

Les fonctions

Table des matières

I. Contexte	3
II. Une fonction en JavaScript	3
III. Exercice : Appliquez la notion	5
IV. Les paramètres des fonctions	5
V. Exercice : Appliquez la notion	6
VI. Les fonctions JavaScript prédéfinies	7
VII. Exercice : Appliquez la notion	8
VIII. La portée des variables	9
IX. Exercice : Appliquez la notion	11
X. Auto-évaluation	11
A. Exercice final	11
B. Exercice : Défi	13
Solutions des exercices	15

I. Contexte

Durée : 45 min

Environnement de travail : Repl.it

Pré-requis : Connaître les bases de JavaScript

Contexte

Lors du développement d'une application, il arrive que l'on souhaite réutiliser, à plusieurs endroits de notre code, un bloc de code que l'on a déjà écrit.

Plutôt que de le copier/coller, nous allons créer une fonction que nous pourrons appeler à différents endroits de notre application et qui exécutera ce bloc d'instructions.

Il y a plusieurs avantages à cela :

- En ne faisant pas un copier/coller à plusieurs endroits, on évite d'alourdir inutilement le code.
- Cela facilite également la maintenance du code, car s'il y a une modification à apporter à ce bloc d'instructions, on ne le fait qu'à un seul endroit : dans la fonction.

JavaScript propose des fonctions prédéfinies afin de faciliter notre développement, nous en étudierons quelques-unes dans ce cours.

II. Une fonction en JavaScript

Objectifs

- Apprendre ce qu'est une fonction JavaScript
- Savoir utiliser le retour d'une fonction (`return`)

Mise en situation

Nous allons voir comment est déclarée une fonction JavaScript et ce qu'il est possible de réaliser, avec quelques exemples simples.

Méthode

Déclarer une fonction JavaScript simple

En JavaScript, une fonction est déclarée en utilisant le mot `function` suivi d'un espace, puis du nom de la fonction (ex : `myFunctionName`) auquel sont collées des parenthèses suivies d'accolades `{ }`. Les actions que feront la fonction sont notées entre ces accolades.

Exemple

```
1 // Declare the function
2 function myFunctionName() {
3   // Here the code of my function
4 }
5
```

Méthode Appeler et exécuter une fonction JavaScript simple

Pour exécuter une fonction, il suffit de l'appeler par son nom.

```
1 // Déclarer la fonction
2 function direBonjour()
3 {
4   console.log("Bonjour !")
5 }
6
7 // Appel de la fonction
8 direBonjour()
```

Méthode Retour d'une fonction avec return

Une fonction peut exécuter un bloc d'instructions (comme dans l'exemple ci-dessus), auquel il est possible de faire retourner une valeur grâce à l'instruction `return`.

Cela permet, par exemple, de récupérer une valeur calculée (nombre, texte, liste de valeurs, booléen `true/false`, etc.) pour continuer à l'utiliser dans le reste de notre code.

Une fonction s'arrête immédiatement à l'instant où l'instruction `return` est traitée.

La valeur retournée peut être directement traitée ou stockée dans une variable.

Exemple

```
1 // Square of a number
2 function squareNumber() {
3   return 10
4 }
5
6 let number = squareNumber()
7
8 // Display in the console : 10
9 console.log(squareNumber())
10 console.log(number);
```

Syntaxe À retenir

- Pour déclarer une fonction, on utilise le mot `function` suivi d'un espace, puis du nom de la fonction auquel sont accolées des parenthèses, et on écrit le code entre accolades.
- Une fonction peut retourner une valeur grâce à l'instruction `return`.
- Pour appeler la fonction, on va simplement utiliser son nom suivi de parenthèses `()`.

Complément

Fonctions (MDN Web docs)¹

¹ <https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Fonctions>

III. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question

[solution n°1 p.17]

Déclarez une fonction qui aura pour objectif de retourner la valeur de la variable `global FIRST_NAME`. Vous donnerez à votre fonction un nom non équivoque.

```
1 const FIRST_NAME = 'Julien';
```

IV. Les paramètres des fonctions

Objectif

- Savoir utiliser les paramètres d'une fonction

Mise en situation

Nous allons voir comment est déclarée une fonction JavaScript avec des paramètres.

