JAVASCRIPT

Revert
Créer une fonction `revert(str)` qui inverse une chaîne de caractère et la retourne
Exemple : `revert("Hello I'm Robert !") // treboR m'I olleH`

UcFirst
Ecrire une fonction `ucFirst (word)` qui transforme la première lettre d'une chaine de caractère en majuscule
exemple : `ucFirst("hello") // Hello`

Capitalize
Ecrire une fonction `capitalize(str)` qui transforme chaque première lettre de chaque mot en majuscule
Exemple : `capitalize("hello i'm robert") // Hello I'm Robert`

PascalCase
Ecrire une fonction `pascalCase(str)` qui transforme chaque première lettre de chaque mot en majuscule et supprime les espaces
Exemple : `pascalCase("sentence in pascalCase") // SentenceInPascalCase`

Palindrome
Créer une fonction `palindrome(str)` qui vérifie si un mot est un palindrome et retourne un booléen
Exemple : `palindrome('kayak') //true`

FindLongestWord
Écrire une fonction `findLongestWord(str)` qui permet de trouver le mot le long plus d'une chaîne de caractère
Exemple : `findLongestWord("Le chemin le plus cours n'est pas toujours le meilleur"); // toujours`

Mapi
Créer une class Mapi
⚠️ Interdiction d'utiliser l'objet Map
Mapi est une classe qui contient une liste d'objets clés valeur, les clés sont uniques .
Elle possède une propriété size, qui permet de connaître la taille de la liste d'élément

Créer une fonction :

- **set(key,elt)**, qui prend une clé et un élément en paramètre puis ajoute l'élément dans Mapi associé à la clé.
- **delete(key)**, qui prend une clé en paramètre et supprime l'élément associé à cette clé
- **get(key)**, qui prend une clé en paramètre et renvoie l'élément associé à cette clé

- **has(key)**, qui prend une clé en paramètre et renvoie un booléen si la clé est présente dans la liste
- **keys()**, qui renvoie toutes les clés de Mapi
- **values()**, qui renvoie toutes les valeurs de Mapi

Si on ajoute un objet dans une instance de Mapi avec une clé déjà présente dans la liste, la valeur de la clé deviendra celle du nouvel objet

Exemple :

```
const map = new Mapi([['hero',true],['moldu',false]]);
console.log(map.size); // 2
map.set(1,'oui');
map.set(2,'non');
console.log(map.values()); // ["oui", "non", true, false]
map.set(2,'oui');
console.log(map.values()); // ["oui", "oui", true, false]
map.delete(1);
console.log(map.keys()); // ["2", "hero", "moldu"]
map.set('az-56-0abdo35',3);
console.log(map.get('az-56-0abdo35')); // 3
console.log(map.has(1)); // false
console.log(map.has(2)); // true
console.log(map.size); // 4
```

Prop access

Écrire une fonction `prop_access(object, path)` qui prend un paramètre un chemin et un objet et retourne la valeur associée.
Si l'attribut n'existe pas alors retourné le chemin + "not exist" (ex: "post.comments.type not exist"
Si le chemin est vide Renvoyez l'objet complet

Exemple :

```
console.log(prop_access({
    "animal":{
        "type":{
            "name": "dog"
        }
    }
},
"animal.type.name")); // dog
console.log(prop_access({
```

```
    "animal":
    {
        "type":{
            "name": "dog"
        }
    }
}, "animal.type"
)); // { name: "dog" }

console.log(prop_access({
    "animals":[
    {
        "type":{
            "name": "dog"
        }
    },
    {
        "type":{
            "name": "cat"
        }
    }
   ]
}, "animals.1.type"
)); // { name: "cat" }
```

Type check

Ecrire une fonction `type_check_v1(value,type)` qui vérifie que l'argument 1 est égale au type de l'argument 2

Exemple : `type_check_v1(1, "number") // true`

Type check v2

Gérer un objet de conf qui vérifie le type, la valeur et enum (un tableau de valeur possible)

Exemple : `type_check_v2("toto", { type: "string", enum: ["toto1", "toto2"] }) // false;`

PARTIE II

Hash tag (⚠️ Écrire en une ligne)

Écrivez une fonction `getHashTags(str)` qui récupère les trois mots les plus longs d'une chaîne de caractère et les transforme en hashtags. Si plusieurs mots sont de la même longueur, récupérez le mot qui apparaît en premier.

```
console.log(getHashTags("How the Avocado Became the Fruit of the
Global Trade")); // ["#avocado", "#became", "#global"]
console.log(getHashTags("Why You Will Probably Pay More for Your
Christmas Tree This Year")); // ["#christmas", "#probably", "#will"]
console.log(getHashTags("Hey Parents, Surprise, Fruit Juice Is Not
Fruit")); // ["#surprise", "#parents", "#fruit"]
console.log(getHashTags("Visualizing Science")); //
["#visualizing","#science"]
```

RemoveDuplicate (⚠️ Écrire en une ligne)
Créer une fonction qui supprime les valeurs dupliquées dans un tableau (Number, String)
Exemple :

```
const tab = [0, 2, 4, 6, 8, 8];
removeDuplicate(tab) // [0,2,4,6,8]
```

Intersection (⚠️ Écrire en une ligne)
Créer une `intersection(arr,arr2)` fonction qui calcule l'intersection entre 2 tableaux (Number, String)

```
const first = [0, 2, 4, 6, 8, 8];
const second = [1, 2, 3, 4, 5, 6];
intersection(first,second) // [2,4,6]
```

