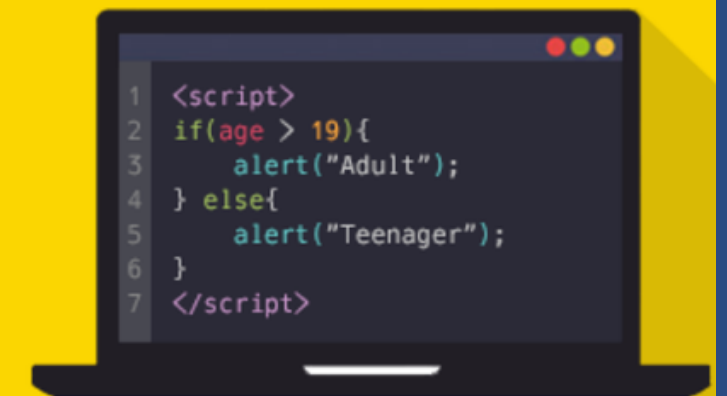


# Javascript

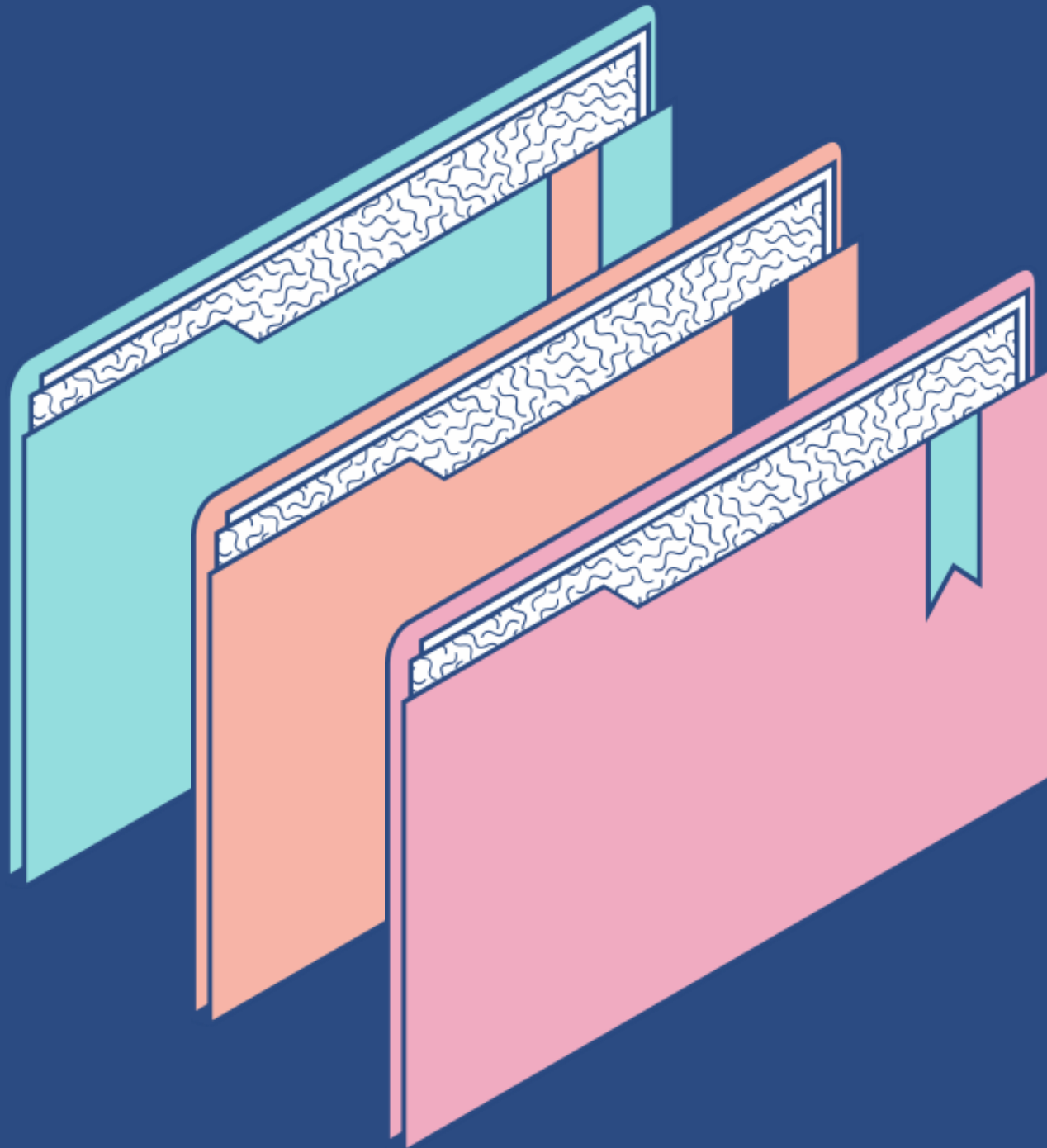


**JavaScript**



# SOMMAIRE

- Présentation du JS
- Variables
- Structures de contrôle
- Fonctions
- POO en JS
- Valeurs primitives
- Manipulation du BOM
- Manipulation du DOM
- Fonctions avancées
- Gestion des erreurs
- Stockage de données persistantes
- Canvas
- Asynchrone



# Valeurs primitives et objets globaux

57

shop.js:68

▼ [] ⓘ

- ▶ 0: {id: 1, name: "Simple générateur", description: "Un simple générateur..."}
  - ▶ 1: {id: 2, name: "Générateur", description: "Un générateur", price: "10"}
  - ▶ 2: {id: 3, name: "Générateur MK1", description: "Un générateur performant..."}
  - ▶ 3: {id: 4, name: "Générateur MK2", description: "Un générateur assez performant..."}
  - ▶ 4: {id: 5, name: "Super générateur", description: "Un générateur très performant..."}
  - ▶ 5: {id: 6, name: "mega générateur", description: "Génération...", price: "20"}
  - ▶ 6: {id: 7, name: "Ultime générateur", description: "Génération...", price: "30"}
- length: 7
- ▶ \_\_proto\_\_: Array(0)



# Valeurs primitives

Une valeur primitive est une valeur qui n'est pas un objet et ne peut pas être modifiée.

Chaque type de valeur primitive (à l'exception de null et undefined) possède un objet JS prédéfini : String, Number, Boolean, Symbol.

Chaque type contient des propriétés et des méthodes.

```
//On définit une valeur primitive  
let ch1 = 'Une chaine de caractères';  
  
//On appelle le constructeur String() pour créer un objet String  
let ch2 = new String('Une chaine de caractères');  
  
