)); // 235
console.log(qtyFuel(3)); // 100
```

# 0.2 ALGOS-série 2 :

## *00 – Count true.*

“Create a function which returns the number of true values there are in an array.

Notes :

- Return 0 if given an empty array.

- All array items are of the type bool (true or false).”

### → *Méthodologie :*

- Introduire une variable qui me servira à compter le nombre de *true*.
- A l'aide d'un *if*, déterminer que si l'array est vide, on return 0.
- Dès que l'array possède minimum un élément, on la parcourt avec une boucle for dans le but de trouver toutes les valeurs === *true* et les ajouter à notre variable de comptage.
- On return le tout et c'est gagné.

### → *Code fonctionnel :*

```
function countTrue (arr) {
let count = 0;
    if (arr.length === 0){
        return 0
    } else {
        for (let i = 0; i < arr.length; i++){
        if (arr[i] === true){
            count++;
        }
        }
            return count;
        }
}
console.log(countTrue([true, false, false, true, false])); //2
console.log(countTrue([false, false, false, false])); //0
console.log(countTrue([])); //0
```

## ***01– Board game placement.***

“In a board game, a piece may advance 1-6 tiles forward depending on the number rolled on a six-sided die. If you advance your piece onto the same tile as another player's piece, both of you earn a bonus. Can you reach your friend's tile number in the next roll? Create a function that takes your position **a** and your friend's position **b** and returns a boolean representation of whether it's possible to earn a bonus on any die roll.

Notes :

- You cannot move backward.

- If you are already on the same tile, return false, as you would be advancing away.

- Expect only positive integer inputs.”

→ ***Méthodologie :***

- Vérifier que les deux nombres soient des entiers positfs (sinon faux)
- Vérifier les 2 premières notes de l'énoncé (sinon faux) :

  - **b** ne peut pas être **plus petit** que **a** (pas de déplacement négatif).

  - **a** et **b** ne peuvent pas être **égaux** (déjà sur la même case).
- Prendre tous les cas ou la différence entre **b** et **a** est **<= 6**, return true.

→ ***Code fonctionnel :***

```
function position (a, b){
    if (isNaN(a) || !Number.isInteger(a) || a < 0
    || isNaN(b) || !Number.isInteger(b) || b < 0){
        return 'Use a positive integer!'
    }
    if (b < a || a === b){
        return false;
    }
    let difference = b - a;
    if (difference <= 6){
        return true;
    }
    return false;
}
console.log(position(3, 7)); // true
console.log(position(2, 2)); // false
console.log(position(5, 3)); // false
```

## *02 – Number length.*

“Create a function that will return an integer number corresponding to the amount of digits in the given integer num.”

→ ***Méthodologie :***

- Vérifier qu'il s'agisse d'un nombre entier.
- Transformer le nombre introduit en string afin de pouvoir compter sa longueur (.toString() into .length).
- Essayer différentes valeurs quand la réponse est affichée.

→ ***Code fonctionnel :***

```
function number (num){
    if (isNaN(num) || !Number.isInteger(num)){
        return `Use an integer!`
    }
    return num.toString().length;
}
console.log(number(1000)); // 4
console.log(number(12)); // 2
console.log(number(1305981031)); // 10
console.log(number(0)); // 1
```

## 03 – From Object To Array.

“Write a function that converts an object into an array, where each element represents a key-value pair in the form of an array.

Notes :

- Return an empty array if the object is empty.”

### → Méthodologie :

- Vérifier qu'un objet vide retourne une array vide également.
- Return & transform tous les objets en tableaux grâce à Object.entries().
- Essayer différents objets pour être certain que cela fonctionne!

### → Code fonctionnel :

```
function objToArray (obj){
    if (obj.length === 0){
        return [];
    }
    return Object.entries(obj);
}
console.log(objToArray({ a: 1, b: 2 })); // [["a", 1], ["b", 2]]
console.log(objToArray({ shrimp: 15, tots: 12 })); // [["shrimp", 15], ["tots", 12]]
console.log(objToArray({})); // []
```

## 04 – Multiply game.

“Create a function that takes two numbers as arguments (num, length) and returns an array of multiples of num until the array length reaches length.

Notes :

- Notice that num is also included in the returned array.”

### → Méthodologie :

- Déclarer une variable qui contiendra l'array de la réponse.
- Vérifier que les nombres utilisés soient des entiers.
- A l'aide d'une boucle et d'un push, ajouter dans la réponse toutes les nombres demandés (num * i).
- Return le resultat et s'amuser avec d'autres essais.

### → Code fonctionnel :

```
function numToArray (num, length){
    let result = [];
    if (isNaN(num) || isNaN(length) || !Number.isInteger(num) || !Number.isInteger(length)){
        return error;
    }
    for (i = 1; i <= length ; i++){
        result.push(num * i);
    }
    return result
}
console.log(numToArray(7, 5)); // [7, 14, 21, 28, 35]
console.log(numToArray(12, 10)); // [12, 24, 36, 48, 60, 72, 84, 96, 108, 120]
console.log(numToArray(17, 6)); // [17, 34, 51, 68, 85, 102]
```

## *05 – Trailing and leading zeros.*

“Create a function that takes in a number as a string n and returns the number without trailing and leading zeros.

- Trailing Zeros are the zeros after a decimal point which don't affect the value (e.g. the last three zeros in 3.4000 and 3.04000).

- Leading Zeros are the zeros before a whole number which don't affect the value (e.g. the first three zeros in 000234 and 000230).

Notes :

- Return a string.

- If you get a number with .0 on the end, return the integer value (e.g. return "4" rather than "4.0").

- If the number is 0, 0.0, 000, 00.00, etc... return "0".”

→ ***Méthodologie :***

- Vérifier qu'on ne garde que les nombres positifs.
- Utiliser parseFloat pour transformer l'argument en nombre décimal.
- Transformer le nombre décimal en string (.toString()) afin de faire disparaitre les zéros inutiles.

→ ***Code fonctionnel :***

```
function delZero (num) {
    if (isNaN(num) || num <0){
        return `Error, we need a positive number!`
    }
    return parseFloat(num).toString();
    }
console.log(delZero("230.000")); // 230
console.log(delZero("00402")); // 402
console.log(delZero("03.1400")); // 3.14
console.log(delZero("30")); // 30
```

## *06 – Array organization.*

“In this challenge you will be given an array similar to the following:

[[3], 4, [2], [5], 1, 6]. In words, elements of the array are either an integer or an array containing a single integer. We humans can clearly see that this array can reasonably be sorted according to **"the content of the elements"** as: [1, [2], [3], 4, [5], 6]. Create a function that, given an array similar to the above, sorts the array according to the **"content of the elements"**.

Notes :

- To reiterate, elements of the array will be either integers or arrays with a single integer”

### → *Méthodologie :*

- Utilisation de .map() pour obtenir une nouvelle version du tableau contenu dans une nouvelle variable (*downloadValue*).
- Les éléments du tableau sont modifiés en fonction du callback :

  - item -> focus tous les éléments du tableau arr

  - vérifier si l'élément (*item*) est un tableau avec Array.isArray(*item*) :

  -- si oui, on prend le premier élément du tableau

  -- si non, on prend directement l'élément lui-même
- Return un tableau à deux éléments : **value** & **item**.
- On trie maintenant les valeurs dans *downloadValue* car si **a – b** est **négatif** a vient avant b (et inversément).
- Return final, nouveau map car nouveau tableau créé à partir de *downloadValue.* Le but étant de ne garder que item[1] qui est l'élément orignal après transformation.

### → *Code fonctionnel :*

