Les opérateurs Logique

et

Leur fonctionnement

I) qu'est ce qu'un opérateur logique :

Les opérateurs logiques sont des opérateurs qui vont principalement être utilisés avec des valeurs booléennes et au sein de conditions.

JavaScript support trois opérateur logique :

- L'opérateur logique « ET » écrit **&&**
- L'opérateur logique « OU » écrit **||**
- L'opérateur logique « NON » écrit **!**

II)Le Fonctionnement des différents opérateurs logique

```
1
2   let a = 1 , b = 2 , c = 3;
3
4   //opérateur logiques
5
6   //le "ET" -> && vrai si les 2 comparaison sont vrai uniquement
7   let test=a < b && b < c;
8   console.log(test)  true
9
10  test=a < b && b > c;
11  console.log(test)  false
12
13  //le "OU" -> || vrai si l'une des 2 comparaison est vrai (faux si les deux sont fausse)
14  test =a < b || b < c;
15  console.log(test)  true
16
17  test =a < b || b > c;
18  console.log(test)  true
19
20  //Négation -> "!"='non' si vrai devient faux et vice versa
21  !(a < b);
22  test = a < b;
23  !test;
24  console.log(!test)  false
```

1. Renvoi ***True*** car les ***a*** est bien inférieur à ***b*** **ET** ***b*** inferieur à ***c***
2. Renvoi ***False*** car ***a*** est bien inferieur à ***b*** **ET** ***b*** n'est pas supérieur à ***c***
3. Renvoi ***True*** dans les deux exemples car **l'un des deux cas et vrai,** ici **a** est inférieur à ***b***
4. Renvoi ***False*** car on demande si ***a*** **N'EST PAS** inférieur à ***b*** hors ***a*** est inférieur à ***b***

L'opérateur logique « ET » écrit && :

- Lorsqu'il est utilisé avec des valeurs booléennes, renvoie ***True*** si toutes les comparaisons sont évaluées à ***True*** ou sinon renvoi ***False*** ;

L'opérateur logique « OU » écrit || :

- Lorsqu'il est utilisé avec des valeurs booléennes, renvoie ***True*** si au moins l'une des comparaisons est évaluée à ***True*** ou sinon renvoi ***False***

L'opérateur logique « NON » écrit ! :

- Renvoie ***False*** si une comparaison est évaluée à ***True*** ou renvoie ***True*** dans le cas contraire

Il est tout à fait possible de créer des conditions qui utilise plusieurs opérateurs logiques comme :

```
27   if( a == 1 && b < c || c == 3){
28       test = "vrai";
29   }
30 1 console.log(test)  vrai
31
32   if (a <c || b> a || b< c){
33       test =" vraiee"
34   }
35 2 console.log(test)   vraiee
36
37
38   if( a < b && b != c && b > c){
39       test = "blop"
40   }
41 3 console.log(test);   vraiee
42
```

1. Ici la condition est vrai donc le code dans le **if** est exécuter.
2. Ici la condition est vrai aussi donc le code dans le **if** est exécuter.
3. Ici la condition est fausse donc le code dans le **if** n'est pas exécuter.

III) **Leur évaluation :**

Les operateur logique **&&**(ET) et **||**(OU) ont la particularité d'être *séquentiels*, c'est-à-dire de donner lieu à une évaluation de gauche à droite :

- *A* **&&** *B* est faux si *A* est faux, et vrai si *A* et *B* sont vrais : *B* n'est pas évalué si *A* est faux
- *A* **||** *B* est vrai si *A* est vrai, et faux si *A* et *B* sont faux : *B* n'est pas évalué si *A* est vrai

Dans une expression avec un ou plusieurs OU **||**, dès qu'une valeur est évaluée a vraie, elle est retournée, si toutes sont fausses, la dernière et retournée :

```
let firstName = "";
let lastName; //undefined
let nickName = null;
//0 et NaN sont également évalués à false
let result = firstName || lastName || nickName;
console.log(result);  null

result = firstName || lastName || nickName || "Anonymous";
console.log(result);  Anonymous

lastName = "Doe" ;
result = firstName || lastName || nickName || "Anonymous";
console.log(result);  Doe
```

Dans une expression avec un ou plusieurs ET **&&** dès qu'une valeur est évaluée a fausse, elle est retournée, si toutes sont vraies, la dernière et retournée :

```
let firstName = "John";
let lastName = "Doe";
let nickName = "JD";

let result = firstName && lastName && nickName && "Anonymous";
console.log(result);  Anonymous

lastName = undefined;
result = firstName && lastName && nickName && "Anonymous";
console.log(result);  undefined
```

IV) **Leur priorité :**

Quand plusieurs opérations ont lieu dans une expression, chaque partie est évaluée et résolue dans un ordre prédéterminé.

Cet ordre est connu sous le nom de priorité des opérateurs.

Des parenthèses peuvent être utilisées pour annuler l'ordre de priorité et forcer l'évaluation de certaines parties d'une expression avant d'autres.

Sans parenthèse l'ordre et le suivant :

- Le **!** « NON » à la plus grosse priorité il sera évalué en premier.
- Le **&&** « ET » et prioritaire sur le ou il sera évalué avant le **||** « OU ».
- Le **||** « OU » à la plus faible priorité il sera évalué en dernier.

```
test = a<b || b < c && c > a
//écriture équivalente
a < b ||( b < c && c > a)

test = a < b && b < c || a > b && b > c
//écriture équivalente
(a < b && b < c) || (a > b && b > c)
```