**Mise en application**

# Boucles

1. Créer une boucle qui affiche les nombres de 1 à 20 var i=1

while (i<=20){

console.log(i) ;

i=i+1 ;

}

OU

let i=1

for (i=1 ; i<=20 ;++i){

console.log(i) ;

} mieux car je peux mettre directement sur une seule ligne

1. Créer une boucle qui affiche les nombres de 20 à 0 par étape de 2

for (let cuicui=20 ; cuicui>=0 ;cuicui=cuicui-2){

console.log(cuicui) ;

}

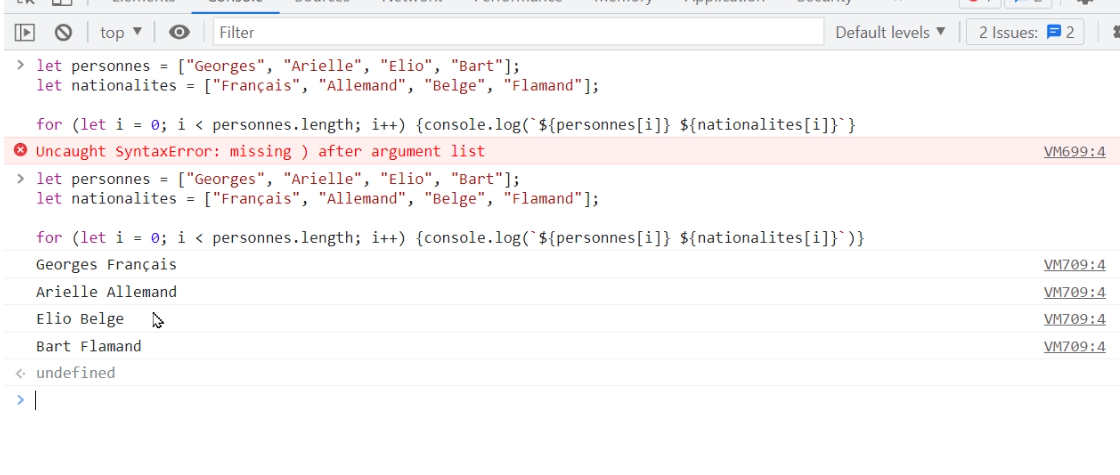
1. Etant donné deux arrays **personnes** et **nationalites,** utilisez une bonne boucle pour afficher le nom de chaque personne, suivi de sa nationalité.

personnes = ["Georges", "Arielle", "Elio", "Bart"]; nationalites = ["Français", "Allemand", "Belge", "Flamand"];

for (let i=0 ; i<personnes.length; i++){

console.log(personnes[i] + " est " + nationalites[i]);

}



1. Grâce aux boucles affichez tous les positions d'un plateau d'échecs dans l'ordre, en partant de A1. Pensez à vous inspirer de l'exercice précédent.

On n’a pas besoin de mettre les chiffres de 1 à 8 dans un array, pas besoin de les stocker quelques part.

var piou=["A", "B","C","D","E","F","G","H"] 🡪je dois l’écrire manuellement

for (compteur=0 ; compteur<piou.length ; compteur++){

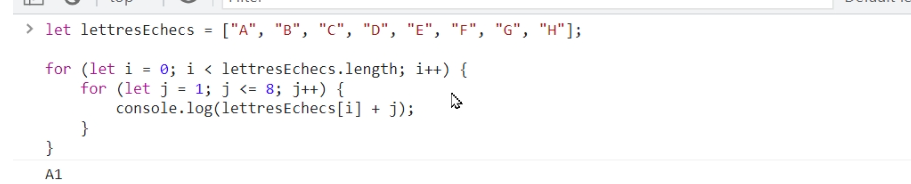
for (var cuicui=1 ; cuicui<=piou.length ; cuicui++){

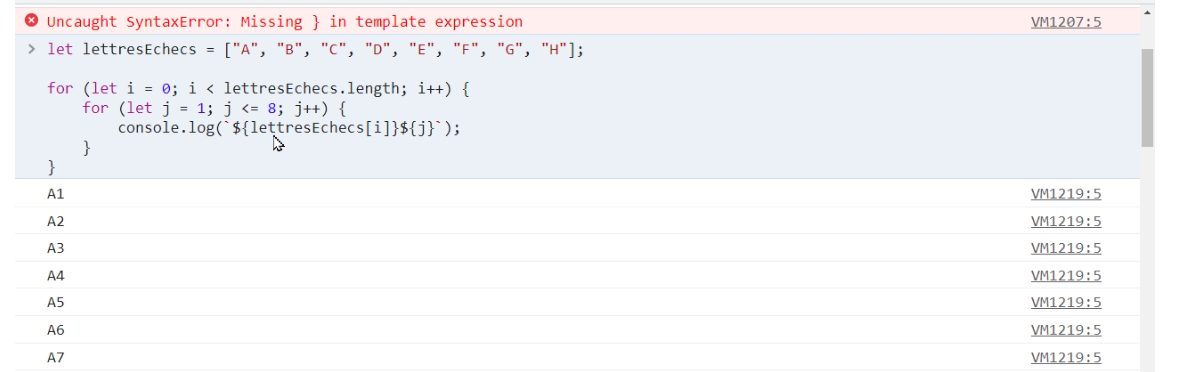
console.log(piou[compteur] + (compteur+1)) ;