```
function sortArray (arr){
    let downloadValue = arr.map(item => {
        let value = Array.isArray(item) ? item[0] : item;
        return [value, item];
    });
    downloadValue.sort((a, b) => a[0] - b[0]);
    return downloadValue.map(item => item[1]);
}
console.log(sortArray([4, 1, 3])); // [1, 3, 4]
console.log(sortArray([[4], [1], [3]])); // [[1], [3], [4]]
console.log(sortArray([4, [1], 3])); // [[1], 3, 4]
console.log(sortArray([[4], 1, [3]])); // [1, [3], [4]]
console.log(sortArray([[3], 4, [2], [5], 1, 6])); // [1, [2], [3], 4, [5], 6]
```

## *07 – Calculator.*

“Create a function that takes two numbers and a mathematical operator + - / * and will perform a calculation with the given numbers.

Notes :

- If the input tries to divide by 0, return: "Can't divide by 0!"”

### → *Méthodologie :*

- Vérifier que les données soient bien deux nombres et un signe d'opération mathématiques (sinon faux).
- Vérifier que si l'opérateurn est **divisé** et que le **diviseur** est **zéro** alors "Can't divide by 0!”.
- Faire un **IF** pour chaque opérateur (ou un switch) et ajouter l’opération adéquate.
- Afficher la réponse et effectuer quelques tests.

### → *Code fonctionnel :*

```
function calc (num1, operator, num2) {
    if (isNaN(num1) || isNaN(num2) ||
    operator !== "+" && operator !== "-" && operator !== "/" && operator !== "*"){
        return "Can't be calculated. Check values and mathematical operator. Try again!"
    }
    if (operator === '/' && num2 === 0) {
        return "Can't divide by 0!"
    }
    switch (operator) {
        case '+' :
            return num1 + num2;
        case "-" :
            return num1 - num2;
        case "*" :
            return num1 * num2;
        case "/" :
            return num1 / num2;
    }
}
console.log(calc(2, "+", 2)); // 4
console.log(calc(2, "*", 2)); // 4
console.log(calc(4, "/", 2)); // 2
```

## *08 – Country's area.*

"Create a function that takes a country's name and its area as arguments and returns the area of the country's proportion of the total world's landmass.

Notes :

- The total world's landmass is 148,940,000 [Km^2]

- Round the result to two decimal places."

### → *Méthodologie :*

- Vérifier que le pays est une "string" et que le nombre de km est un nombre positif.
- Créer le pourcentage (ici dans une variable) :

  - Surface du PAYS / Surface mondiale * 100 (formule du pourcentage)
- Return la petite phrase demandée avec les bonnes données.

### → *Code fonctionnel :*

```
function areaOfCountry (country, area){ // How to verify if it's a real country without list?
    if (typeof country !== 'string' || isNaN(area) || area < 0){
        return `Please type a real country with his real area (in km²) to process the code.`
    }
    let percentage = (area / 148940000)*100
    return `${country} is ${percentage.toFixed(2)}% of the total world's landmass`
}
console.log(areaOfCountry("Russia", 17098242)); // "Russia is 11.48% of the total world's landmass"
console.log(areaOfCountry("USA", 9372610)); // "USA is 6.29% of the total world's landmass"
console.log(areaOfCountry("Iran", 1648195)); // "Iran is 1.11% of the total world's landmass"
```

## ***09 – reverseWords.***

“Given an input string, reverse the string word by word, the first word will be the last, and so on.

Notes :

- A word is defined as a sequence of non-space characters.

- The input string may contain leading or trailing spaces.

However, your reversed string should not contain leading or trailing spaces.

- You need to reduce multiple spaces between two words to a single space in the reversed string.”

### → *Méthodologie :*

- Supprimer les espaces inutiles avec .trim().
- Créer une variable qui représente **TOUS** les mots de la phrase en veillant à bien .split(' ') l'argument pour laisser les mots entiers et j'applique un filtre pour que les mots soient plus long que 0 caractère afin d'enlever les espaces inutiles.
- Je créé une variable dans laquelle je .reverse() tous les mots obtenu précédemment.
- Quand les mots sont .reverse() je n’oublie pas de les .join(‘ ‘) dans le return pour reformer une **string**.

### → *Code fonctionnel :*

```
function reverseWords (string) {
    string = string.trim();
    let everyWords = string.split(' ').filter(word => word.length > 0);
    let reverseEveryWords = everyWords.reverse();
    return reverseEveryWords.join(' ');
}
console.log(reverseWords(" the sky is blue")); // "blue is sky the"
console.log(reverseWords("hello   world!  ")); //  "world! hello"
console.log(reverseWords("a good example")); // "example good a"
```

# 0.3 ALGOS-série 3 :

## *00 – Oddish or Evenish.*

“Create a function that determines whether a number is Oddish or Evenish. A number is Oddish if the sum of all of its digits is odd, and a number is Evenish if the sum of all of its digits is even. If a number is Oddish, return "Oddish". Otherwise, return "Evenish".

For example :

- oddishOrEvenish(121) should return "Evenish", since 1 + 2 + 1 = 4.

- oddishOrEvenish(41) should return "Oddish", since 4 + 1 = 5.”

### → *Méthodologie :*

- Créer une variable qui me permet de traiter chaque chiffre de mon nombre (*digits*). J'utilise un .string() puis un .split('').
- Définir la somme de ces chiffres qui pour l'instant est nulle.
- Utiliser une boucle pour parcourir tous les chiffres du nombre et les ajouter un à un à la somme.
- Vérifier si la somme est paire ou non avec un modulo et return les valeurs demandées.
- Personnellement j'ai affiché : Oddish/Evenish, la somme des chiffres et la preuve de la parité avec le modulo.

### → *Code fonctionnel :*

```
function oddOrEven (num){
    let digits = num.toString().split('');
    let sum = 0;
    for (let i = 0; i<digits.length; i++){
        sum += parseInt(digits[i]);
    }
    if (sum % 2 === 0){
        console.log("Evenish");

    } else {
    console.log("Oddish");
    }
    console.log(digits.join(' + ')+ ' = ' + sum);
    console.log(sum + ` % 2 = ` + sum%2);
}
oddOrEven(43); // Oddish // 4 + 3 = 7 // 7 % 2 = 1
oddOrEven(373); // Oddish // 3 + 7 + 3 = 13 // 13 % 2 = 1
oddOrEven(4433); // Evenish // 4 + 4 + 3 + 3 = 14 // 14 % 2 = 0
```

## 01 – Groceries's price.

"Create a function that takes an array of objects (groceries) which calculates the total price and returns it as a number. A grocery object has a product, a quantity and a price."

### → Méthodologie :

- Créer la variable totalPrice afin de stocker le prix final (elle est à 0 atm).
- Utiliser une boucle for pour parcourir tous les éléments du tableau *groceries*.
- Créer la variable qui représente un objet du tableau à un instant donné (*item*).
- Calculer le *totalPrice* en multipliant le nombre d'item par le prix unitaire.
- Return la valeur de *totalPrice* avec 2 décimales (plus courant pour un prix)
- De beaux exemples et c'est fini !

### → Code fonctionnel :

```
function getTotalPrice (groceries){
    let totalPrice = 0;
    for (i = 0; i < groceries.length; i++){
        let item = groceries[i];
        totalPrice += parseFloat(item.quantity * item.price);
    }
    return totalPrice.toFixed(2);
}
console.log(getTotalPrice([{ product: "Milk", quantity: 1, price: 1.50 }])); // 1.50
console.log(getTotalPrice([{ product: "Milk", quantity: 3, price: 1.50 }])); // 4.50
console.log(getTotalPrice([
    { product: "Milk", quantity: 1, price: 1.50 },
    { product: "Eggs", quantity: 12, price: 0.10 },
    { product: "Bread", quantity: 2, price: 1.60 },
    { product: "Cheese", quantity: 1, price: 4.50 }
])); // 10.40
console.log(getTotalPrice([
    { product: "Chocolate", quantity: 1, price: 0.10 },
    { product: "Lollipop", quantity: 1, price: 0.20 }
])); // 0.30
```

## ***02 – Reverse Odd Length Words.***

“Given a string, reverse all the words which have odd length. The even length words are not changed.

Notes :

- There is exactly one space between each word and no punctuation is used.”

### → *Méthodologie :*

- Créer une variable pour réceptionner les mots de la phrase après l’avoir **.split(‘ ‘)**.
- Parcourir les mots à l’aide d’une boucle for.
- Si le mot parcouru est impair en caractère (pas pair) alors on **.split(‘’).reverse().join(‘’)** afin de séparer le mot en suite de caractères qu’on inverse et qu’on joint à nouveau ensemble.
- On return les mots sans oublier de les joindre pour ravoir une *string*.
- On essaie et ça fonctionne.

### → *Code fonctionnel :*