ArrayDiff (⚠️ Écrire en une ligne)
Créer une fonction `arrayDiff(first,second)` qui calcule la différence entre 2 tableaux (Number, String)
Exemple :

```
const first = [0, 2, 4, 6, 8, 8];
const second = [1, 2, 3, 4, 5, 6];
arrayDiff(first,second) // [2,4,6]
```

Combination (⚠️ Écrire en une ligne)
Créez une fonction `combinations` qui prend un nombre variable d'arguments, chaque argument représentant le nombre d'articles dans un groupe, et qui renvoie le nombre de permutations (combinaisons) d'articles que vous pourriez obtenir en prenant un article de chaque groupe.
Exemple :

```
combinations(2, 3) // 6
combinations(3, 7, 4) // 84
combinations(2, 3, 4, 5) // 120
```

Fiscal code

En Italie, chaque personne possède un code d'identification unique délivré par l'administration fiscale nationale après l'enregistrement de la naissance : le code fiscal (Codice Fiscale). Consultez la page https://en.wikipedia.org/wiki/Italian_fiscal_code pour plus d'informations

Vous aurez à disposition un objet contenant les données personnelles d'une personne (nom, prénom, sexe et date de naissance), Créer une fonction `fiscalCode` qui renvoie les 11 caractères du code sous forme de chaîne de caractères en suivant ces étapes :

- Générez 3 lettres majuscules à partir du nom de famille, s'il y a:
    - Au moins 3 consonnes,prendre les 3 premières consonnes. (Newman -> NWM).
    - Moins de 3 consonnes, les voyelles remplaceront les caractères manquants dans le même ordre qu'ils apparaissent (Fox -> FXO | Hope -> HPO).
    - Moins de trois lettres, "X" prendra le troisième emplacement après la consonne et la voyelle (Yu -> YUX).
- Générez 3 lettres majuscules à partir du nom, s'il y a :
    - Exactement 3 consonnes ,les consonnes sont utilisées dans l'ordre où elles apparaissent (Matt -> MTT).
    - Plus de 3 consonnes , la première, troisième et quatrième consonnes sont utilisées (Samantha -> SNT | Thomas -> TMS).
    - Moins de 3 consonnes, les voyelles remplaceront les caractères manquants dans le même ordre qu'ils apparaissent (Bob -> BBO | Paula -> PLA).
    - Moins de trois lettres, "X" prendra le troisième emplacement après la consonne et la voyelle (Al -> LAX).
- Générez 2 chiffres, 1 lettre et 2 nombres à partir de la date de naissance et du sexe :

- Prenez les deux derniers chiffres de l'année de naissance (1985 -> 85).

- Générez une lettre correspondant au mois de naissance (janvier -> A | décembre -> T) en utilisant la table de conversion incluse dans le code.
- Pour les hommes, prenez le jour de naissance en ajoutant un zéro au début s'il est inférieur à 10 (tout 9e jour -> 09 | tout 20e jour -> 20).
- Pour les femmes, prenez le jour de naissance et ajoutez 40 (tout 9e jour -> 49 | tout 20e jour -> 60).

Notes
Les lettres de code doivent être en majuscules.
La date de naissance est indiquée dans le format J/M/AAAA.
Y n'est pas une voyelle.
La table de conversion des mois est la suivante :
Janvier => A
Février => B
Mars => C
Avril => D
Mai => E
Juin => H
Juillet => L
Août => M
Septembre => P
Octobre => R
Novembre => S
Décembre => T

Exemple :

```
fiscalCode({
  name: "Matt",
  surname: "Edabit",
  gender: "M",
  dob: "1/1/1900"
}) //"DBTMTT00A01"

fiscalCode({
  name: "Helen",
  surname: "Yu",
  gender: "F",
  dob: "1/12/1950"
}) //"YUXHLN50T41"

fiscalCode({
  name: "Mickey",
  surname: "Mouse",
  gender: "M",
  dob: "16/1/1928"
}) // "MSOMKY28A16"
```

Least Common multiple

A partir d'un tableau d'entiers, créez une fonction **lcm** qui trouvera le plus petit entier positif qui est divisible par tous les nombres du tableau. En d'autres termes, trouvez le plus petit multiple commun (LCM).

Exemple :

```
lcm([1, 2, 3, 4, 5, 6, 7, 8, 9]) // 2520
lcm([5]) // 5
lcm([5, 7, 11]) // 385
lcm([5, 7, 11, 35, 55, 77]) // 385
```

Merge
Créer une méthode merge qui prend en paramètres des objets et les fusionnent.

Exemple :

```
const object = {
  a: [{ x: 2 }, { y: 4 }],
  b: 1
};
const other = {
  a: { z: 3 },
  b: [2, 3],
  c: 'foo'
};
console.log(merge(object, other)); //
{"a":[{"x":2},{"y":4},{"z":3}],"b":[1,2,3],"c":"foo"}
```

Comments filter
Créer une fonction qui à partir de l'url donné en paramètre récupère tous les commentaires commentaires des adresses mails finissant par **.us** ou **uk.**, ensuite trie selon l'id de manière décroissante et supprime tous les commentaires qui ont un **name** supérieur à 31 caractères

Url à utiliser : https://jsonplaceholder.typicode.com/comments

Exemple :

```
;(async function main () {
  try {
    const commentsFiltered = await
```

```
commentsFilter('https://jsonplaceholder.typicode.com/comments');
    console.log(commentsFiltered); // (16)[...]
  } catch(err){
    // handle error
    console.err('error')
  }
})()
```