Méthode Fonction JavaScript avec des paramètres

Les parenthèses d'une fonction peuvent éventuellement accueillir des paramètres : il s'agit de variables qui vont être utilisées dans le code de la fonction (cela peut être un nombre, un texte, une liste de valeurs, booléen true/false, etc.). Ces paramètres sont appelés des arguments.

S'il y a plusieurs arguments, alors on les sépare par une virgule suivie d'un espace (pour une meilleure lisibilité).

On souhaite créer une fonction qui nous permet de calculer le carré d'un nombre (square en anglais), le carré d'un nombre est obtenu en multipliant un nombre par lui-même. On cherche donc déclarer une fonction qui multiplie un nombre par lui même (`number*number`).

```
1 function squareNumber(){  
2   console.log(number*number)  
3 }  
4
```

En l'état actuel il n'est pas possible d'appeler notre fonction en donnant à ce nombre une valeur, pour cela il faut faire passer à la fonction `squareNumber()` un argument qui sera reconnu et utilisé par la fonction soit `squareNumber(number)`;

Résultat

```
1 // Square of a number  
2 function squareNumber(number) {  
3   console.log(number * number)  
4 }  
5  
6 // Display in the console : 49  
7 squareNumber(7)  
8
```

1 <https://repl.it/>

```
9 // Display in the console : 256
10 squareNumber(16)
```

Exemple Un exemple avec plusieurs paramètres

```
1 function hello(userName, age) {
2   console.log('Bonjour ' + userName + ' / ' + age + ' ans')
3 }
4
5 // Display in the console : Bonjour John / 36 ans
6 hello('John', 36)
7
8 // Display in the console : Bonjour Jane / 41 ans
9 hello('Jane', 41)
```

On déclare une fonction hello() qui à pour argument un nom d'utilisateur (userName) et un âge (age). Ces arguments sont séparés d'une virgule.

Syntaxe À retenir

- Dans les parenthèses d'une fonction, on peut éventuellement passer un ou plusieurs paramètres, qui seront utilisés dans le code de la fonction.

Complément

Fonctions (MDN Web docs)¹

V. Exercice : Appliquez la notion

Vous créez un site Internet permettant à l'utilisateur de calculer automatiquement des volumes en fonction des formes géométriques. Dans un chapitre consacré aux parallélépipèdes, vous proposez de calculer le volume en fonction de ces dimensions.

Le volume d'un parallélépipède est égal à la hauteur*largeur*profondeur.

Votre fonction sera appelée par le programme ci-dessous :

```
1 const height = prompt('Hauteur du parallélépipèdes ');
2 const width = prompt('Largeur du parallélépipèdes ');
3 const depth = prompt('Profondeur du parallélépipèdes ');
4
5 const volume = cube(height, width, depth);
```

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question

[solution n°2 p.17]

Déclarez avant ce programme une fonction qui retourne le volume d'un parallélépipède.

¹ <https://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Fonctions>

² <https://repl.it/>

VI. Les fonctions JavaScript prédéfinies

Objectif

- Découvrir les possibilités offertes par les fonctions JavaScript prédéfinies

Mise en situation

Le langage JavaScript dispose de nombreuses fonctions que nous pouvons utiliser pour effectuer différentes tâches. Les fonctions définies dans le langage sont appelées fonctions prédéfinies ou fonctions prêtes à l'emploi car il nous suffit de les appeler pour nous en servir.

Pour être tout à fait précis, les fonctions prédéfinies en JavaScript sont des méthodes. Une méthode est tout simplement le nom donné à une fonction définie au sein d'un objet. Pour le moment, nous allons considérer que ce sont simplement des fonctions.

Vous les avez déjà utilisées sans vous en rendre compte à chaque `console.log()` que vous ajoutez dans votre application.

L'intérêt principal des fonctions prédéfinies est de nous permettre de réaliser des opérations complexes de manière très simple : en les appelant, tout simplement. En effet, vous devez bien comprendre que derrière ces noms de fonctions se cachent des codes parfois longs et complexes qui vont être exécutés lorsqu'on appelle la fonction et qui vont permettre de réaliser une opération précise (générer un nombre aléatoire, etc.).

En plus de cela, le code d'une fonction est réutilisable : cela veut dire qu'on va pouvoir appeler une même fonction autant de fois qu'on le souhaite afin qu'elle accomplisse plusieurs fois la même opération.

Méthode `toUpperCase()`

Cette méthode, de l'objet global `String`, modifie une chaîne de caractères afin de la passer en majuscules.