```
function reverseWord (string) {
    let words = string.split(' ');

    for (let i = 0; i < words.length; i++){
        if (words[i].length % 2 !== 0){
            words[i] = words[i].split('').reverse().join('');
    }
}
return words.join(' ');
}
console.log(reverseWord("Bananas")); //
console.log(reverseWord("One two three four")); // "enO owt eerht four"
console.log(reverseWord("Make sure uoy only esrever sdrow of ddo length")); // "Make sure you only reverse words of odd length"
```

## 03 – Smooth senteces.

“ Carlos is a huge fan of something he calls smooth sentences. A smooth sentence is one where the last letter of each word is identical to the first letter the following word (and not case sensitive, so "A" would be the same as "a"). The following would be a smooth sentence "Carlos swam masterfully" because "Carlos" ends with an "s" and swam begins with an "s" and swam ends with an "m" and masterfully begins with an "m". Create a function that determines whether the input sentence is a smooth sentence, returning a boolean value true if it is, false if it is not.

Notes :

- The last and first letters are case insensitive.

- There will be no punctuation in each sentence.”

### → *Méthodologie :*

- Créer une variable pour stocker tous les mots de la phrase en minuscule (**.toLowerCase().split(‘ ‘)**).
- Parcourir les mots à l’aide d’une boucle for.
- Créer la notion de mot actuel et mot suivant en établissant deux variables bien spécifiques (*words[i] et words[i+1]*).
- Vérifier avec un si que la dernière lettre du mot actuel est identique (ou non) à la première lettre du mot suivant.
- Return en adéquation avec l’énoncé.

### → *Code fonctionnel :*

```
function smoothSentences (sentence) {
    let words = sentence.toLowerCase().split(' ')
    for (i = 0; i < words.length -1 ; i++){
        let currentWord = words[i];
        let nextWord = words[i + 1];
        if (currentWord[currentWord.length - 1] !== nextWord[0]){
            return false
        }
    }
    return true;
}
console.log(smoothSentences("Marta appreciated deep perpendicular right trapezoids")); // true
console.log(smoothSentences("Someone is outside the doorway")); // false
console.log(smoothSentences("She eats super righteously")); // true
```

## *04 – Function 7Seven.*

“Create a function that takes an array of numbers and return **"Boom!"** if the digit 7 appears in the array. Otherwise, return **"there is no 7 in the array"**.”

### → *Méthodologie :*

- A l’aide d’une boucle for...of je parcours tous les éléments de numbers.
- Je créer une variable pour stocker mon élément en « *string* ».
- Je vérifie que cette variable incluse ou non la valeur *string* « 7 ».
- Je return en adéquation avec l’énoncé.

### → *Code fonctionnel :*

```
function seven (numbers) {
for (let num of numbers){
    let numString = num.toString();
    if (numString.includes("7")) {
        return "Boom!";
    }
}
return "There is no 7 in the array";
}
console.log(seven([1, 2, 3, 4, 5])); // "there is no 7 in the array"
console.log(seven([1, 2, 7, 4, 5])); // "Boom!"
console.log(seven([10, 20, 30, 70])); // "Boom!"
```

## *05 – Celsius & Fahrenheit.*

“Create a function that converts Celsius to Fahrenheit and vice versa.

Notes :

- Round to the nearest integer.

- If the input is incorrect, return "Error".

- Look on Google how to convert F to C and vice-versa”

### → *Méthodologie :*

- Je déclare la variable **unit** qui sera mon unité donc je ne prends que les deux dernières caractères de mon argument *temp*. Je n’oublie pas de l’uppercase pour me faciliter la tâche pour la suite.
- Je fais de même avec la valeur de la température (**value**), je ne prends que les caractères qui vont de l’index 0 jusqu’aux deux derniers caractères. Je parseFloat pour que ce soit un nombre décimal.
- Je vérifie que ma **value** est un nombre, que ma **unit** est soit en degré soit en fahrenheit (grâce à l’unité). Si ce n’est pas le cas je return « *Error* ».
- Si c’est le cas je fais deux cas :

  - °C → °F = (valeur * 9/5) + 32. Stocké dans une variable et arrondie.

  - °F → °C = (valeur - 32) * 5/9. Idem

### → *Code fonctionnel :*

```
function farenCels(temp) {
    let unit = temp.slice(-2).toUpperCase();
    let value = parseFloat(temp.slice(0, -2).trim());

if (isNaN(value) || (unit !== "°C" && unit !== "°F")){
    return "Error!";
} else {
    if (unit === "°C"){
        let fahrenheit = (value * 9/5) + 32;
        return Math.round(fahrenheit) + "°F";
    } else {
        let celsius = (value - 32) * 5/9;
        return Math.round(celsius) + "°C";
    }
}
}
console.log(farenCels("35°C")) // → "95°F"
console.log(farenCels("19°F")) // → "-7°C"
console.log(farenCels("33")) // → "Error"
```

## *06 – Function 7Seven.*

“Given what is supposed to be typed and what is actually typed, write a function that returns the broken key(s). The function looks like:

Notes :

- Broken keys should be ordered by when they first appear in the sentence.

- Only one broken key per letter should be listed.

- Letters will all be in lower case. ”

### → *Méthodologie :*

- Création d’un tableau vide pour stocker les **broken keys**.
- On parcourt les caractères de *expected* (le texte attendu) avec une boucle for.
- On vérifie auprès de *expected* et *typed* quels sont les caractères qui diffèrent afin de les **.push** dans notre variable prévue à cet effet.
- Il faut également vérifier, dans la condition, que ce qu’on veut *.push* n’est pas déjà inclus dans la variable !
- Une fois que c’est fait, on peut return la variable afin d’observer les **broken keys**.

### → *Code fonctionnel :*

```
function findBrokenKeys (expected, typed){
    let brokenKeys = [];
    for (let i = 0; i < expected.length; i++){
        if (expected[i] !== typed[i] && !brokenKeys.includes(expected[i])){
            brokenKeys.push(expected[i]);
        }
    }
    return brokenKeys;
}
console.log(findBrokenKeys("happy birthday", "hawwy birthday")) // ["p"]
console.log(findBrokenKeys("starry night", "starrq light")) // ["y", "n"]
console.log(findBrokenKeys("beethoven", "affthoif5")) // ["b", "e", "v", "n"]
```

# 0.4 ALGOS-advanced :

## *00 – FizzBuzz function.*

“Create a functions that takes a num argument. You should then console.log all numbers from 1 to num. But here’s the catch : multiple of 3 should print “Fizz” and multiples of 5 should print “Buzz”.

Lastly if the number is multiple of 3 and 5, it should print FizzBuzz

Notes :

- Your code should be modular. You must be able to pass any numbers as n and the code should still work.”

### → *Méthodologie :*

- Vérifier que l'argument soit bien un nombre entier.
- A l'aide d'une boucle, parcourir tous les nombres allant de 1 jusqu'au nombre (i).
- Préciser les différents cas avec un If :

  - Si i est multiple de 3 && de 5 -> "FizzBuzz"

  - Si i est multiple de 3 -> "Fizz"

  - Si i est multiple de 5 -> "Buzz"
- Afficher sa réponse et effectuer des tests.

### → *Code fonctionnel :*

