

[Welcome home !](#)[Règlement intérieur](#)[Poster une issue](#)[JavaScript](#)[Environnement de dev](#)[👋 Semaine 1](#)[Terminal](#)[Variables](#)[Conditions](#)[Boucles](#)[👋 Semaine 2](#)[Manipuler des chaînes de caractères](#)[Fonctions](#)[Tableaux](#)[👋 Semaine 3](#)[Déroulement d'un programme](#)[Les booléens](#)[Les valeurs truthy et falsy](#)[Notion de scope](#)[Conditions d'existence](#)[Conditions imbriquées](#)[👋 Semaine 4](#)[Objets](#)[Requêtes à API](#)[Backend & Strapi](#)[HTML / CSS](#)[Vue.js](#)[Après la formation](#)

Conditions "d'existence"

Tester si une valeur est `falsy`

Si d'aventure vous voulez vérifier qu'une valeur est `falsy`, voici comment procéder... Pour rappel, il existe 6 valeurs `falsy` en JavaScript : `null`, `undefined`, `NaN`, `0`, `false` et `""`.

Il faudrait donc se demander si notre valeur est identique à la première de ces valeurs `falsy`, ou bien à la deuxième, ou bien à la troisième, etc...

Vous pouvez donc utiliser `||` de la façon suivante :

```
const valueToTest = undefined;

if (valueToTest === null || valueToTest ==
    console.log("Notre variable contient une
  }
```

Un peu fastidieuse à mettre en place cette condition, non ? 🤔

Et bien il existe une "abréviation" à cette fastidieuse condition ! La voici :

```
const valueToTest = undefined;

if (!valueToTest) { // cette condition est
    console.log("Notre variable contient une
  }
```

🚩 Mettre un `!` devant la variable permet de valider la condition si cette variable n'est pas `truthy` (par conséquent `falsy`).

Tester si une valeur est `truthy`