Cet objet dispose de nombreuses autres fonctions pour le traitement des chaînes de caractères (mettre en minuscules, supprimer les espaces blancs en début et fin, récupérer une sous-chaîne, extraire, remplacer, etc.).

Exemple

```
1 let hello = 'hello world'
2
3 // Display in the console : HELLO WORLD
4 console.log(hello.toUpperCase())
```

Méthode `Math.round()`

Arrondit une valeur à l'entier le plus proche.

L'objet `Math` propose de nombreuses fonctions (ex : connaître le plus grand ou le plus petit nombre d'une liste, sinus, tangente, puissance, etc.).

Exemple

```
1 // Display in the console : 0
2 console.log(Math.round(0.4))
3
4 // Display in the console : 1
5 console.log(Math.round(0.5))
6
7 // Display in the console : -1
8 console.log(Math.round(-1.5))
```

Méthode **Number.isInteger()**

Permet de déterminer si la valeur passée est un entier, et renvoie vrai ou faux (`true` / `false`).

L'objet `Number` permet aussi de déterminer si la valeur fournie est un nombre fini, d'analyser et de convertir une chaîne de caractères en un nombre flottant ou un entier, etc.

Exemple

```
1 // Display in the console : true
2 console.log(Number.isInteger(5));
3
4 // Display in the console : false
5 console.log(Number.isInteger(5.05));
```

Syntaxe À retenir

- Avant de créer une fonction, il est important de vérifier s'il n'y a pas déjà une fonction existante qui répond au besoin.
- Pour cela, il existe plusieurs documentations JavaScript dont certaines ont été mises en complément de ce cours.

Complément

String¹

Math²

Number³

VII. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question

[solution n°3 p.17]

En utilisant les méthodes de l'objet `String`, écrivez une fonction qui transforme un mot en mettant une majuscule au début, et le reste en minuscules.

Indice :

Pour cela vous créerez une fonction qui prendra en argument un mot.

La première lettre de ce mot devra être récupérée dans une variable (voir utilisation méthode `substring()`). Une fois fait, elle sera passée en majuscule.

Dans une seconde variable seront récupérées toutes les lettres du mot sauf la première (voir utilisation `substring()`). Une fois récupérés, elles seront passées en minuscule.

Enfin la fonction retournera la concaténation de ses deux variables.

1 https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/String

2 https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Math

3 https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Number

4 <https://repl.it/>

Pour aller plus loin :

On appelle une fonction qui est une propriété d'un objet une méthode. Pour gagner des lignes de code, on peut écrire plusieurs méthodes sur une seule ligne. La méthode prendra alors en compte le résultat de la première méthode.

VIII. La portée des variables

Objectif

- Comprendre la portée des variables dans une fonction

Méthode La portée des variables dans une fonction JavaScript

Nous pouvons définir des variables dans les blocs d'instructions de nos fonctions JavaScript.

Celles-ci ne seront pas disponibles en dehors de la fonction, ce qui permet d'éviter les conflits (par exemple, si des variables dans le code appelant la fonction ou dans d'autres fonctions ont le même nom).

Exemple

```
1 function hello(userName) {
2   let helloMessage = 'Bonjour ' + userName + ' !'
3
4   console.log(helloMessage)
5 }
6
7 // Display in the console : Bonjour John !
8 hello('John')
9
10 // Display in the console : ReferenceError: helloMessage is not defined
11 console.log(helloMessage)
```



```
1 function hello(userName) {
2   let helloMessage = 'Bonjour ' + userName + ' !'
3
4   console.log(helloMessage)
5 }
6
7 let helloMessage = 'Hello world !'
8
9 // Display in the console : Bonjour John !
10 hello('John')
11
12 // Display in the console : Hello world !
13 console.log(helloMessage)
```

Rappel Différences var / let / const

L'instruction Let:

Sa principale caractéristique est sa portée : elle est limitée à celle du bloc courant. Pour rappel, un bloc en Javascript, c'est ce qu'on retrouve entre accolades : une comparaison en if, une boucle while etc...

```
1 function test(){
2   if( 1 === 1 ){
3     let maVar = true;
4     console.log(maVar);
5     // Retourne true
6   }
```

```
7 console.log(maVar);
8 // Retourne une erreur, let n'existe pas hors du bloc "if"
9 }
10
```

L'instruction var:

C'est la plus ancienne, elle existe depuis JavaScript 1.0, L'instruction var permet de déclarer une variable dont la portée est la fonction qui la contient, ou le contexte global si la variable est déclarée en dehors de toute fonction

Exemple

```
1 function showNumber() {
2   let n = 5
3   var x = 10
4
5   if (true) {
6     let n = 7
7     var x = 12
8   }
9
10  // Display in the console : 5
11  console.log(n)
12
13  // Display in the console : 12
14  console.log(x)
15 }
16
17 showNumber()
```

Syntaxe À retenir

- La portée des variables définies dans une fonction s'arrête à la fonction.
- `var` : portée de la fonction ou contexte global.
- `let` : portée du bloc courant.

Bonne pratique

Vous allez continuer de croiser des variables var dans certains projets, en général cette présence est dû à l'ancienneté du code produit. De nos jours les bonnes pratiques tendent vers une utilisation formelle de let et const.

Complément

var¹

let²

1 <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/var>

2 <https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Instructions/let>

IX. Exercice : Appliquez la notion

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question

[solution n°4 p.17]

En modifiant uniquement la déclaration des variables, que faudrait-il faire pour que la console affiche 0 ?

```
1 function count() {
2   var compteur = 0;
3   for (let i = 0 ; i < 10; i++){
4     var compteur = i
5   }
6   console.log(compteur);
7 }
8
9 count(); // 9
```

X. Auto-évaluation

A. Exercice final

Exercice 1

[solution n°5 p.18]

Exercice

Que retournera la console ?

```
1 function random(min, max){
2   return Math.floor(Math.random() * (max - min + 1)) + min;
3 }
4
5 const myNumber= random(10, 100);
6 console.log(myNumber)
```

- ☐ Retourne un nombre entier entre 10 et 100 (10 et 100 exclus)
- ☐ Retourne un nombre entier entre 10 et 100 (10 et 100 inclus)
- ☐ Retourne un nombre décimal arrondi au centième
- ☐ Retourne la multiplication de 10 et de 100

Exercice

Que retournera la console ?

```
1 function Asc(array){
2   return array.sort(function(a, b){
3     return a - b
4   });
5 }
6
7 const myArray = [5, 9, 8, 2, 7];
8 console.log(Asc(myArray));
9
```

1 <https://repl.it/>

- ☐ Un tableau trié dans l'ordre croissant
- ☐ Un tableau trié dans l'ordre décroissant
- ☐ Un tableau trié dans un ordre aléatoire
- ☐ null

Exercice

À quel objet appartient la méthode `getTime()` ?

Exercice

`const myDate = new Date()` retourne la date du jour.

- ☐ Vrai
- ☐ Faux

Exercice

Quelles méthodes permettent d'ajouter un nombre dans un tableau ?

- ☐ `pop()`
- ☐ `push()`
- ☐ `unshift()`
- ☐ `filter()`

Exercice

Que retournera la console ?

```
1 function getNumber(myNumber) {
2   return myNumber
3 }
4
5 function initNumber(myNumber) {
6   return getNumber(myNumber)
7 }
8
9 console.log(initNumber(8));
```

Exercice

Pour les besoins d'un algorithme, je dois créer une fonction. Quelle est la première chose que je vais faire ?

- ☐ Écrire le pseudo-code
- ☐ Me lancer dans l'écriture du code
- ☐ Vérifier dans la documentation que cette fonction n'existe pas déjà sous forme de méthode

Exercice

Une fonction a obligatoirement besoin de paramètres pour fonctionner.

- ☐ Vrai
- ☐ Faux

Exercice

Une fonction retournera toujours une valeur.

- ☐ Vrai
- ☐ Faux

Exercice

Quelle méthode de l'objet `Array` retourne l'index du premier élément qui satisfait une condition ?

B. Exercice : Défi

Vous êtes développeur pour la boutique en ligne d'un créateur de mobilier indépendant. Ce matin, votre client vous téléphone pour vous annoncer que, cette année, il veut industrialiser la mise en place des soldes sur son site vitrine : il ne veut plus jamais calculer tous les prix manuellement. Très à cheval sur les détails, il attend que les prix soient mis à jour dans l'interface, en gardant le même format que les anciens prix.

Vous devez lui proposer une solution qui lui permette de gagner du temps pour les prochaines soldes.