```
function fzbz (num){
if (isNaN(num) || num <= 0 || !Number.isInteger(num)){
    return "Error, use a positive integer!";
} else {
    for (let i = 1; i <= num; i++){
        if (i % 3 === 0 && i % 5 === 0) {
            console.log("FizzBuzz");
        } else if (i % 3 === 0) {
            console.log("Fizz");
        } else if (i % 5 === 0){
            console.log("Buzz");
        } else {
            console.log(i);
        }
    };
}
}
fzbz(15);
```

## ***01 – RansomNote.***

“Write a function called ransomNote which takes two parameters : noteText and magazineText. The goal is : with all the words of magazineText, you should be able to create the noteText. The function should return true if we are able to write the noteText with the words from magazineText or false if we can't.”

### → *Méthodologie :*

- Dans deux nouvelles variables, je *.split(' ')* pour transformer mes deux arguments en suite de mots.
- Je crée des objets qui vont stocker combien de fois un mot apparaît dans chaque argument.
- Première boucle pour parcourir *noteText*, on incrémente le compteur de 1 dans *noteCout*. (Idem avec *magazineText*).
- Dernière boucle, on parcourt *noteCount* si le mot apparaît plus de fois dans la note que dans le magazine, return "false". Si le mot n'existe pas c'est le "|| 0" qui prend le relais.
- Finir avec un return **true** car **false** est énoncée ci-dessus.
- Essayer en attribuant une valeur aux arguments!

### → *Code fonctionnel :*

```
function ransomNote (noteText, magazineText){
    let noteWords = noteText.split(' ');
    let magazineWords = magazineText.split(' ');
    let noteCount = {};
    let magazineCount = {};

    for (let word of noteWords) {
        noteCount[word] = (noteCount[word] || 0) + 1;
    }

    for (let word of magazineWords) {
        magazineCount[word] = (magazineCount[word] || 0) + 1;
    }

    for (let word in noteCount) {
        if (noteCount[word] > (magazineCount[word] || 0)) {
            return false;
        }
    }
    return true;
}
```

## 02 – Palindrome.

“A palindrome is any word or phrase that spells the same way both backward and forward. For example : race car is a palindrome. (don't take spaces, periods, dots, etc in consideration). Other examples : Madam, I'm Adam, radar, kayak, etc. You should create a function **isPalindrome** that takes a parameter text and returns a boolean indicating if the provided text is a palindrome or not.

Note :

- You should be able to to ignore all special characters, spaces, dots, periods, etc. and be case insensitive”

### → *Méthodologie :*

- Création du variable qui représentera mon texte entièrement nettoyé (voir la note de l'énoncé). Les espaces et caractères spéciaux sont nettoyé grâce à la Regex `replace(/[^a-zA-Z]/g, "")` sans oublier de forcer les minuscules avec *.toLowerCase()*.
- Création d'une variable qui contiendra le texte retourné :

  - .split('') pour cibler chaque caractères.

  - .reverse() pour retourner tous les caractères (inversion).

  - .join('') pour que les caractères reforment un mot.
- Return lorsque le text nettoyé === le texte retourné.

### → *Code fonctionnel :*

```
function isPalindrome (text){
let cleanText = text.replace(/[^a-zA-Z]/g, "").toLowerCase();
let reverseText = cleanText.split('').reverse().join('');
return cleantText === reverseText;
}
console.log(isPalindrome("hello"));
```

## ***03 – Caesar Cipher.***

“Caesar Cipher is a way of "encrypting" a text that, supposably, Julius Caesar used in his communication. We switch every letter with, for example, two letters before in the alphabet. The Caesar cipher function will take two parameters str and num; str represents the text we want to cipher and num de amount of letters you want to go forward or backward (using negative numbers)

Notes :

- Make it case incensitive.

- You should be able to loop back or forward into the alphabet (if you have to go 3 from z, it would be c)”

### → *Méthodologie :*

- Forcer la *string* à être en *LowerCase*.
- Créer une variable et *.split('')* en caractère la *string*.
- Stocker le résultat dans un élément vide.
- Créer une variable alphabet pour permettre de s'y déplacer facilement.
- On crée une boucle pour parcourir chaque caractère de *str*.
- Premièrement il faut vérifier si le caractère trouvé est une lettre

  - Si oui je déclare une nouvelle variable qui me donne l'index de chaque caractère dans l'alphabet.

  - Je trouve le nouvel index en ajoutant le *num* à l'index actuel et j'applique un modulo sur la taille de l'alphabet pour ne pas être dans le vide.
- Il faut gérer les indexs négatifs, on a juste à ajouter la longueur de l'alphabet pour garder une valeur positive.
- On ajoute la lettre chiffrée à la variable *result* prévue à cet effet.
- Le *else* est utilisé pour gérer les caractères non alphabétiques. Si ce n'est pas une lettre, il est ajouté au *result* sans être modifié.
- Return le result.
- On essaie avec plusieurs exemple!

→ *Code fonctionnel :*

```
function caesarCipher (str, num){
    str = str.toLowerCase();
    let charArray = str.split('');
    let result = '';

    const alphabet = 'abcdefghijklmnopqrstuvwxyz';

    for (let i = 0; i < charArray.length; i++){
        let char = charArray[i];
        if (alphabet.includes(char)){
            let index = alphabet.indexOf(char);
            let newIndex = (index + num) % alphabet.length;
            if (newIndex < 0){
                newIndex += alphabet.length;
            }
            result += alphabet[newIndex];
        } else {
            result += char;
        }
    }
    return result;
}
console.log(caesarCipher("hello world", 3));  // khoor zruog
console.log(caesarCipher("khoor zruog", -3)); // hello world
console.log(caesarCipher("abc xyz", 5));     // fgh cde
console.log(caesarCipher("z", 1));            // a
```

## ***04 – ReverseCharacter.***

“Create a function that takes a string parameter and return another string with all the words inversed.

Notes :

- The order of the words doesn't change, just the letters of each word”

### → *Méthodologie :*

- Forcer la *string* à être en *LowerCase*.
- Nouvelle variable pour *.split(' ')* la *string* en MOTS.
- Créer une variable vide dans laquelle nous allons stocker les mots retournés.
- A l'aide d'une boucle parcourir l'ensemble des mots de la variable.
- Appliquer un *.split('')* caractères, *.reverse()* et *.join('')*
- *.push()* le mot retourné dans la variable prévue à cet effet.
- Ne pas oublier de *.join(' ')* les mots entre-eux pour pouvoir return une *string*!
- On essaie avec plusieurs exemples.

### → *Code fonctionnel :*

```
function reverseWords (string) {
    string = string.toLowerCase();
    let words = string.split(' ');
    let reverseWords = [];

    for (i=0 ; i< words.length; i++){
    let word = words[i];
    let reverseWord = word.split('').reverse().join('');
    reverseWords.push(reverseWord);
    }

    let result = reverseWords.join(' ');
    return result;
}
console.log(reverseWords('This is a string of words')); // Siht si a gnirts fo sdrow
```

## *05 – ReverseArrayWithoutReverse.*

“Create a function that takes an array and reverses it.

Notes :

- Don't use reverse()

- Don't create a new array and push elements to it.”

### → *Méthodologie :*

- Créer une variable qui va accueillir notre array retournée
- Utiliser une boucle pour parcourir l'array à l'envers (du dernier item au premier)
- Ajouter chaque item croisé dans la variable prévue pour.
- Return notre variable de réponse.
- Effectuer plusieurs essais !

### → *Code fonctionnel :*