}

}

Deux manières de le faire : le premier sans backtick, le second avec





1. Créer une fonction **additionneTout** qui prend une suite de nombres et les additionne entre eux.

function additionneTout(...rest){ d’abord créer ma fonction puis mettre les parenthèses d’appel et ensuite les arguments …rest

ensuite pour tester déjà mettre les paramètres tout en dessous (ex additionneTout(4,12,7,21))

ensuite on devra créer 2 variables pour permettre de stocker les informations

var calcul = 0; 🡪pour enregistrer mon addition et continuer d’additionner

for (var i=0; i<rest.length; i++) {

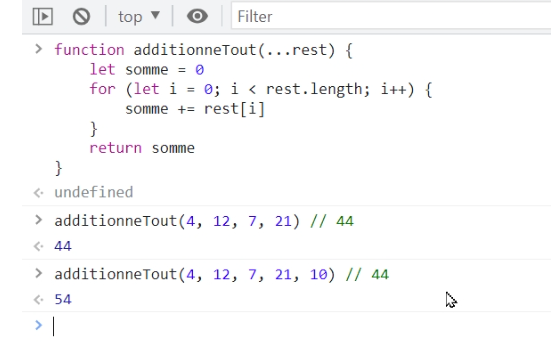
calcul += rest[i] ;

}

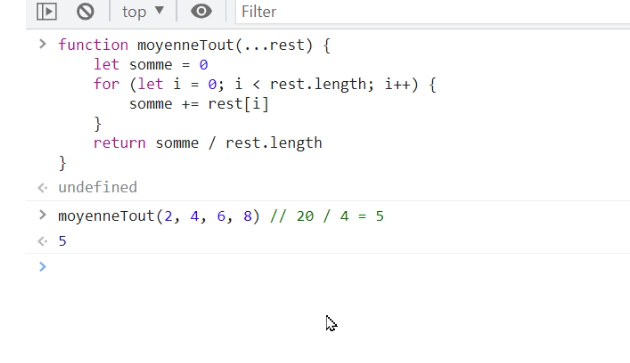
return calcul

}

additionneTout(2, 4, 6, 8)



Si je veux ensuite calculer la moyenne (question bonus) :



Je divise par 4 car 4 éléments dans mon tableau

1. Créer votre propre fonction join qui portera le nom **joindreValeurs**. Elle doit se comporter comme la fonction du même nom liée aux arrays sauf qu'elle prendra une suite de valeurs ou de strings à la place d'un array.

On regarde d’abord ce qu’est la fonction join. Elle sert à concaténer.

<https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Global_Objects/Array/join>

Comme la fonction join met par défaut une virgule entre chaque valeur, je vais devoir faire de même – je vais devoir prévoir un « séparateur » dans les paramètres

Un paramètre par défaut va prendre sa valeur soit quand on lui dit spécifiquement undifined ou soit quand je lui donne sa valeur.

Ici je vais mettre la virgule en premier argument et pas définir par défaut une valeur à mon séparateur pour qu’il prenne comme valeur par défaut mon premier argument.

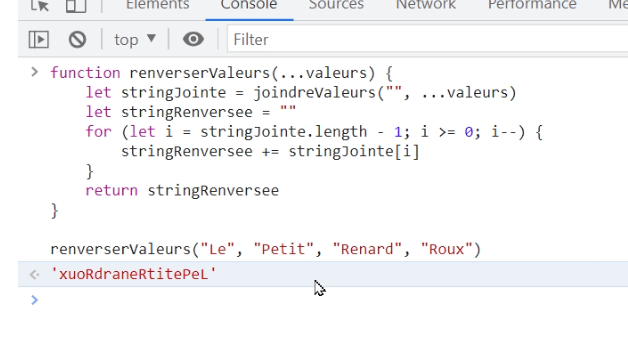


1. Créer votre propre fonction reverse qui portera le nom **renverserValeurs**. Elle doit se comporter comme la fonction du même nom liée aux arrays sauf qu'elle prendra une suite de valeurs ou de strings à la place d'un array et concatènera d'abord toutes ces valeurs ensemble.

On va créer notre fonction. On va lui mettre un paramètre de rest.

Notre boucle partira de la fin pour aller vers le début et devra concaténer (ce qu’on codera en premier). On peut utiliser join, mais autant utiliser la fonction qu’on a créé à l’exercice précédent.

Attention que si on fait cela, on ne pourra pas utiliser un array – contrairement à la fonction join, notre fonction joindreValeurs est un string. Là, les « … » vont être très utiles. S’ils ne sont pas utilisés dans un array, alors ils sont alors « opérateur de déconstruction ».



On a « simplifié » en réutilisant la fonction précédente. Dans le futur on utilisera plutôt join directement

1. En vous inspirant de la fonction suivante, créer une fonction récursive qui affiche les nombres en partant d'une valeur de départ jusqu'à une valeur maximum toutes deux données lors de l'appel.

On doit faire une fonction récursive qui va faire le contraire de celle que j’ai actuellement. Donc elle va compter (au lieu de décompter) d’un nombre de départ jusqu’à un nombre final – genre de 1 à 10.

function decompteurRecursif(max = 0) { if ( max >= 0 ) { console.log(max);

decompteurRecursif(max - 1);

}

}

Cela va donner ci-dessous. ATTENTION : quand on appelle le curseurRecursif, je dois bien préciser min+1 **ET** mettre « , max » sinon quand la fonction va se rappeler il ne va pas tenir compte du fait que j’ai changé la valeur de mon maximum.