Voici le code HTML et CSS de la galerie :

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width">
6     <title>Les fonctions JavaScript</title>
7     <link href="https://fonts.googleapis.com/css?family=Lato:400,100,300" rel="stylesheet">
8     <link href="style.css" rel="stylesheet" type="text/css"/>
9   </head>
10  <body>
11    <h1>Galerie de créateur</h1>
12    <div class="wrapper">
13      <div class="product">
14        
15        <h2>Chaise maxi confort</h2>
16        <span id="chaise">120,50€</span>
17      </div>
18      <div class="product">
19        
20        <h2>Table en bois exotique</h2>
21        <span id="table">1200,99€</span>
22      </div>
23      <div class="product">
24        
25        <h2>Vase en céramique</h2>
26        <span id="vase">420,01€</span>
27      </div>
28      <div class="product">
29        
30        <h2>Canapé familial en laine</h2>
31        <span id="canape">12585,55€</span>
```

```
32     </div>
33   </div>
34   <script src="script.js"></script>
35 </body>
36 </html>
```

```
1 body {
2   font-family: 'Lato', sans-serif;
3 }
4
5 .wrapper {
6   display: flex;
7 }
8
9 .product {
10  margin: 2rem;
11 }
```

Et voici la liste des produits que vous avez pu récupérer en base de données depuis le back-end du site.

Elle vous servira à effectuer les calculs sur les prix depuis le code JavaScript.

```
1 let chaise = 120.50
2 let table = 1200.99
3 let vase = 420.01
4 let canape = 12585.55
```

Galerie de créateur

			
150 x 150	150 x 150	150 x 150	150 x 150
Chaise maxi confort	Table en bois exotique	Vase en céramique	Canapé familial en laine
120,50€	1200,99€	420,01€	12585,55€

Pour réaliser cet exercice, vous aurez besoin de travailler sur l'environnement de travail :



Question 1

[solution n°6 p.21]

Vous allez réaliser ce travail en plusieurs étapes, qui, mises bout à bout, vous permettront de répondre au besoin du client.

Dans un premier temps, vous allez devoir écrire la fonction qui permet de calculer le nouveau prix d'un seul produit.

La promotion doit être de 25 % du prix initial.

Afin de tester son bon fonctionnement, vous appellerez cette fonction avec le prix de la chaise et afficherez le résultat dans la console du navigateur.

Indice :

Vous allez devoir déclarer une fonction prenant deux paramètres : le prix avant promotion et le taux de promotion.

1 <https://repl.it/>

Question 2

[solution n°7 p.21]

Vous vous souvenez que votre client est très pointilleux et qu'il remarquera que le format du prix n'est pas le bon. Là encore, une fonction vous permettra de remplacer le point du prix par une virgule, et vous pourrez ajouter le symbole "€".

Cette fois-ci, vous testerez directement votre fonction dans la console du navigateur.

Indice :

La méthode `replace()` ne peut s'utiliser que sur un `String` : il faudra d'abord *caster* le prix pour l'utiliser.

Question 3

[solution n°8 p.21]

Vous remarquez que le prix après promotion affiche trop de chiffres après la virgule.

L'année dernière, quand vous avez dû calculer tous les prix promotionnels manuellement, le client vous avait indiqué de toujours arrondir cette partie du prix à l'entier supérieur.

Vous écrirez donc une fonction qui vous permettra d'obtenir le prix arrondi, à partir du prix promotionnel.

Indice :

L'objet `Math` et ses fonctions prédéfinies pourront vous être utiles.

Question 4

[solution n°9 p.21]

Maintenant que vous avez une base solide qui fonctionne pour le produit "chaise", il faut appliquer ce traitement à tous les produits.

Pour cela, vous bouclerez sur cette liste et appellerez les fonctions précédentes en passant les bons paramètres.

Pour cet exercice vous utiliserez l'array suivant :

```
1 const prices = [120.50, 1200.99, 420.01, 12585.55]
```

Puis vous bouclerez sur cet array et appellerez les fonctions précédentes en passant les bons paramètres et afficherez les prix au bon format dans la console du navigateur.

Solutions des exercices

p.5 Solution n°1

```
1 const FIRST_NAME = 'Julien';
2
3 function getName() {
4     return FIRST_NAME;
5 }
6
7 console.log(getName());
```

p.6 Solution n°2