```
function reverseArray (array){
    let reversed = [];

    for (let i=array.length - 1; i>=0 ; i--){
        reversed.push(array[i]);
    }
    return reversed;
}
console.log(reverseArray([1, 2, 3, 4, 5])) // [5, 4, 3, 2, 1]
console.log(reverseArray(['apple', 'banana', 'cherry', 'date'])) // ['date', 'cherry', 'banana', 'apple'];
console.log(reverseArray([{ name: 'Alice', age: 30 },{ name: 'Bob', age: 25 },{ name: 'Charlie', age: 35 }]));
// [{name: 'Charlie', age: 35},{name: 'Bob', age: 25}, {name: 'Alice', age: 30}]
```

## ***06 – FindSumPair.***

“Write a function that takes as argument an array of numbers numArray and the sum we want to obtain. Your function should return every pair of numbers from numArray that adds up to the 'sum'.

Notes :

- The result should be an array of arrays.

- Any number in the array can be used in multiple pairs (look at the "4" in the exemple below) ”

### → *Méthodologie :*

- Créer une variable prête à accueillir notre array de réponse.
- Parcourir l'array donnée (*numArray*) à l'aide d'une boucle for.
- Boucle intérieure pour trouver le deuxième nombre de la pair. Index + 1 pour éviter de tester le même nombre 2 fois dans une seule paire.
- Si le nombre de la première boucle additionné au nombre de la deuxième boucle === *sum* **alors** on push le résultat dans l'array de réponse.
- On return le résultat pour découvrir toutes les paires de nombres qui ont été push dans la réponse.

### → *Code fonctionnel :*

```
function sumArray (numArray, sum){
    let result = [];
    for (let i = 0; i < numArray.length; i++){
        for (let j = i + 1; j <numArray.length; j++){
            if (numArray[i] + numArray[j] === sum){
                result.push([numArray[i],numArray[j]]);
            }
        }
    }
    return result;
}
console.log(sumArray([1, 2, 8, 13, 12, 9], 14));
```

## *07 – Fibonacci.*

“Fibonacci sequence starts with 1 and 1 and the next numbers are always the sum of the last two numbers. So... Here is the sequence : 1 1 2 3 5 8 13 21 34 … Write a function that takes a num number and returns an array with the num first elements of the Fibonacci sequence.

Notes :

- Recursion would be the best way to solve it but there is another simple way.”

### → *Méthodologie :*

- Créer une array vide qui servira de stockage pour notre réponse.
- A l'aide d'une boucle for parcourir tous les nombres de 0 jusqu'au nombre rentré (*num*).
- Si l'index vaut 0 et 1, il faut push la solution 1 car dans une suite de Fibonacci les deux premières valeurs sont toujours 1 et 1.
- Si l'index ne vaut pas 0 || 1 (else) :

  - J'établis une nouvelle variable qui représente le prochain nombre qui est la somme des deux nombres précédents (le principe même de la suite de Fibonacci)
- Je push cette nouvelle variable dans mon array de réponse.
- Je return la réponse et essaie avec plein de valeurs!

### → *Code fonctionnel :*

```
function fibonacci (num){
    let fibArray = [];
    for (i = 0; i < num; i++){
        if (i === 0 || i === 1){
            fibArray.push(1);
        } else {
            let nextNumber = fibArray[i-1] + fibArray[i-2];
            fibArray.push(nextNumber);
        }
    }
    return fibArray;
}
console.log(fibonacci(4)); // [1, 1, 2, 3]
console.log(fibonacci(9)); // [1, 1, 2, 3, 5, 8, 13, 21, 34]
console.log(fibonacci(6)); // [1, 1, 2, 3, 5, 8]
```

# 0.5 ALGOS-optional-oneLiners :

## 01 – Remove Duplicates.

"Create a function that remove duplicates from an array."

→ ***Méthodologie :***

- new Set(arr) => Créer un nouvel objet en ne gardant que les valeurs uniques à partir de l'array
- [... arr] => transforme le Set en array

→ ***Code fonctionnel :***

```
const removeDuplicates = (arr) =>[...new Set(arr)]

console.log(removeDuplicates([4, 9, 5, 1, 3, 2, 4, 1, 8]));  Array(7) [ 4, 9, 5, 1, 3, 2, 8 ]
console.log(removeDuplicates(["hello", "world", "goodbye", "world"]));  [ 'hello', 'world', 'goodbye' ]
console.log(removeDuplicates([true, true, false, true, true, false]));  [ true, false ]
```

## 02 – Capitalize.

"Create a function that capitalizes a string."

→ ***Méthodologie :***

- Sélectionner le premier index, uppercase, ajouter le reste avec slice() et préciser en lowercase

→ ***Code fonctionnel :***

```
const capitalize = (string) => string[0].toUpperCase() + string.slice(1).toLowerCase();

console.log(capitalize("belgium"));  'Belgium'
console.log(capitalize("brazil"));  'Brazil'
console.log(capitalize("congo"));  'Congo'
```

## 03 – The Days Between.

"Find the days between 2 given days"

→ ***Méthodologie :***

- Arrondire la différence des deux dates (beurk les décimaux pour parler de jours).
- Diviser par le nombre de millisecondes qu'il y a dans un jour :
  24h dans un jour & 60min dans une heure & 60sec dans une minute & 1000 milliseconds dans une seconde.

→ ***Code fonctionnel :***

```
const dayDiff = (date1, date2) => Math.round((date1 - date2)) / (1000 * 60 * 60 * 24);

console.log(dayDiff(new Date("2020-10-21"), new Date("2021-10-22")));  -366
```

## 04 – Average btwn numbers.

“Find the average between multiple numbers using reduce method.”

→ ***Méthodologie :***

- Toucher l'ensemble de l'argument (sinon arr obligatoire)
- Vérifier que l'argument ne soit pas vide
- Si ok : on filtre les éléments pour garder les valeurs truthy
- On fait la somme avec reduce et on divise par le nombre d'élément (moyenne)
- Si non "0"

→ ***Code fonctionnel :***

```
const average = (...arr) => arr.length > 0 ? (arr.filter(num => num).reduce((sum, num) => sum + num, 0)) / arr.length : 0;

console.log(average(1, 2, 3, 4));  2.5
```

## 05 – Smallest Element.

“Get the Smallest Element of an Array”

→ ***Méthodologie :***

- Trouver le minimum en comparant TOUTE l'array

→ ***Code fonctionnel :***

```
const minArr = (arr) => Math.min(...arr);

console.log(minArr([13, 7, 11, 3, 9, 15, 17]));  3
```

## 06 – Same Value.

“Check if Two Arrays Contain the Same Values”

→ ***Méthodologie :***

- Vérifier que les arr ont la même longueur (sinon d'office false).
- Vérifier qu'après le tri et la conversion en string, ce soit les mêmes.

→ ***Code fonctionnel :***

```
const compareArray = (arr1, arr2) => arr1.length === arr2.length && arr1.sort().toString() === arr2.sort().toString();

const arr1 = [1, 2, 3, 4];
const arr2 = [3, 1, 4, 2];
const arr3 = [1, 2, 3];
console.log(compareArray(arr1, arr2));  true
console.log(compareArray(arr1, arr3));  false
```

## 07 – RGB Random Generator.

"Create a function that generates a random rgb value. The outcome should be the same as we declare it in CSS : rgb(?, ?, ?)"

→ ***Méthodologie :***

- Préparer le format de la réponse `rgb()`.
- Dans cette réponse générer les trois nombre arrondis entre 1 et 255.

→ ***Code fonctionnel :***

```
const rgbGen = (red, blue, green) => `rgb(${Math.round(Math.random()* 255)}, ${Math.round(Math.random()* 255)}, ${Math.round(Math.random()* 255)})`

console.log(rgbGen());  'rgb(208, 76, 211)'
```

## 08 – Letter Count.

"Create a function that takes a string and a letter and counts how many times the letter appears in the string."

→ ***Méthodologie :***

- Split str en utilisant letter comme séparateur
- .length nous donne le nombre de sous-tableau crée, il faut faire -1 pour connaître le nombre de séparateurs (lettre)

→ ***Code fonctionnel :***

```
const letterCount = (str, letter) => str.split(letter).length-1;

console.log(letterCount("hello", "l"));  2
console.log(letterCount("abracadabra", "a"));  5
console.log(letterCount("oups", "z"));  0
```

## 09 – SumArray.

"Create a function that returns the sum of all positive numbers in an array. (negative numbers should be ignored). If only negative numbers are present, it should return 0"

→ ***Méthodologie :***

- Filtrer tous les nombres > 0 puis reduce avec sum comme accumulateur, on ajoute num à la sum et la valeur de départ est 0.

→ ***Code fonctionnel :***

