



# JavaScript - Les tests conditionnels

[Mise à jour le 20/6/2021]

- **Sources** et compléments sur **MDN Web Docs**
  - [if... else](#)
  - [L'opérateur conditionnel](#)
- **Lectures connexes**
  - Wikis WebPEM : ["Préparer un projet de site Web avec l'IDE VSCode"](#)

## Introduction

Les tests conditionnels permettent d'orienter l'exécution vers des branchements d'instructions selon le résultat de conditions définies par le développeur.

### 1. Le test simple



Le bloc d'instructions n'est exécuté que si la condition est vraie.

*Exemple* : calcul de frais de port à un seuil.

\*.js

```
if (montant >= 100) {  
    var port = 0;  
}
```

### 2. Le test si - sinon



Si la condition du if n'est pas vraie, les instructions du if sont ignorées, celles du else sont exécutées. Les test si sinon peuvent être imbriqués.

*Exemple 1* : calcul de frais de port à un seuil.

\*.js

```
var port = 0;
if (montant >= 100) {
    port = 0;
} else {
    port = 6.9;
}
```

Exemple 2 : calcul de frais de port à deux seuils.

\*.js

```
var port = 0;
if (montant >= 100) {
    port = 0;
} else if (montant >= 50){
    port = 3.9;
} else {
    port = 6.9;
}
```

### 3. Les tests multiples



Les tests multiples sont traités sur des conditions d'égalité.

Syntaxe

\*.js

```
switch (expression) {
    case valeur1 :
        blocInstructions1 // Ce bloc d'instructions est exécuté si
        expression est égal à valeur1
        break;

    ...

    case valeurN :
        blocInstructionsN
        break;
    default :
        blocInstructionsDefault
}
```

## Exemple

\*.js

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="UTF-8">
  <title>Jours semaine</title>
</head>

<body>
  <h1 id="message"></h1>
  <script type="text/javascript">
    var dt = new Date();
    var jour = dt.getDay();
    var msg = "";
    switch (jour) {
      case 1:
        msg = "C'est lundi";
        break;
      case 0:
      case 6:
        msg = "C'est le week-end";
        break;
      default:
        msg = "En semaine";
        break;
    }
    document.getElementById("message").innerHTML=msg;
  </script>
</body>

</html>
```

## 4. L'opérateur ternaire



L'opérateur ternaire doit son nom aux trois parties qui le composent.

Condition ? valeurSiVrai : valeurSiFaux

Exemple : calcul de frais de port à un seuil.

\*.js

```
var port= montant >= 100? 0 : 6.9;
```



Le gros avantage de l'opérateur ternaire est qu'il peut être utilisé au sein d'une autre instruction.

Exemple : ajout d'un s à la fin d'un mots en fonction de la valeur d'un nombre

\*.js

```
console.log("Votre devis comporte " + nb + " article" + (nb>1 ? "s" : ""));
```

## 5. Les conditions



Une expression de condition retourne une valeur booléenne.

Une variable peut recevoir le résultat d'une expression de test de condition.

Exemple

\*.js

```
var estMajeur = age >= 18;
```



Une condition est considérée comme vraie si elle renvoie toute valeur autre que 0.

Exemple

\*.js

```
if (nbArticle) {  
    msg = "Il y a " + nbArticles + "articles (s)dans le panier."  
}  
else {  
    msg = "Le panier est vide."  
}
```

### • Opérateurs d'égalité et de différence

- égalité : ==
- différence : !=

Exemples

\*.js

```
var nb = 5;
nb == 5; // true
nb == 6; // false
nb == "5"; // true (comportement dû au typage faible, est annulé si on utilise "===")
nb == "05.00"; // true (idem)
nb = 10 / 2; // true
```



Une erreur courante est d'utiliser l'opérateur d'affectation à la place de l'opérateur de comparaison. Cette erreur entraîne un double dysfonctionnement :

- Le test est toujours vrai, car une affectation réussie renvoie *true*.
- La variable à tester change de valeur.

### • Opérateur d'égalité (différence) et de contenu

- égalité : `===`
- différence: `!==`



Il existe un niveau supérieur d'égalité qui consiste à comparer, en plus des contenus, les **types** des deux parties du test d'égalité.

### Exemples

\*.js

```
var nb = 5;
nb == 5; // true
nb == "5"; // true
nb === "5"; // false
```

### • Autres opérateurs

- `>` : strictement supérieur à
- `<` : strictement inférieur à
- `>=` supérieur ou égal à
- `<=` inférieur ou égal à

### • Valeur *null*



La valeur **null** représente l'absence de valeur, c'est-à-dire le **vide**. Il ne faut pas confondre *null*, *false* et *undefined*.

### Exemples

\*.js

```
// Egalités faibles
console.log(null == false) // false
console.log(null == undefined) // true

// Egalités strictes
console.log(null === false) // false
console.log(null === undefined) // false

// Affectations et types
var nonNull = null, nonNullBis = null;
console.log(typeof nonNull) // object
console.log(nonNull === nonNullBis) // true
```

## 6. Combinaison de conditions

Les combinaisons de condition sont réalisées avec les opérateurs :

- **ET** matérialisé par **&&** (double ET commercial)
- **OU** matérialisé par **||** (double pipe)

Exemples

\*.js

```
if ((a == 1) && (b == 2)) { }
if ((a != 1) && (b == 2)) { }
if ((a == 1) || (b == 2)) { }
if ((a != 1) || (b == 2)) { }
if ((a == 0) || ((a == 1) && (b == 2))) { }
```

### • Inversion de sens

Une inversion de sens simplifie parfois la syntaxe et améliore la compréhension d'une condition.



En algèbre booléenne, la négation d'une expression consiste à inverser les ET et les OU, les égalités et les différences, et les supérieurs et les inférieurs.

Exemples

\*.js

```
// La condition ci-dessous :
if ((a != 1) || (b != 2)) { }
// peut être remplacée par
```

```
if ((a == 1) && (b == 2)) { }
```

From:

<https://webge.fr/dokuwiki/> - **WEBGE Wikis**

Permanent link:

<https://webge.fr/dokuwiki/doku.php?id=web:javascript:fondamentaux:conditions>

Last update: **2021/08/11 10:55**