alert('Type de ch1 : ' + typeof(ch1) + '\nType de ch2 : ' + typeof(ch2));
```



# Valeurs primitives vs objet

En utilisant un objet, vous aurez accès à toutes ses propriétés et ses méthodes.

Cependant, il est plus optimisé de créer des valeurs primitives que de déclarer des objets.

```
//On définit une valeur primitive
let ch1 = 'Une chaine de caractères';

//On appelle le constructeur String() pour créer un objet String
let ch2 = new String('Une chaine de caractères');

//La propriété length compte la longueur de la chaine
document.getElementById('p1').innerHTML = ch2.length;

//La méthode toUpperCase() renvoie la chaine en majuscules sans modifier l'objet
document.getElementById('p2').innerHTML = ch2.toUpperCase();

document.getElementById('p3').innerHTML = ch2;
```

# Valeurs primitives vs objet

En pratique, il est possible d'utiliser les propriétés et méthodes relatifs à vos objets sur vos valeurs primitives !

Ainsi, vous pourrez allier l'optimisation des valeurs primitives avec la puissance des objets.

```
//On définit une valeur primitive
let ch1 = 'Une chaine de caractères';

//On appelle le constructeur String() pour créer un objet String
let ch2 = new String('Une chaine de caractères');

//La propriété length compte la longueur de la chaine
document.getElementById('p1').innerHTML = ch1.length;

//La méthode toUpperCase() renvoie la chaine en majuscules sans modifier l'objet
document.getElementById('p2').innerHTML = ch1.toUpperCase();

document.getElementById('p3').innerHTML = ch1;
```



# Objet String

L'objet String possède qu'une propriété length.

