Module : JavaScript

Exercice n°8 - Loops

# Durée de l’exercice : 30minLoops

Les boucles sont pratiques, si vous voulez exécuter le même code encore et encore, chaque fois avec une valeur différente.

**Objectif de l'exercice** : connaître et comprendre les boucles en JavaScript, comprendre le contrôle dans les boucles (comment le code fonctionne dans les boucles).

**Énoncé** : Suivez les étapes ci-dessous pour réaliser votre exercice.

# créer un dossier js\_loops à l'intérieur de ce dossier créer les fichiers ci-dessous taper le code prédire le résultat et le comparer avec le résultat de l'impression après avoir exécuté le code.

Résultat et comparez avec le résultat imprimé après avoir exécuté le code # code-1.js. Faites une prédiction de ce qu'il va imprimer. Exécutez-le.

console.log("hello Friends") ;

for (let i = 0 ; i < 5 ; i++) {

console.log("code-1") ;

}

console.log("goodbye code-1") ;

# code-2.js

console.log("au revoir code-2") ;

for (let i = 3 ; i <= 7 ; i++) {

console.log("programme forme France") ;

console.log(i) ;

}

console.log("bye code-2") ;

#code-3.js

let foo = fonction () {

for (let num = 10 ; num > 0 ; num -= 2) {

console.log(num) ;

}

} ;

console.log("début") ;

foo() ;

console.log("end") ;

foo() ;

# code-4.js

let word = "langue" ;

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

console.log(i) ;

console.log(mot[i]) ;

}

#code-5.js

let sum = 0 ;

for (let i = 1 ; i < 5 ; i++) {

sum += i ;

console.log(sum) ;

}

console.log("grand total : " + somme) ;

// Il est maintenant temps de montrer votre habileté en écrivant un code.

#loop-one-to-six.js

// Écrivez une fonction `oneToSix` qui imprime tous les nombres entiers de un à six, inclus.

// La fonction ne prend aucun argument et ne doit pas retourner de valeur. Elle doit juste s'afficher dans le terminal.

oneToSix() ;

// imprime

// 1

// 2

// 3

// 4

#loop-min-to-max.js

// Écrivez une fonction `minToMax(min, max)` qui accepte deux nombres comme arguments.

// La fonction doit imprimer tous les nombres de min à max inclus.

// La fonction n'a pas besoin de retourner une valeur. Elle doit simplement imprimer sur le terminal.

minToMax(4, 12) ;

// imprime

// 4

// 5

// 6

// 7

// 8

// 9

// 10

// 11

// 12

minToMax(11, 14) ;

// imprime

// 11

// 12

// 13

// 14

#loop-string-iterate.js

// Écrivez une fonction `stringIterate` qui accepte une chaîne de caractères en tant qu’argument.

// La fonction doit imprimer chaque caractère de la chaîne, un par un.

// La fonction n'a pas besoin de retourner une valeur. Elle doit simplement s'imprimer sur le terminal.

stringIterate("cement") ;

// imprime

// c

// e

// m

// e

// n

// t

stringIterate("rat") ;

// imprime

// r

// a

// t

#Créer un fichier loop-five-multiples-of.js et résoudre la fonction ci-dessous :

// Ecrivez une fonction nommée `fiveMultiplesOf` qui accepte un nombre comme argument.

// La fonction doit imprimer les cinq premiers multiples du nombre donné.

// La fonction n'a pas besoin de retourner une valeur. Elle doit simplement s'afficher dans le terminal.

cinqMultiplesOf(7) ;

// imprime

// 7

// 14

// 21

// 28

// 35

cinqMultiplesDe(3) ;

// imprime

// 3

// 6

// 9

// 12

// 15

# Créer un fichier loop-sum-up-to.js et résoudre la fonction ci-dessous.

// Écrivez une fonction nommée `sumUpTo(max)` qui accepte un nombre max comme argument.

// La fonction doit renvoyer la somme totale de tous les nombres entiers compris entre 1 et le max, inclus.

// Par exemple, sumUpTo(4) doit renvoyer 10 car 1 + 2 + 3 + 4 = 10.

console.log(sumUpTo(4)) ; // 10

console.log(sumUpTo(5)) ; // 15

console.log(sumUpTo(2)) ; // 3

# Créer un fichier loop-div-by-either.js et résoudre la fonction ci-dessous.

// Écrivez une fonction nommée `divByEither(num1, num2, max)`. Elle accepte trois nombres comme arguments.

// La fonction doit afficher tous les nombres positifs inférieurs à max qui sont divisibles par num1 ou num2.

// La fonction n'a pas besoin de renvoyer de valeur. Elle doit simplement s'afficher dans le terminal.

divByEither(4, 3, 16) ;

// imprime

// 3

// 4

// 6

// 8

// 9

// 12

// 15

divByEither(7, 5, 20) ;

// imprime

// 5

// 7

// 10

// 14

// 15

# Créez un fichier loop-divisible-range.js et résolvez la fonction ci-dessous.

// Écrivez une fonction `divisibleRange(min, max, num)` qui accepte trois nombres comme arguments.

// La fonction doit imprimer tous les nombres compris entre `min` et `max` (exclusif) qui sont également divisibles par num.

divisibleRange(17, 40, 9) ;

// imprime

// 18

// 27

// 36

divisibleRange(10, 24, 4) ;

// imprime

// 12

// 16

// 20

# Créer un fichier loop-divisible-range.js et résoudre la fonction ci-dessous.

// Écrivez une fonction `reverseIterate` qui accepte une chaîne de caractères comme argument.

// La fonction doit imprimer les caractères de la chaîne un par un, dans l'ordre inverse.

// La fonction n'a pas besoin de retourner une valeur. Elle doit simplement s'afficher dans le terminal.

reverseIterate("carrot") ;

// imprime

// t

// o

// r

// r

// a

// c

reverseIterate("box") ;

// imprime

// x

// o

// b

# Créez un fichier loop-fizz-buzz.js et résolvez la fonction ci-dessous.

// Ecrivez une fonction `fizzBuzz` qui accepte un nombre max comme argument.

// La fonction doit imprimer tous les nombres inférieurs ou égaux à max qui sont divisibles par 3 ou 5, mais pas par 3 et 5.

// La fonction n'a pas besoin de retourner une valeur. Elle doit simplement s'afficher dans le terminal.

fizzBuzz(18) ;

// imprime

// 3

// 5

// 6

// 9

// 10

// 12

// 18

fizzBuzz(33) ;

// imprime

// 3

// 5

// 6

// 9

// 10

// 12

// 18

// 20

// 21

// 24

// 25

// 27

// 33

# Créez un fichier loop-remove-capitals.js et résolvez la fonction ci-dessous.

// Écrivez une fonction `removeCapitals` qui accepte une chaîne de caractères comme argument.

// La fonction doit retourner une nouvelle version de la chaîne avec toutes les majuscules enlevées.

console.log(removeCapitals("fOrEver")) ; // 'frver'.

console.log(removeCapitals("raiNCoat")) ; // "raioat".

console.log(removeCapitals("clr Door")) ; // "clr oor