1. La declaration et l’appel des variables :
2. Syntaxe de déclaration :

**type** nom\_variable = valeur ;

Console.log (‘’message’’+nom\_variable) ;

1. Var : pour déclarer une variable, caractérisé par une portée global et la redéclaration.
2. Let : pour déclarer une variable, caractérisé par portée de bloc et pas de redéclaration.
3. Const : pour déclarer une constante.
4. Console.log () : pour afficher un message ou/et les variable.
5. Type de variable :
6. String : une chaîne de caractères.
7. Number : un nombre, il peut entier, riel, positif ou négatif.
8. Undefined : une valeur par défaut pour les variables non initialisées ou pour les propriétés manquantes.
9. Null : indique l'absence d'une valeur d'objet.
10. Symbol : Une donnée unique et non-modifiable.
11. Array : un tableau de valeurs.
12. Objet : une structure de données qui permet de stocker des collections de données sous forme de paires clé-valeur.
13. Type des opérateurs :
14. Arithmétique :
15. Addition : +
16. Soustraction : -
17. Division : /
18. Produit : \*
19. Reste : %
20. Logique :
21. && : et
22. || : ou
23. ! : non
24. Comparaison :
25. == : vérifier l’égalité
26. === : strictement égal, vérifier l’égalité par rapport leur type
27. < : inférieur strictement
28. > : supérieur strictement
29. =< : inférieur ou égal
30. >= : supérieur ou égal
31. != : différent
32. !== : strictement différent
33. Les conditions :
34. If :

Syntaxe :

If (condition){

// code à exécuter si la condition est vraie

}

1. If-else :

Syntaxe :

If (condition){

// code à exécuter si la condition est vraie

}else{

// code à exécuter si la condition est faut

}

1. If-else if-else :

Syntaxe :

If (condition1){

// code à exécuter si la condition1 est vraie

}….else if(condition\_n){

// code à exécuter si la condition\_n est vraie

}else{

// code à exécuter si la condition est faut

}

1. Switch-case :

Syntaxe :

Type varX = val ;

switch(varX){

case 1 : val1

// code à exécuter si l'expression est égale à val1

break ;

case n : valn

// code à exécuter si l'expression est égale à valn

break ;

default :

// code à exécuter si aucune des valeurs n'est correspondante

}

1. Les opérateurs ternaires :

Syntaxe :

Type message = (condition) ? ‘’message1’’+’’message2’’ ;

1. Les boucles :
   1. for :

Syntaxe :

for (initialisation; condition; incrémentation) {

// code à exécuter à chaque itération

}

* 1. while :

Syntaxe :

while (condition) {

// code à exécuter tant que la condition est vraie

}

* 1. do – while :

Syntaxe :

do {

// code à exécuter

} while (condition);

* 1. for in :

syntaxe :

for (let clé in objet) {

// code à exécuter pour chaque propriété de l'objet

}

* 1. for of :

syntaxe :

for (let index of array) {

// code à exécuter pour chaque élément de l'itérable

}

1. Les fonctions :

Syntaxe :

1. Fonctions et méthodes déjà définies :
2. charAt() : renvoie le caractère à un index spécifié dans une chaîne de caractères.

Syntaxe :

Var.charAt(index);

1. indexOf() : renvoie l'index de la première occurrence d'une valeur dans une chaîne.

Syntaxe :

Var. indexOf(chercher, start);

1. length() : donner le nombre caractères dans la chaîne.

Syntaxe :

varX.length ;

1. split() : supprimer un ou plusieurs caractères dans la chaîne.

Syntaxe :

Var. split(separator, limit);

1. trim() : supprime les espaces qui existe dans la chaîne.

Syntaxe :

varX.length ;

1. startwith() : vérifier si une chaîne commence par la sous-chaîne spécifiée.

Syntaxe :

varX.startwith(‘’sous\_chaîne’’, position) ;

1. endwith() : : vérifier si une chaîne trmine par la sous-chaîne spécifiée.

Syntaxe :

varX.endwith(‘’sous\_chaîne’’, position) ;

1. toUpperCase() : convertit tous les caractères d'une chaîne en majuscules.

Syntaxe :

varX.toUpperCase() ;

1. toLowerCase() : convertit tous les caractères d'une chaîne en miniscules.

Syntaxe :

varX.toLowerCase() ;

1. ReplaceAll() : remplacer toutes les occurrences d'un motif dans une chaîne par une nouvelle sous-chaîne.

Syntaxe :

varX.replaceAll(‘’occurrences’’, ‘’nouveau\_sous\_chaîne’’) ;

1. Includes() : vérifie si une chaîne contient une sous-chaîne spécifiée.

Syntaxe :

varX.includes(‘’sous\_chaîne’’, position) ;

1. Map() : crée un nouveau tableau avec les résultats de l'application d'une fonction donnée à chaque élément du tableau.

Syntaxe :

array.map(function(element, index, array) {

// transformation de chaque élément

});

1. Filter() : crée un nouveau tableau avec tous les éléments qui passent un test(qui sont filtrer).

Syntaxe :

array.filtrer(function(element, index, array) {

// condition de chaque élément

});

1. Raduce() : applique une fonction contre un accumulateur et chaque valeur d'un tableau (de gauche à droite) pour réduire le tableau à une seule valeur.

Syntaxe :

array.raduce(function(accumulator, currentValue, index, array) {

// condition de chaque élément

});

Ex : arr.reduce((acc, curr) => acc + curr, 0);

1. Sort() : trie les éléments d'un tableau.

Syntaxe :

array.sort(compareFunction);

1. Push() : ajoute un ou plusieurs éléments à la fin d'un tableau.

Syntaxe :

array.push(element1, element2, ...);

1. Unshift() : ajoute un ou plusieurs éléments au début d'un tableau.

Syntaxe : retire le premier élément d'un tableau et le renvoie

array.unshift(element1, element2, ...);

1. Sheft() : retire le premier élément d'un tableau et le renvoie.

Syntaxe :

array.shift() ;

1. Pope() : retire le dernière élément d'un tableau et le renvoie.

Syntaxe :

array.pope() ;