Vous pourrez ainsi récupérer la taille d'une chaîne avec `chaîne.length`

```
let ch1 = 'Pierre';  
let ch2 = 'Pierre Giraud'; // L'espace est un caractère  
  
// La propriété length renvoie la longueur d'une chaîne  
document.getElementById('p1').innerHTML = 'ch1.length : ' + ch1.length;  
document.getElementById('p2').innerHTML = 'ch2.length : ' + ch2.length;
```



# Objet String

L'objet String possède une 30e de méthodes spécifiques

```
let prez = 'Bonjour, je m'appelle Pierre et j'ai 29 ans';

/*Si "Pierre" est trouvé dans la chaîne stockée dans prez, includes() renvoie true
*et on exécute le code de la condition. Dans le cas contraire, includes() renvoie
*false et le code n'est pas exécuté*/
if(prez.includes('Pierre')){
    document.getElementById('p1').textContent = '"Pierre" présent dans let prez';
}

if(prez.startsWith('Bonjour')){
    document.getElementById('p1').textContent = 'La chaîne commence par "Bonjour"';
}

if(prez.endsWith('29 ans')){
    document.getElementById('p2').textContent = 'La chaîne se termine par "29 ans"';
}
```







# Exercice d'application : String

Créez une nouvelle fonction `testString`, prenant en entrée 2 chaînes de caractères. Si l'un des chaînes est incluse dans l'autre, la fonction retournera `true`, sinon elle retournera `false`.  
Tester cette fonction en demandant à l'utilisateur de saisir 2 chaînes de caractères, et affichés le retour.



# Objet Number

Les propriétés à connaître sont les suivantes :

- Les propriétés `MIN_VALUE` et `MAX_VALUE` plus petite et plus grande valeur possible en JS ;
- Les propriétés `NEGATIVE_INFINITY` et `POSITIVE_INFINITY` servent respectivement à représenter l'infini côté négatif et côté positif ;
- La propriété `NaN` représente une valeur qui n'est pas un nombre (« NaN » est l'abréviation de « Not a Number ») et est équivalente à la valeur `NaN`.

# Objet Number

```
alert(  
  'MIN_VALUE : ' + Number.MIN_VALUE  
+ '\nMAX_VALUE : ' + Number.MAX_VALUE  
+ '\nMIN_SAFE_INTEGER : ' + Number.MIN_SAFE_INTEGER  
+ '\nMAX_SAFE_INTEGER : ' + Number.MAX_SAFE_INTEGER  
+ '\nNEGATIVE_INFINITY : ' + Number.NEGATIVE_INFINITY  
+ '\nPOSITIVE_INFINITY : ' + Number.POSITIVE_INFINITY  
+ '\nNaN : ' + Number.NaN  
);
```

This page says

MIN\_VALUE : 5e-324  
MAX\_VALUE : 1.7976931348623157e+308  
MIN\_SAFE\_INTEGER : -9007199254740991  
MAX\_SAFE\_INTEGER : 9007199254740991  
NEGATIVE\_INFINITY : -Infinity  
POSITIVE\_INFINITY : Infinity  
NaN : NaN

OK



# Objet Number

L'objet String possède une 10e de méthodes spécifiques

```
let nb1 = 10;
let nb2 = Number.POSITIVE_INFINITY;

if(Number.isFinite(nb1)){
    document.getElementById('p1').textContent = 'Le nombre ' + nb1 + ' est fini';
}

if(Number.isInteger(nb1)){
    document.getElementById('p1').textContent = 'Le nombre ' + nb1 + ' est entier';
}

if(Number.isNaN(nb1)){
    document.getElementById('p1').textContent = 'nb1 stocke la valeur NaN';
}

document.getElementById('p1').textContent = Number.parseFloat(nb1);
document.getElementById('p1').textContent = Number.parseInt('0F', 16);
document.getElementById('p1').textContent = nb1.toString(16);
```





# Objet Math

L'objet Math stocke des constantes mathématiques utiles :

Math.E a pour valeur le nombre d'Euler (exponentielle de 1), soit environ 2,718 ;

Math.LN2 a pour valeur le logarithme naturel de 2, soit environ 0,693 ;

Math.LN10 a pour valeur le logarithme naturel de 10, soit environ 2,302 ;

Math.LOG2E a pour valeur le logarithme de e en base 2, soit environ 1,442;

Math.LOG10E a pour valeur le logarithme de e en base 10, soit environ 0,434 ;

Math.PI a pour valeur pi, soit environ 3,14159 ;

Math.SQRT1\_2 a pour valeur la racine carrée de  $\frac{1}{2}$ , soit environ 0,707 ;

Math.SQRT2 a pour valeur la racine carrée de 2, soit environ 1,414.