```
const sumArray = (arr) => arr.filter(num => num > 0).reduce((sum, num) => sum + num, 0);

console.log(sumArray([1, 6, 2, -3, 5, -12]));  14
console.log(sumArray([-3, -4, -2]));  0
```

## ***10 – Value Check.***

“Create a function that takes an array of objects and an object with one key/value pair as arguments. The function should return every entries that are the same than the object.”

→ ***Méthodologie :***

- Filtrer tous les éléments du tableau
- Object.entries(obj) convertit l'objet en un tableau de paires de valeur clé
- Every teste si toute nos paires de valeur clé sont conforme à l'array de départ
- On structure l'élément avec [Key, Value]
- La dernière égalité vérifie si la veleur de la key de l'objet item correspond à celle attendue

→ ***Code fonctionnel :***

```
const scanAndFind = (arr, obj) => arr.filter(item => Object.entries(obj).every(([key, value]) => item[key] === value));
```

# Code modulable

## 0.1 ALGOS-série 1 :

### *00 – Condition number.*

```
function conditionNumber (a, b) {
if (typeof a === 'number' && typeof b === 'number'){
  if (Number.isInteger(a) && Number.isInteger(b)) {
    if (a === 10 || b === 10 || a + b === 10){
      return true;
    } else {
      return false;
    }
  } else {
    return false;
  }
} else {
  return false;
}
}
console.log(conditionNumber(6, 4)); // true
```

### *01 – Age in days.*

```
function convertAge (age){
  if (typeof age === 'number' && age >= 0 && Number.isInteger(age)){
    return age * 365;
  } else {
    return "Wrong input!";
  }
}
console.log(convertAge(10)); // 3650
```

## 02 – Special sum.

```
function randomNum (num) {
  if (typeof num !== 'number' || num < 0 || !Number.isInteger(num)){
    return "Error, try with a positiv number!";
  } else {
    let sum = 0;
    for (let i = 1; i <= num; i++){
    sum += i;
    }
    return sum;
  }
}
console.log(randomNum(15)); // 120
console.log(randomNum(-12)); // Error, try with a positiv number!
console.log(randomNum("abs")); // Error, try with a positiv number!
```

## 03 – Min & max – array.

```
function numberArr(arr) {
  if (arr.length <= 0) {
    return "Invalid array"
  } else if (arr.length === 1){
    return [arr[0], arr[0]];
  } else {
    let min = Math.min(...arr);
    let max = Math.max(...arr);
    return [min, max];
  }
}

console.log(numberArr([1, 9, -7])); // [-7, 9]
console.log(numberArr([2])); // [2, 2]
console.log(numberArr([])); // Invalid array
```

## 04 – Word is on the loose.

```
function findWord (letters){
  let wantedWord = '';

  for (let character of letters){
    if (character >= 'a' && character <= 'z') {
      wantedWord += character;
    }
  }
  return wantedWord;
}
console.log(findWord("UcUNFYGaFYFYGtNUH"));
console.log(findWord("bEEFGBuFBRrHgUHlNFYaYr"));
console.log(findWord("YFemHUFBbezFBYzFBYLleGBYEFGBMENTment"));
```

## 05 – Sort by prices.

```
function sortedDrink (drinks){
  drinks.sort((first, second) => {
    if (first.price < second.price){
      return -1;
    } else if (first.price > second.price) {
      return 1;
    } else {
      return 0;
    }
  });
  return drinks;
}
const drinks = [
  {name: "lemonade", price: 10},
  {name: "lime", price: 10},
  {name: "coke", price: 8},
  {name: "vodka", price: 17},
  {name: "water", price: 3},
]
  console.log(sortedDrink(drinks));
```

## 06 – Chickens, cows & pigs.

```
function legsTot (chickens, cows, pigs){
  if (typeof chickens !== 'number' || !Number.isInteger(chickens) || chickens < 0
  || typeof cows !== 'number' || !Number.isInteger(cows) || cows < 0
  || typeof pigs !== 'number' || !Number.isInteger(pigs) || pigs < 0){
  return "Oops mistake, legs are counted with positive integers!";
} else {
  let totalChickenLegs = chickens * 2;
  let totalCowsLegs = cows * 4;
  let totalPigsLegs = pigs * 4;
  let totalLegs = totalChickenLegs + totalCowsLegs + totalPigsLegs;
  return totalLegs;
}
}
console.log(legsTot(1, 2, 3)); // 22
```

## 07 – Gamble.

```
function gamble(proba, prize, pay) {
  if (isNaN(proba) || isNaN(prize) || isNaN(pay)) {
    return "You have to use number to use this equation!"
  } else {
    if (proba * prize > pay){
      return true;
    } else {
      return false;
    }
  }
}
console.log(gamble(0.7, 40, 28)); // false
```

## 08 – Frames per second (FPS).

```
function fps(factor, res){
  if (isNaN(factor) || isNaN(res) || factor < 0 || res < 0){
    return 'Sorry bro, you should use positive numbers to get your FPS'
  } else {
    return factor * res * 60;
  }
}
console.log(fps(1, 1)); // 60
console.log(fps(10, 1)); // 600
console.log(fps(10, 25)); // 15000
```

## 09 – Fuel's quantity.

```
function qtyFuel (km) {
  if (isNaN(km) || km < 0){
    return "To use this function you have to use a positive integer!"
  } else {
    if (km > 0 && km <= 10){
      return 100;
    } else {
      return km * 10;
    }
  }
}
console.log(qtyFuel(15)); // 150
console.log(qtyFuel(23.5)); // 235
console.log(qtyFuel(3)); // 100
```

# 0.2 ALGOS-serie 2 :

## 00 – Count true.

```
function countTrue (arr) {
let count = 0;
  if (arr.length === 0){
    return 0
  } else {
    for (let i = 0; i < arr.length; i++){
    if (arr[i] === true){
      count++;
    }
    }
      return count;
    }
}
console.log(countTrue([true, false, false, true, false])); //2
console.log(countTrue([false, false, false, false])); //0
console.log(countTrue([])); //0
```

## 01– Board game placement.

```
function position (a, b){
  if (isNaN(a) || !Number.isInteger(a) || a < 0
  || isNaN(b) || !Number.isInteger(b) || b < 0){
    return 'Use a positive integer!'
  }
  if (b < a || a === b){
    return false;
  }
  let difference = b - a;
  if (difference <= 6){
    return true;
  }
  return false;
}
console.log(position(3, 7)); // true
console.log(position(2, 2)); // false
console.log(position(5, 3)); // false
```

## 02 – Number length.

```
function number (num){
  if (isNaN(num) || !Number.isInteger(num)){
    return `Use an integer!`
  }
  return num.toString().length;
}
console.log(number(1000)); // 4
console.log(number(12)); // 2
console.log(number(1305981031)); // 10
console.log(number(0)); // 1
```

## 03 – From Object To Array.

```
function objToArray (obj){
  if (obj.length === 0){
    return [];
  }
  return Object.entries(obj);
}
console.log(objToArray({ a: 1, b: 2 })); // [["a", 1], ["b", 2]]
console.log(objToArray({ shrimp: 15, tots: 12 })); // [["shrimp", 15], ["tots",
12]]
console.log(objToArray({})); // []
```

## 04 – Multiply game.

```
function numToArray (num, length){
  let result = [];
  if (isNaN(num) || isNaN(length) || !Number.isInteger(num) || !
Number.isInteger(length)){
    return error;
  }
  for (i = 1; i <= length ; i++){
    result.push(num * i);
  }
  return result
}
console.log(numToArray(7, 5)); // [7, 14, 21, 28, 35]
console.log(numToArray(12, 10)); // [12, 24, 36, 48, 60, 72, 84, 96, 108, 120]
console.log(numToArray(17, 6)); // [17, 34, 51, 68, 85, 102]
```

## 05 – Trailing and leading zeros.

