DÉBOGUER VOTRE CODE (PARTIE 1/3)

**Le débogage**

Comme son nom l'indique, cette technique consiste à supprimer les bugs qui existent dans votre code. Pour chaque type de bug vous avez plusieurs solutions bien particulières.

Les bugs syntaxiques sont les plus simples à résoudre, car l'interpréteur JavaScript vous signalera généralement l'endroit où l'erreur est apparue, cette signalisation peut être consultée dans la console de votre navigateur. Vous verrez comment vous servir de la console un peu plus loin.

En ce qui concerne les bugs algorithmiques, là il va falloir faire travailler votre cerveau et chercher par vous-mêmes où vous avez bien pu vous tromper. La méthode la plus simple consiste à remonter les couches de votre code pour trouver à quel endroit s'est produite l'erreur.

Par exemple, si vous avez un calcul qui affiche une mauvaise valeur, vous allez immédiatement vérifier ce calcul. Si ce calcul n'est pas en cause mais qu'il fait appel à des variables, alors vous allez vérifier la valeur de chacune de ces variables, etc. De fil en aiguille vous allez parvenir à déboguer votre code.

Pour vérifier les valeurs de vos variables, calculs, etc., la fonction alert() vous viendra probablement à l'esprit, mais sachez qu'il existe une méthode spécialement adaptée à ce cas de figure, il s'agit de console.log(), nous en parlerons en détails dans ce chapitre.

**En résumé**

* Il existe deux types de bugs : les bugs syntaxiques et les bugs algorithmiques. Le dernier est souvent le plus compliqué à résoudre.
* Tout navigateur récent et un tant soit peu populaire possède un kit de développement permettant de faciliter le débogage de votre code.
* La console permet d'afficher les erreurs syntaxiques de votre code mais vous permet aussi d'y afficher du texte ou le contenu d'une variable à l'aide de la méthode console.log().
* Il est possible d'exécuter du code dans la console et d'accéder aux variables globales de la page actuelle.
* Les points d'arrêt permettent de mettre en pause votre script à un point précis de votre code et d'analyser l'état du scope courant ainsi que de l'espace global.
* La pile d'exécution définit par quelles fonctions votre code est passé afin d'atteindre la ligne actuellement mise en pause par un point d'arrêt.