4 minutes

Remontée des déclarations ou hoisting

Le `hoisting`

En `JavaScript` vous entendrez probablement parler du `hoisting` ou remontée des déclarations ou hissage en français.

Il s'agit d'une spécificité du langage qui fait que lorsque l'interpréteur `parse` (parcourt et analyse) votre code, il va automatiquement remonter toutes les déclarations en haut de votre code.

Cela permet d'écrire cela et de ne pas avoir d'erreur :

```
a = 2;
let a;
console.log(a); // 2
```

Copier

C'est totalement déconseillé ! Il est toujours recommandé de déclarer l'ensemble de ses variables en haut de son code pour plus de lisibilité.

Différence entre `var` et `let`

Pour bien comprendre la remontée il faut distinguer la **déclaration**, **l'initialisation** et l'**assignation**.

La **déclaration** est le fait que la variable soit rendue disponible dans sa portée (globale ou de bloc, nous y reviendrons). Lorsque le moteur `JavaScript parse` votre code il va remonter les déclarations avant d'exécuter le code durant la phase de compilation. (En réalité, elles ne sont pas remontées physiquement mais préparées en mémoire).

L'**initialisation** est le fait que le moteur `JavaScript` va assigner une valeur spécifique, `undefined`, aux variables. L'initialisation est une étape sur laquelle nous n'avons aucun contrôle

Il y a une différence importante entre `var` et `let`.

Pour les `var`, l'initialisation à `undefined` se fait pendant la phase de remontée immédiatement. Pour `let` et `const` ce n'est pas le cas. L'initialisation est effectuée lors de l'exécution.

L'**assignation** est le fait de donner une valeur explicitement à une variable avec = durant l'exécution.

Ainsi avec `var` :

```
console.log(b); // undefined car l’initialisation se fait lors de la remontée
var b = 1; // assignation
console.log(b); // 1
```

Copier

Pour `let` ou `const` :

```
console.log(a); // Uncaught ReferenceError: a is not defined
let a = 2;
```

Copier

La variable `a` est préparée en mémoire lors de la remontée mais elle n'est pas initialisée à `undefined`.

Les phases sont les suivantes :

```
// Phase de déclaration lors de la remontée, si on tente d'accéder à la variable let
// on a une ReferenceError
let a; // initialisation à undefined lors de l’exécution.
// ici a vaut undefined
a = 2; // assignation de la valeur 2
// ici a vaut 2
```

Copier

Le `hoisting` n'a maintenant plus de secret pour vous ! Ce n'est pas très important dans la mesure où avec l'abandon des `var` en `JavaScript` moderne vous n'aurez pas d'erreur. Cependant c'est une question qui peut revenir souvent lors de tests techniques.