```
1 function cube(height, width, depth) {
2     return height * width * depth
3 }
4
5 const height = prompt('Hauteur du parallélépipèdes ');
6 const width = prompt('Largeur du parallélépipèdes ');
7 const depth = prompt('Profondeur du parallélépipèdes ');
8
9 const volume = cube(height, width, depth);
```

p.8 Solution n°3

```
1 function transformText(word) {
2     const firstLetter = word.substring(0, 1) // Récupère la première lettre
3     const firstLetterMaj = firstLetter.toUpperCase() // et la transforme en
    majuscule
4     const otherLetters = word.substring(1) // Récupère les autres lettres
5     const otherLettersMin = otherLetters.toLowerCase() //et les transforme en
    minuscule
6     return firstLetterMaj+otherLettersMin // Concatenation des deux variables
7 }
8
9 //Pour aller plus loin
10 function transformTextVersion2(word) {
11     const firstLetter = word.substring(0, 1).toUpperCase() // Récupère la première lettre
    et la transforme en majuscule
12     const otherLetters = word.substring(1).toLowerCase() // Récupère les autres lettres
    et les transforme en minuscule
13     return `${firstLetter}${otherLetters}` // Concatenation des deux variables
14 }
15
16 const newWord = transformText('MAGIQUE');
17 const newWord2 = transformTextVersion2('MAGIQUEVERSIONDEUX');
18
19 console.log(newWord);
20 console.log(newWord2);
```

p.11 Solution n°4

La boucle `for` possède son propre scope. Si nous déclarons la variable `compteur` avec `let` plutôt que `var`, alors la valeur de `compteur` ne sera pas modifiée dans la boucle et la console retournera 0.


```
1 function count() {
2     let compteur = 0;
3     for (let i = 0 ; i < 10; i++){
4         let compteur = i
5     }
6     console.log(compteur);
7 }
8
9 count(); // 0
```

Exercice p. 11 Solution n°5

Exercice

Que retournera la console ?

```
1 function random(min, max){
2     return Math.floor(Math.random() * (max - min + 1)) + min;
3 }
4
5 const myNumber= random(10, 100);
6 console.log(myNumber)
```

- ☐ Retourne un nombre entier entre 10 et 100 (10 et 100 exclus)
 - ☒ Retourne un nombre entier entre 10 et 100 (10 et 100 inclus)
 - ☐ Retourne un nombre décimal arrondi au centième
 - ☐ Retourne la multiplication de 10 et de 100
-  `Math.floor()` retourne le plus grand entier inférieur ou égal à un nombre.
`Math.random()` retourne un nombre flottant compris entre 0 et 1 (1 exclu).

Si on décompose la fonction :

```
1 const max = 10
2 const min = 1
3
4 let myFirstNumber= Math.random() * (10 - 1 + 1)
5 // => myFirstNumber= Math.random() * 10
6 // => 0.85682 * 10 ou 0.778952 * 10 ou 0.4456238 * 10 etc...
7 // prenons myFirstNumber= 8.5682
8
9 const mySecondNumber= Math.floor(myFirstNumber)
10 console.log(mySecondNumber) // 8
11
12
```

Exercice

Que retournera la console ?

```
1 function Asc(array){
2     return array.sort(function(a, b){
3         return a - b
4     });
5 }
```

```

6
7 const myArray = [5, 9, 8, 2, 7];
8 console.log(Asc(myArray));
9

```

- ☒ Un tableau trié dans l'ordre croissant
- ☐ Un tableau trié dans l'ordre décroissant
- ☐ Un tableau trié dans un ordre aléatoire
- ☐ null



La méthode `sort` appartient à l'objet `Array`. Elle permet de trier un tableau.

Pour trier le tableau de manière décroissante, il suffit d'inverser les arguments dans la fonction :

```

1 function Desc(array){
2   return array.sort(function(a, b){
3     return b - a
4   });
5 }
6
7 const myArray = [5, 9, 8, 2, 7];
8 console.log(Desc(myArray));
9 // [9, 8, 7, 5, 2]

```

Exercice

À quel objet appartient la méthode `getTime()` ?

`Date()`

Exercice

`const myDate = new Date()` retourne la date du jour.

- ☒ Vrai
- ☐ Faux

Exercice

Quelles méthodes permettent d'ajouter un nombre dans un tableau ?

- ☐ `pop()`
La méthode `pop()` supprime le dernier élément d'un tableau et le retourne.
- ☒ `push()`
La méthode `push()` ajoute un élément à la fin d'un tableau.
- ☒ `unshift()`
La méthode `unshift()` ajoute un élément au début d'un tableau.
- ☐ `filter()`
La méthode `filter()` permet de retourner un tableau filtré en fonction de conditions.

Exercice

Que retournera la console ?