```
function delZero (num) {
  if (isNaN(num) || num <0){
    return `Error, we need a positive number!`
  }
  return parseFloat(num).toString();
  }
console.log(delZero("230.000")); // 230
console.log(delZero("00402")); // 402
console.log(delZero("03.1400")); // 3.14
console.log(delZero("30")); // 30
```

## 06 – Array organization.

```
function sortArray (arr){
  let downloadValue = arr.map(item => {
    let value = Array.isArray(item) ? item[0] : item;
    return [value, item];
  });
  downloadValue.sort((a, b) => a[0] - b[0]);
  return downloadValue.map(item => item[1]);
}
console.log(sortArray([4, 1, 3])); // [1, 3, 4]
console.log(sortArray([[4], [1], [3]])); // [[1], [3], [4]]
console.log(sortArray([4, [1], 3])); // [[1], 3, 4]
console.log(sortArray([[4], 1, [3]])); // [1, [3], [4]]
console.log(sortArray([[3], 4, [2], [5], 1, 6])); // [1, [2], [3], 4, [5], 6]
```

## 07 – Calculator.

```
function calc (num1, operator, num2) {
  if (isNaN(num1) || isNaN(num2) ||
  operator !== "+" && operator !== "-" && operator !== "/" && operator !== "*"){
    return "Can't be calculated. Check values and mathematical operator. Try
again!"
  }
  if (operator === '/' && num2 === 0) {
    return "Can't divide by 0!"
  }
  switch (operator) {
    case '+' :
      return num1 + num2;
    case "-" :
      return num1 - num2;
    case "*" :
      return num1 * num2;
    case "/" :
      return num1 / num2;
  }
}
console.log(calc(2, "+", 2)); // 4
console.log(calc(2, "*", 2)); // 4
console.log(calc(4, "/", 2)); // 2
```

## 08 – Country's area.

```
function areaOfCountry (country, area){ // How to verify if it's a real country
without list?
  if (typeof country !== 'string' || isNaN(area) || area < 0){
    return `Please type a real country with his real area (in km²) to process the
code.`
  }
  let percentage = (area / 148940000)*100
  return `${country} is ${percentage.toFixed(2)}% of the total world's landmass`
}
console.log(areaOfCountry("Russia", 17098242)); // "Russia is 11.48% of the total
world's landmass"
console.log(areaOfCountry("USA", 9372610)); // "USA is 6.29% of the total world's
landmass"
console.log(areaOfCountry("Iran", 1648195)); // "Iran is 1.11% of the total world's
landmass"
```

## 09 – reverseWords.

```
function reverseWords (string) {
  string = string.trim();
  let everyWords = string.split(' ').filter(word => word.length > 0);
  let reverseEveryWords = everyWords.reverse();
  return reverseEveryWords.join(' ');
}
console.log(reverseWords(" the sky is blue")); // "blue is sky the"
console.log(reverseWords("hello   world!  ")); //  "world! hello"
console.log(reverseWords("a good example")); // "example good a"
```

# 0.3 ALGOS-série 3 :

## 00 – Oddish or Evenish.

```
function oddOrEven (num){
  let digits = num.toString().split('');
  let sum = 0;
  for (let i = 0; i<digits.length; i++){
    sum += parseInt(digits[i]);
  }
  if (sum % 2 === 0){
    console.log("Evenish");

  } else {
  console.log("Oddish");
  }
  console.log(digits.join(' + ')+ ' = ' + sum);
  console.log(sum + ` % 2 = ` + sum%2);
}
oddOrEven(43); // Oddish // 4 + 3 = 7 // 7 % 2 = 1
oddOrEven(373); // Oddish // 3 + 7 + 3 = 13 // 13 % 2 = 1
oddOrEven(4433); // Evenish // 4 + 4 + 3 + 3 = 14 // 14 % 2 = 0
```

## 01 – Groceries's price.

```
function getTotalPrice (groceries){
  let totalPrice = 0;
  for (i = 0; i < groceries.length; i++){
    let item = groceries[i];
    totalPrice += parseFloat(item.quantity * item.price);
  }
  return totalPrice.toFixed(2);
}
console.log(getTotalPrice([{ product: "Milk", quantity: 1, price: 1.50 }])); //
1.50
console.log(getTotalPrice([{ product: "Milk", quantity: 3, price: 1.50 }])); //
4.50
console.log(getTotalPrice([
  { product: "Milk", quantity: 1, price: 1.50 },
  { product: "Eggs", quantity: 12, price: 0.10 },
  { product: "Bread", quantity: 2, price: 1.60 },
  { product: "Cheese", quantity: 1, price: 4.50 }
])); // 10.40
console.log(getTotalPrice([
  { product: "Chocolate", quantity: 1, price: 0.10 },
  { product: "Lollipop", quantity: 1, price: 0.20 }
])); // 0.30
```

## 02 – Reverse Odd Length Words.

```
function reverseWord (string) {
  let words = string.split(' ');

  for (let i = 0; i < words.length; i++){
    if (words[i].length % 2 !== 0){
      words[i] = words[i].split('').reverse().join('');
  }
}
return words.join(' ');
}
console.log(reverseWord("Bananas")); //
console.log(reverseWord("One two three four")); // "enO owt eerht four"
console.log(reverseWord("Make sure uoy only esrever sdrow of ddo length")); //
"Make sure you only reverse words of odd length"
```

## 03 – Smooth senteces.

```
function smoothSentences (sentence) {
  let words = sentence.toLowerCase().split(' ')
  for (i = 0; i < words.length -1 ; i++){
    let currentWord = words[i];
    let nextWord = words[i + 1];
    if (currentWord[currentWord.length - 1] !== nextWord[0]){
      return false
    }
  }
  return true;
}
console.log(smoothSentences("Marta appreciated deep perpendicular right
trapezoids")); // true
console.log(smoothSentences("Someone is outside the doorway")); // false
console.log(smoothSentences("She eats super righteously")); // true
```

## 04 – Function 7Seven.

```
function seven (numbers) {
for (let num of numbers){
  let numString = num.toString();
  if (numString.includes("7")) {
    return "Boom!";
  }
}
return "There is no 7 in the array";
}
console.log(seven([1, 2, 3, 4, 5])); // "there is no 7 in the array"
console.log(seven([1, 2, 7, 4, 5])); // "Boom!"
console.log(seven([10, 20, 30, 70])); // "Boom!"
```

## 05 – Celsius & Fahrenheit.

```
function farenCels(temp) {
  let unit = temp.slice(-2).toUpperCase();
  let value = parseFloat(temp.slice(0, -2).trim());

if (isNaN(value) || (unit !== "°C" && unit !== "°F")){
  return "Error!";
} else {
  if (unit === "°C"){
    let fahrenheit = (value * 9/5) + 32;
    return Math.round(fahrenheit) + "°F";
  } else {
    let celsius = (value - 32) * 5/9;
    return Math.round(celsius) + "°C";
  }
}
}
console.log(farenCels("35°C")) // → "95°F"
console.log(farenCels("19°F")) // → "-7°C"
console.log(farenCels("33")) // → "Error"
```

## 06 – Function 7Seven.

```
function findBrokenKeys (expected, typed){
  let brokenKeys = [];
  for (let i = 0; i < expected.length; i++){
    if (expected[i] !== typed[i] && !brokenKeys.includes(expected[i])){
      brokenKeys.push(expected[i]);
    }
  }
  return brokenKeys;
}
console.log(findBrokenKeys("happy birthday", "hawwy birthday")) // ["p"]
console.log(findBrokenKeys("starry night", "starrq light")) // ["y", "n"]
console.log(findBrokenKeys("beethoven", "affthoif5")) // ["b", "e", "v", "n"]
```

# 0.4 ALGOS-advanced :

## 00 – FizzBuzz function.