# Objet Math

```
document.getElementById('p1').innerHTML =
  'Math.E : ' + Math.E
+ '<br>Math.LN2 : ' + Math.LN2
+ '<br>Math.LN10 : ' + Math.LN10
+ '<br>Math.LOG2E : ' + Math.LOG2E
+ '<br>Math.LOG10E : ' + Math.LOG10E
+ '<br>Math.PI : ' + Math.PI
+ '<br>Math.SQRT1_2 : ' + Math.SQRT1_2
+ '<br>Math.SQRT2 : ' + Math.SQRT2;
```

## Titre principal

Un paragraphe

Math.E : 2.718281828459045

Math.LN2 : 0.6931471805599453

Math.LN10 : 2.302585092994046

Math.LOG2E : 1.4426950408889634

Math.LOG10E : 0.4342944819032518

Math.PI : 3.141592653589793

Math.SQRT1\_2 : 0.7071067811865476

Math.SQRT2 : 1.4142135623730951



# Objet Math

L'objet Math possède également de nombreuses méthodes de calcul utiles :

- $\text{floor}(n)$  arrondit la valeur  $n$  à l'entier inférieur ou égal à la valeur
- $\text{ceil}(n)$  arrondit la valeur à l'entier supérieur ou égal à la valeur
  - $\text{round}(n)$  arrondit la valeur à l'entier le plus proche
  - $\text{trunc}(n)$  ignore la partie décimale d'un nombre
  - $\text{random}()$  génère un nombre aléatoire en 0 et 1
  - $\text{min}(n)$  renvoi le plus petit nombre de la série  $n$
  - $\text{max}(n)$  renvoi le plus grand nombre de la série  $n$ 
    - $\text{abs}(n)$  renvoi la valeur absolue d'un nombre
    - $\text{exp}()$ ,  $\text{log}()$ ,  $\text{sin}()$ ,  $\text{cos}()$



# Objet Math

```
let nb1 = 12.3456;
let nb2 = 2.45;
let nb3 = 2.54;

document.getElementById('p1').innerHTML =
'Nombre : ' + nb1 +
'<br>floor() : ' + Math.floor(nb1) +
'<br>ceil() : ' + Math.ceil(nb1) +
'<br>round() : ' + Math.round(nb1) +
'<br>trunc() : ' + Math.trunc(nb1);

document.getElementById('p2').innerHTML =
'Nombre : ' + nb2 +
'<br>floor() : ' + Math.floor(nb2) +
'<br>ceil() : ' + Math.ceil(nb2) +
'<br>round() : ' + Math.round(nb2) +
'<br>trunc() : ' + Math.trunc(nb2);

document.getElementById('p3').innerHTML =
'Nombre : ' + nb3 +
'<br>floor() : ' + Math.floor(nb3) +
'<br>ceil() : ' + Math.ceil(nb3) +
'<br>round() : ' + Math.round(nb3) +
'<br>trunc() : ' + Math.trunc(nb3);
```

## Titre principal

Un paragraphe

Nombre : 12.3456  
 floor() : 12  
 ceil() : 13  
 round() : 12  
 trunc() : 12

Nombre : 2.45  
 floor() : 2  
 ceil() : 3  
 round() : 2  
 trunc() : 2

Nombre : 2.54  
 floor() : 2  
 ceil() : 3  
 round() : 3  
 trunc() : 2



# Objet Math

```
//Renvoie un nombre décimal aléatoire entre  
0 et 1 et l'affiche dans p id='p1'  
document.getElementById('p1').innerHTML =  
Math.random();  
  
/*Renvoie un nombre décimal aléatoire entre  
0 et 1, multiplie ce nombre par  
*100 et l'affiche dans p id='p2'*/  
document.getElementById('p2').innerHTML =  
Math.random()*100;  
  
/*Renvoie un nombre décimal aléatoire entre  
0 et 1, multiplie ce nombre par  
*100 puis l'arrondi à l'entier le plus  
proche avec Math.round() et l'affiche  
*dans p id='p3'*/  
document.getElementById('p3').innerHTML =  
Math.round(Math.random()*100);
```


## Titre principal

Un paragraphe

0.027883721131296335

46.56057426372253

70



# Exercice d'application n°5 : String, Number, Math

Créez une nouvelle fonction au sein de votre fichier

Générez un nombre réel à 1 virgule, aléatoire, et compris entre -100 et 100.

Mettez en place un algorithme qui demandera à l'utilisateur de saisir une valeur tant que celle-ci n'est pas égale à la valeur aléatoire.

