Javascript : les erreurs

Contenu

Introduction

Le langage Javascript paraît souvent rébarbatif aux débutants car il est très réceptif aux erreurs (sensibilité à la casse, non conversions de types etc.) lesquelles peuvent bloquer le chargement complet d'une page web.

Pour ne pas aider, il y a encore 4 ou 5 ans, les navigateurs étaient très loin d'être homogènes (surtout Internet Explorer) dans l'interprétation du Javascript (tout comme dans celle du HTML et CSS), les résultats étaient donc différents d'un navigateur à l'autre.

Ces problèmes de compatibilité ont nettement diminué avec l'arrivée de la dernière version de Javascript (ES6) et des nouvelles générations de navigateurs (Edge notamment) mais il subsiste cependant quelques subtilités d'un navigateur à l'autre.

Maîtriser les erreurs fréquentes peut être d'un grand secours et heureusement, les navigateurs embarquent désormais nativement des outils de débogage et d'inspection du code Javascript (mais aussi de tout ce qui compose une page web : HTML, CSS, cookies, requêtes HTTP, médias etc.).

Les types d'erreurs

Il y a 3 grandes catégories d'erreurs dans l'utilisation d'un programme JavaScript :

Les erreurs au chargement

Tout d'abord, il faut bien entendu en premier lieur charger le bon fichier Javascript : il peut y avoir une erreur dans le nom ou le chemin du fichier (attribut `src` de la balise HTML `<script>` dans votre HTML).

Au chargement du script par le navigateur, JavaScript passe en revue les différentes erreurs qui peuvent empêcher le bon déroulement de celui-ci.

S'il décèle une erreur de syntaxe, c'est **tout le chargement du bloc qui est annulé**.

Les erreurs à l'exécution

Si le chargement s'est bien passé, l'exécution peut aboutir à un message d'erreur lors de l'exécution d'une partie du code JavaScript. Les erreurs à l'exécution proviennent d'un mauvais usage des commandes ou des objets JavaScript. Un exemple d'erreur à l'exécution est un appel erroné à une variable ou une fonction inexistante (car il y a, par exemple, une erreur dans le nom de la variable ou de la fonction).

Les erreurs de logique

Ce sont les plus vicieuses car le débogueur de JavaScript ne signale bien entendu aucune erreur et votre script se déroule correctement. Hélas, à l'arrivée, le résultat ne correspond pas à celui espéré. Il n'y a plus qu'à revoir la construction logique de votre script.

Comment bien déboguer

> **Certains de ces conseils peuvent s'appliquer à d'autres langages !**

Identifier l'erreur

Tout d'abord, il convient de parvenir à identifier l'emplacement de l'erreur et son type. Pour cela, utilisez la console des outils de développement de votre navigateur (touche F12, voir chapitre 11 Les outils de développement) qui indique le message d'erreur ainsi que le nom du fichier JS et la ligne où se situe l'erreur.

Ensuite, il faut parvenir à interpréter le message d'erreur, pas toujours explicite :

- Les erreurs courantes en Javascript

Vérifier le code

- Vérifier la syntaxe, en vous aidant notamment de la coloration syntaxique de l'éditeur de code que vous utilisez : points-virgules à la fin des lignes, fermetures des instructions et blocs (accolades, parenthèses).
- Vérifier la casse des noms de variables et de fonctions puisque le Javascript y est sensible
- Dans les conditions, n'oubliez pas que l'opérateur de comparaison d'égalité est spécifiée par « == » et non pas un seul (c'est une erreur fréquente)
- Afficher la valeur et le type des variables (voir les fonctions utiles ci-après) : permettra notamment de voir si elle vaut `null` ou `undefined`, si on a une chaîne au lieu d'un entier (oubli de `parseInt()` lors d'un prompt par exemple)
- Mettre en commentaires des lignes ou blocs de code pour les désactiver temporairement

Fonctions utiles

- `alert(variable)` ou `console.log(variable)` : affiche la valeur de la variable passée en argument
- `typeof variable` : retourne le type d'une variable; peut être affiché (via `alert` ou `console.log()`) ou utilisé dans une condition.

Aller plus loin : gestion des erreurs (facultatif)

Gérer les erreurs en testant les données attendues, en affichant des messages à l'utilisateur, en utilisant les exceptions. Exemples ici.