```
function fzbz (num){
if (isNaN(num) || num <= 0 || !Number.isInteger(num)){
  return "Error, use a positive integer!";
} else {
  for (let i = 1; i <= num; i++){
    if (i % 3 === 0 && i % 5 === 0) {
      console.log("FizzBuzz");
    } else if (i % 3 === 0) {
      console.log("Fizz");
    } else if (i % 5 === 0){
      console.log("Buzz");
    } else {
      console.log(i);
    }
  };
}
}
fzbz(15);
```

## 01 – RansomNote.

```
function ransomNote (noteText, magazineText){
  let noteWords = noteText.split(' ');
  let magazineWords = magazineText.split(' ');
  let noteCount = {};
  let magazineCount = {};

  for (let word of noteWords) {
    noteCount[word] = (noteCount[word] || 0) + 1;
  }
  for (let word of magazineWords) {
    magazineCount[word] = (magazineCount[word] || 0) + 1;
  }
  for (let word in noteCount) {
    if (noteCount[word] > (magazineCount[word] || 0)) {
      return false;
    }
  }
  return true;
}
let noteText = "this is a note to you from a secret admirer";
let magazineText = "puerto rico is a great place you must hike far from town to
find a secretwaterfall that i am an admirer of but note that it is not as hard as
it seems this is my advice to you";
console.log(ransomNote(noteText, magazineText));
```

## 02 – Palindrome.

```
function isPalindrome (text){
let cleanText = text.replace(/[^a-zA-Z]/g, "").toLowerCase();
let reverseText = cleanText.split('').reverse().join('');
return cleantText === reverseText;
}
console.log(isPalindrome("hello"));
```

## 03 – Caesar Cipher.

```
function caesarCipher (str, num){
  str = str.toLowerCase();
  let charArray = str.split('');
  let result = '';

  const alphabet = 'abcdefghijklmnopqrstuvwxyz';

  for (let i = 0; i < charArray.length; i++){
    let char = charArray[i];
    if (alphabet.includes(char)){
      let index = alphabet.indexOf(char);
      let newIndex = (index + num) % alphabet.length;
      if (newIndex < 0){
        newIndex += alphabet.length;
      }
      result += alphabet[newIndex];
    } else {
      result += char;
    }
  }
  return result;
}
console.log(caesarCipher("hello world", 3));  // khoor zruog
console.log(caesarCipher("khoor zruog", -3)); // hello world
console.log(caesarCipher("abc xyz", 5));     // fgh cde
console.log(caesarCipher("z", 1));            // a
```

## 04 – ReverseCharacter.

```
function reverseWords (string) {
  string = string.toLowerCase();
  let words = string.split(' ');
  let reverseWords = [];

  for (i=0 ; i< words.length; i++){
  let word = words[i];
  let reverseWord = word.split('').reverse().join('');
  reverseWords.push(reverseWord);
  }

  let result = reverseWords.join(' ');
  return result;
}
console.log(reverseWords('This is a string of words')); // Siht si a gnirts fo
sdrow

// MAP METHOD :

function reverseWords(string) {
string = string.toLowerCase();
  // Séparer la chaîne en mots
  let words = string.split(' ');
  // Inverser chaque mot en utilisant map
  let reversedWords = words.map(word =>
    word.split('').reverse().join('')
  );
  // Assembler les mots inversés en une seule chaîne
  let result = reversedWords.join(' ');
  // Retourner le résultat
  return result;
}

// Exemples d'utilisation
console.log(reverseWords("Hello World"));  // "olleH dlroW"
console.log(reverseWords("JavaScript is fun"));  // "tpircSavaJ si nuf"
```

## 05 – ReverseArrayWithoutReverse.

```
function reverseArray (array){
  let reversed = [];

  for (let i=array.length - 1; i>=0 ; i--){
    reversed.push(array[i]);
  }
  return reversed;
}
console.log(reverseArray([1, 2, 3, 4, 5])) // [5, 4, 3, 2, 1]
console.log(reverseArray(['apple', 'banana', 'cherry', 'date'])) // ['date',
'cherry', 'banana', 'apple'];
console.log(reverseArray([{ name: 'Alice', age: 30 },{ name: 'Bob', age: 25 },{
name: 'Charlie', age: 35 }]));
// [{name: 'Charlie', age: 35},{name: 'Bob', age: 25}, {name: 'Alice', age: 30}]
```

## 06 – FindSumPair.

```
function sumArray (numArray, sum){
  let result = [];
  for (let i = 0; i < numArray.length; i++){
    for (let j = i + 1; j <numArray.length; j++){
      if (numArray[i] + numArray[j] === sum){
        result.push([numArray[i],numArray[j]]);
      }
    }
  }
  return result;
}
console.log(sumArray([1, 2, 8, 13, 12, 9], 14));
```

## 07 – Fibonacci.

```
function fibonacci (num){
  let fibArray = [];
  for (i = 0; i < num; i++){
    if (i === 0 || i === 1){
      fibArray.push(1);
    } else {
      let nextNumber = fibArray[i-1] + fibArray[i-2];
      fibArray.push(nextNumber);
    }
  }
  return fibArray;
}
console.log(fibonacci(4)); // [1, 1, 2, 3]
console.log(fibonacci(9)); // [1, 1, 2, 3, 5, 8, 13, 21, 34]
console.log(fibonacci(6)); // [1, 1, 2, 3, 5, 8]
```

# 0.5 ALGOS-optional-oneLiners :

*Utilisation de Quokka.js → disparition des réponses aux log*

## 01 – Remove Duplicates.

```
const removeDuplicates = (arr) =>[...new Set(arr)]

console.log(removeDuplicates([4, 9, 5, 1, 3, 2, 4, 1, 8]));
console.log(removeDuplicates(["hello", "world", "goodbye", "world"]));
console.log(removeDuplicates([true, true, false, true, true, false]));
```

## 02 – Capitalize.

```
const capitalize = (string) => string[0].toUpperCase() +
string.slice(1).toLowerCase();

console.log(capitalize("belgium"));
console.log(capitalize("brazil"));
console.log(capitalize("congo"));
```

## 03 – The Days Between.

```
const dayDiff = (date1, date2) => Math.round((date1 - date2)) / (1000 * 60 * 60 *
24);

console.log(dayDiff(new Date("2020-10-21"), new Date("2021-10-22")));
```

## 04 – Average btwn numbers.

```
const average = (...arr) => arr.length > 0 ? (arr.filter(num => num).reduce((sum,
num) => sum + num, 0)) / arr.length : 0;

console.log(average(1, 2, 3, 4));
```

## 05 – Smallest Element.

```
const minArr = (arr) => Math.min(...arr);

console.log(minArr([13, 7, 11, 3, 9, 15, 17]));
```

## 06 – Same Value.

```
const compareArray = (arr1, arr2) => arr1.length === arr2.length &&
arr1.sort().toString() === arr2.sort().toString();

const arr1 = [1, 2, 3, 4];
const arr2 = [3, 1, 4, 2];
const arr3 = [1, 2, 3];
console.log(compareArray(arr1, arr2));
console.log(compareArray(arr1, arr3));
```

## 07 – RGB Random Generator.

```
const rgbGen = (red, blue, green) => `rgb(${Math.round(Math.random()* 255)}, $
{Math.round(Math.random()* 255)}, ${Math.round(Math.random()* 255)})`

console.log(rgbGen());
```

## 08 – Letter Count.

```
const letterCount = (str, letter) => str.split(letter).length-1;

console.log(letterCount("hello", "l"));
console.log(letterCount("abracadabra", "a"));
console.log(letterCount("oups", "z"));
```

## 09 – SumArray.

```
const sumArray = (arr) => arr.filter(num => num > 0).reduce((sum, num) => sum +
num, 0);

console.log(sumArray([1, 6, 2, -3, 5, -12]));
console.log(sumArray([-3, -4, -2]));
```

## 10 – Value Check.

```
const scanAndFind = (arr, obj) => arr.filter(item =>
Object.entries(obj).every(([key, value]) => item[key] === value));
```