```
1 function getNumber(myNumber){
2   return myNumber
3 }
4
5 function initNumber(myNumber) {
6   return getNumber(myNumber)
7 }
8
9 console.log(initNumber(8));
```

8



Une fonction peut appeler une autre fonction.

Dans le cas présent, la fonction `initNumber()` retourne le résultat retourné par `getNumber()`.

Exercice

Pour les besoins d'un algorithme, je dois créer une fonction. Quelle est la première chose que je vais faire ?

- ☐ Écrire le pseudo-code
C'est une très bonne pratique. Avant de vous lancer dans un code compliqué, il ne faut pas hésiter à écrire le pseudo-code. Cela fait gagner du temps et met en évidence les étapes du développement.
- ☐ Me lancer dans l'écriture du code
Se lancer dans l'écriture d'un code sans y avoir réfléchi au préalable vous induira forcément à un moment ou à un autre en erreur, et vous fera perdre beaucoup de temps au final.
- ☒ Vérifier dans la documentation que cette fonction n'existe pas déjà sous forme de méthode

Exercice

Une fonction a obligatoirement besoin de paramètres pour fonctionner.

- ☐ Vrai
Une fonction peut très bien s'écrire sans paramètre.
- ☒ Faux

Exercice

Une fonction retournera toujours une valeur.

- ☐ Vrai
Pour qu'une fonction retourne une valeur, il faut mettre le mot-clé `return` suivi de la valeur à retourner à la fin de celle-ci. Mais ce n'est pas une obligation.
- ☒ Faux

Exercice

Quelle méthode de l'objet `Array` retourne l'index du premier élément qui satisfait une condition ?

`findIndex()`

p. 14 Solution n°6

```
1 function getPromoPrice(price, percent) {  
2   return price - (price * percent / 100)  
3 }  
4  
5 console.log(getPromoPrice(120.50, 25))
```

p. 15 Solution n°7

```
1 function getPromoPrice(price, percent) {  
2   return price - (price * percent / 100)  
3 }  
4  
5 function roundDecimal(number){  
6   return Math.ceil( number)  
7 }  
8  
9 function formatPrice(price) {  
10  return price.toString().replace(".", ",") + " €"  
11 }  
12  
13 let newPromoPrice = getPromoPrice(120.50, 25)  
14 let roundedPrice = roundDecimal(newPromoPrice)  
15  
16 let finalPrice = formatPrice(roundedPrice)  
17 console.log(finalPrice)
```

p. 15 Solution n°8

```
1 function getPromoPrice(price, percent) {  
2   return price - (price * percent / 100)  
3 }  
4  
5 function roundDecimal(number){  
6   return Math.ceil(number)  
7 }  
8  
9 let newPromoPrice = getPromoPrice(120.50, 25)  
10 console.log(roundDecimal(newPromoPrice))
```

p. 15 Solution n°9

```
1 const prices = [120.50,1200.99,420.01,12585.55]  
2  
3 function getPromoPrice(price, percent) {  
4   return price - (price * percent / 100)  
5 }  
6  
7 function roundDecimal(number){  
8   return Math.ceil(number)  
9 }  
10  
11 function formatPrice(price) {
```

```

12     return price.toString().replace(".", ",") + " €"
13 }
14
15 for (let i = 0; i < prices.length; i++) {
16
17     let newPromoPrice = getPromoPrice(prices[i], 25)
18     let roundedPrice = roundDecimal(newPromoPrice)
19     let finalPrice = formatPrice(roundedPrice)
20     console.log(finalPrice)
21
22 }
23
24 // autre solution
25
26 prices.forEach(function(price)
27 {
28     let newPromoPrice = getPromoPrice(price, 25)
29     let roundedPrice = roundDecimal(newPromoPrice)
30     let finalPrice = formatPrice(roundedPrice)
31     console.log(finalPrice)
32 }
33 );

```

Galerie de créateur



Chaise maxi confort

90,38 €



Table en bois exotique

900,74 €



Vase en céramique

315,01 €



Canapé familial en laine

9439,16 €