L'algorithme affichera "Plus petit" ou "Plus grand" en fonction de la valeur saisie.

Si l'utilisateur trouve la valeur exacte, un message de succès s'affiche et la fonction se termine.



# Objet Array (tableaux)

Les tableaux sont des éléments pouvant contenir plusieurs valeurs.

Chaque clé d'un tableau va contenir une valeur.

Nous pourrions ainsi récupérer une valeur à l'aide de l'indice du tableau (tab[i])

**! Un tableau commence toujours à l'indice 0 !**

```
let prenom = ['Pierre', 'Mathilde', 'Florian', 'Camille'];  
let ages = [29, 27, 29, 30];  
let produits = ['Livre', 20, 'Ordinateur', 5, ['Magnets', 100]];  
  
document.getElementById('p1').innerHTML = prenom[0] + ' possède 1 ' + produits[2];  
document.getElementById('p2').innerHTML = prenom[1] + ' a ' + ages[1] + ' ans';  
document.getElementById('p3').innerHTML = produits[4][1] + ' ' + produits[4][0];
```



# Objet Array (tableaux)

Pour parcourir l'ensemble d'un tableau, on utilise la boucle "for ... of"

```
let prenom = ['Pierre', 'Mathilde', 'Florian', 'Camille'];  
let ages = [29, 27, 29, 30];  
let produits = ['Livre', 20, 'Ordinateur', 5, ['Magnets', 100]];  
  
for(let valeur of prenom){  
  document.getElementById('p1').innerHTML += valeur + '<br>';  
}
```

Un paragraphe

Pierre  
Mathilde  
Florian  
Camille



# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `push(x)` : ajoute l'élément `x` à la fin du tableau. Return la nouvelle taille
- `pop()` : supprime le dernier élément du tableau. Return la valeur supprimée

```
let prenom = ['Pierre', 'Mathilde'];
let ages = [29, 27, 32];

/*On ajoute 2 éléments à "prenom" et on récupère la nouvelle taille du tableau
*renvoyée par push() dans une variable "taille"*/
let taille = prenom.push('Florian', 'Camille');

//On supprime le dernier élément de ages et on récupère l'élément supprimé dans del
let del = ages.pop();

document.getElementById('p1').innerHTML = taille + ' éléments dans prenom';
document.getElementById('p2').innerHTML = del + ' supprimé de ages';
```

# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `unshift(x)` : ajoute un élément en début de tableau. Return la nouvelle taille
- `shift()` : supprime le premier élément du tableau. Return l'élément supprimé

```
let prenom = ['Pierre', 'Mathilde'];
let ages = [29, 27, 32];

/*On ajoute 2 éléments au début de "prenom" et on récupère la nouvelle taille
*du tableau renvoyée par push() dans une variable "taille"*/
let taille = prenom.unshift('Florian', 'Camille');

//On supprime le premier élément de ages et on récupère l'élément supprimé dans del
let del = ages.shift();

document.getElementById('p1').innerHTML = prenom;
document.getElementById('p2').innerHTML = taille + ' éléments dans prenom';
document.getElementById('p3').innerHTML = del + ' supprimé de ages';
```

# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `splice(x,y,z)` : remplace/ajoute les éléments `z`, à partir de la position `x`, en remplaçant `y` éléments.

```
let prenom = ['Pierre', 'Mathilde', 'Florian', 'Camille'];
let ages = [29, 27, 28, 30];

/*On insère 'Thomas' et 'Manon' dans le tableau prenom, après le deuxième élément
*(Mathilde) et sans supprimer d'éléments*/
prenom.splice(2, 0, 'Thomas', 'Manon');

/*On supprime les deux éléments après le premier ( c'est à dire 27 et 28) et on
*insère 35 après le premier élément (29)*/
let del = ages.splice(1, 2, 35);
```





# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `join(x)` : Return une chaîne de caractère créée en concaténant toutes les valeurs d'un tableau séparées par x

```
let prenom = ['Pierre', 'Mathilde', 'Florian', 'Camille'];
let ages = [29, 27, 28, 30];

/*On insère 'Thomas' et 'Manon' dans le tableau prenom, après le deuxième élément
*(Mathilde) et sans supprimer d'éléments*/
prenom.splice(2, 0, 'Thomas', 'Manon');

/*On supprime les deux éléments après le premier ( c'est à dire 27 et 28) et on
*insère 35 après le premier élément (29)*/
let del = ages.splice(1, 2, 35);
```



# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `slice(x,y)` : Return sous tableau de y éléments créé en découpant le tableau de départ de x éléments.

```
let prenom = ['Pierre', 'Mathilde', 'Florian', 'Camille'];
let ages = [29, 27, 28, 30];

/*On insère 'Thomas' et 'Manon' dans le tableau prenom, après le deuxième élément
*(Mathilde) et sans supprimer d'éléments*/
prenom.splice(2, 0, 'Thomas', 'Manon');

/*On supprime les deux éléments après le premier ( c'est à dire 27 et 28) et on
*insère 35 après le premier élément (29)*/
let del = ages.splice(1, 2, 35);
```



# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `concat(x,y)` : Return un tableau concaténant le tableau y à la suite du tableau x.

```
let prenom = ['Pierre', 'Mathilde', 'Thomas', 'Manon', 'Florian', 'Camille'];  
let ages = [29, 27, 28, 30];  
let sports = ['Trail', 'Triathlon', 'Natation'];  
  
let tbglobal = prenom.concat(ages, sports);
```



# Objet Array (tableaux)

L'objet Array possède de nombreuses méthodes utiles :

- `includes(x)` : Return true si x est inclu dans le tableau, false sinon.

La méthode est sensible à la casse de x

```
let prenom = ['Pierre', 'Mathilde', 'Thomas', 'Manon', 'Florian', 'Camille'];  
let ages = [29, 27, 28, 30];  
let sports = ['Trail', 'Triathlon', 'Natation'];  
  
let tbglobal = prenom.concat(ages, sports);
```





# Exercice d'application : Array

Soit le tableau : `let semaine = [ 'lun', 'mra', 'mer', 'jeu', 'ven', 'sam' ]`

1. Retirer la dernière valeur du tableau semaine
2. Afficher toutes les valeurs du tableau
3. Ajouter la valeur 'dim' à la fin du tableau
4. Remplacer la valeur 'mra' par 'mar'
5. Afficher le nombre de valeurs du tableau en utilisant la méthode `document.write`
6. Afficher la troisième valeur du tableau



# Objet Date

L'objet Date possède de nombreuses méthodes pour obtenir ou définir une date.

Il existe différentes façon d'instancier notre classe Date :

- `Date()` : créé un objet date contenant la date actuelle complète (en fonction de l'heure locale)
- `Date(x)` où `x` est une date littérale (formatée en anglais) : créé un objet date basée sur la date `x` fournie (méthode déconseillée car présente des variations en f° des navigateurs)
- `Date(n)` où `n` représente le nombre de millisecondes écoulées de puis le 1e janvier 1970 : créé un objet en fonction de la duréen passé en paramètre.



# Objet Date

L'objet Date possède de nombreuses méthodes pour obtenir ou définir une date.

Il existe différentes façon d'instancier notre classe Date :

- Date(yyyy, mm, dd, hh, min, s, ms) : où :
  - yyyy représente l'année (obligatoire)
  - mm représente le mois (obligatoire) en 0 janvier et 11 décembre
  - dd représente le jour (facultatif) entre 1 et 31
  - hh représente l'heure (facultatif) entre 0 et 23
  - min représente les minutes (facultatif) entre 0 et 59
  - s représente les secondes (facultatif) entre 0 et 59
  - ms représente les millisecondes (facultatif) entre 0 999





# Objet Date

```
let date1 = new Date();  
let date2 = new Date('March 23, 2019 20:00:00');  
let date3 = new Date(1553466000000);  
let date4 = new Date(2019, 0, 25, 12, 30);
```

Date 1 : Sun Mar 24 2019 23:38:14 GMT+0100 (Central European Standard Time)  
Date 2 : Sat Mar 23 2019 20:00:00 GMT+0100 (Central European Standard Time)  
Date 3 : Sun Mar 24 2019 23:20:00 GMT+0100 (Central European Standard Time)  
Date4 : Fri Jan 25 2019 12:30:00 GMT+0100 (Central European Standard Time)



# Objet Date

## Getters de Date :

- `getDay()` renvoie le jour de la semaine sous forme de chiffre (avec 0 pour dimanche, 1 pour lundi et 6 pour samedi) pour la date spécifiée selon l'heure locale ;
  - `getDate()` renvoie le jour du mois en chiffres pour la date spécifiée selon l'heure locale ;
- `getMonth()` renvoie le numéro du mois de l'année (avec 0 pour janvier, 1 pour février, 11 pour décembre) pour la date spécifiée selon l'heure locale ;
  - `getFullYear()` renvoie l'année en 4 chiffres pour la date spécifiée selon l'heure locale ;
  - `getHours()` renvoie l'heure en chiffres pour la date spécifiée selon l'heure locale ;
  - `getMinutes()` renvoie les minutes en chiffres pour la date spécifiée selon l'heure locale ;
  - `getSeconds()` renvoie les secondes en chiffres pour la date spécifiée selon l'heure locale ;
- `getMilliseconds()` renvoie les millisecondes en chiffres pour la date spécifiée selon l'heure locale.



# Objet Date

```
let date1 = new Date(2019, 0, 25, 12, 30, 15);  
  
let jourSemaine = date1.getDay();  
let jourMois = date1.getDate();  
let mois = date1.getMonth();  
let annee = date1.getFullYear();  
let heures = date1.getHours();  
let heuresUTC = date1.getUTCHours();  
let minutes = date1.getMinutes();  
let secondes = date1.getSeconds();  
let ms = date1.getMilliseconds();
```

Date : Fri Jan 25 2019 12:30:15 GMT+0100 (Central European Standard Time)  
Jour de la semaine : 5  
Jour du mois : 25  
Numéro du mois : 0  
Année : 2019  
Heures : 12 (heure UTC : 11)  
Minutes : 30  
Secondes : 15  
Millisecondes : 0





# Objet Date

## Setters de Date :

- setDate() définit le jour du mois en chiffres pour la date spécifiée selon l'heure locale
- setMonth() définit le numéro du mois de l'année (avec 0 pour janvier, 1 pour février, 11 pour décembre) pour la date spécifiée selon l'heure locale;
- setFullYear() définit année en 4 chiffres pour la date spécifiée selon l'heure locale ;
  - setHours() définit l'heure en chiffres pour la date spécifiée selon l'heure locale ;
- setMinutes() définit les minutes en chiffres pour la date spécifiée selon l'heure locale;
- setSeconds() définit les secondes en chiffres pour la date spécifiée selon l'heure locale
- setMilliseconds() définit les millisecondes en chiffres pour la date spécifiée selon l'heure locale

Toutes les méthodes existes également avec UTC entre le getter et le setter pour définir selon l'UTC et non le local

# Objet Date

## Setters de Date :

```
//On crée une date  
let date1 = new Date(2019, 0, 25, 12, 30, 15);  
  
//On modifie la date  
date1.setDate(31);  
date1.setMonth(2);  
date1.setFullYear(2018);  
date1.setHours(10);  
date1.getUTCHours();  
date1.setMinutes(0);  
date1.setSeconds(0);  
date1.setMilliseconds(0);
```

Date : Sat Mar 31 2018 10:00:00 GMT+0200 (Central European Summer Time)



# Objet Date

Convertir une date au format local :

L'objet Date dispose de 3 méthodes spécifiques pour convertir un objet Date dans un format local ( français par exemple )

- toLocaleDateString() : Return "jour-mois-année" en fonction d'une locale et d'options.
- toLocaleTimeString() : Return "heures-minutes-secondes" en fonction d'une locale et d'options.
- toLocaleString() : Return toute la date en fonction d'une locale et d'options.

La locale permet de définir la langue. Pour la France, on utilisera fr-FR.

Les options permettent de modifier le comportement des fonctions ( 12 ou 24 heures par exemple)





# Objet Date

```
//On crée une date  
let date1 = new Date();  
  
let dateLocale = date1.toLocaleString('fr-FR',{  
  weekday: 'long',  
  year: 'numeric',  
  month: 'long',  
  day: 'numeric',  
  hour: 'numeric',  
  minute: 'numeric',  
  second: 'numeric'});
```

Date : lundi 25 mars 2019 à 10:51:39



## Exercice d'application n°6 : Array, Date

Créez une nouvelle fonction au sein de votre fichier.

Tant que l'utilisateur ne rentre pas la valeur 0, la fonction lui demandera de saisir un nom.

Ce nom sera ajouté au sein d'un tableau.

Lorsque l'utilisateur appui sur 0, la fonction tire au sort l'un des noms saisi.

La fonction affichera : "Le tirage au sort du DATE FR a désigné comme grand gagnant NOM"