openclassrooms.com

Adoptez la logique de débogage

9–11 minutes

Félicitations, vous connaissez à présent les outils de débogage. Vous allez pouvoir enquêter sur vos bugs comme un enquêteur le fait sur une scène de crime !

Què ? Je crois que l'auteur du cours est devenu fou...

Mais non, rassurez-vous, je ne suis pas devenu fou (du moins, pas encore). Par contre, je trouve la comparaison intéressante ! En effet, résoudre un bug, c'est un peu comme résoudre une scène de crime, mais en moins grave bien sûr.


Pour résoudre une scène de crime, les enquêteurs utilisent souvent une **méthodologie** pour les aider dans leurs investigations. C'est justement ce que vous allez faire dans ce cours : vous utiliserez une méthodologie pour résoudre vos bugs. Cette dernière vous permettra de bien décomposer la résolution de votre bug.

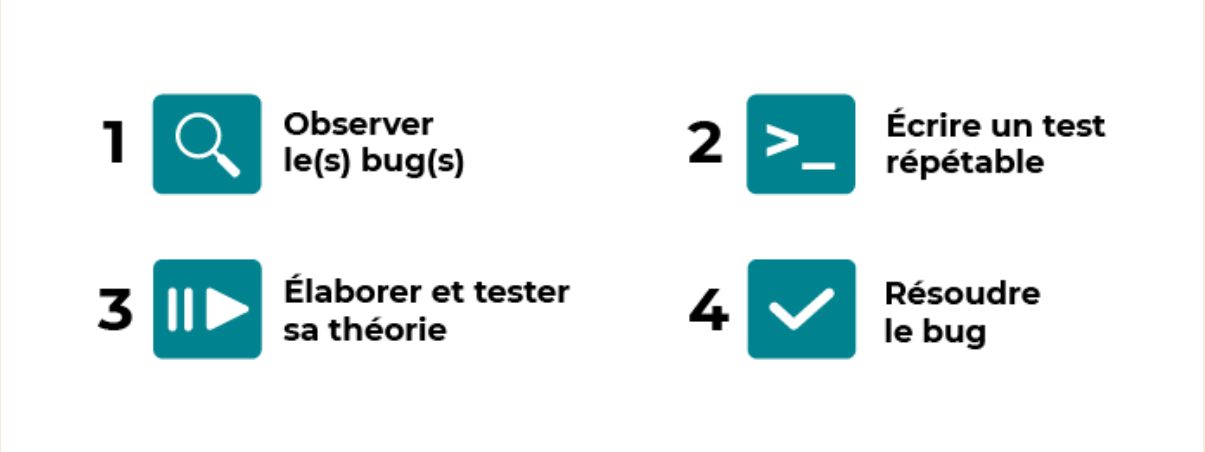
Allez, c'est parti mon cher Watson !

Découvrez la méthodologie de débogage

Pour résoudre un bug, comme pour résoudre une scène de crime, il est important de **procéder par étape**.

Chaque étape va vous permettre de voir les questions à vous poser ainsi que l'attitude à adopter. **Une fois l'étape assimilée, vous pourrez l'expérimenter sur notre projet fil rouge.**

Les étapes de la méthodologie sont les suivantes :



Les 4 étapes de la méthodologie de débogage

Dans ce type de cas, il va vous falloir **répéter cette méthodologie jusqu'à résolution complète** du ou des bugs. C'est quelque chose qui se produit très régulièrement quand on code.

Observez le bug

> Watson, un crime terrible a eu lieu ! Le Colonel Moutarde a été assassiné !

Désolé d'avoir dû tuer le Colonel Moutarde pour notre exemple, mais sachez que sa mort ne sera pas vaine !

Quand on enquête sur un bug, **il est important de ne pas plonger directement dans le code**. Vous devez essayer de **prendre du recul**, d'observer la situation, et donc le bug, et de vous poser des questions !

Essayez, par exemple, de vous demander :

- Où se trouve le bug ? S'agit-il d'un bug d'affichage et donc potentiellement d'un bug côté CSS ou d'un bug côté JavaScript ?
- Y a-t-il une erreur dans la console ? La comprenez-vous ? À quelle ligne de votre code cette erreur se produit-elle ?
- Si le bug se trouve côté JavaScript, quel type de bug est-ce à

première vue ? S'agit-il d'un bug plutôt lié à une API ou plutôt lié au DOM ? Ou peut-être les deux ?

- Si le bug se trouve côté CSS, apparaît-il sur un appareil en particulier, par exemple sur un iPhone et pas sur un Android ?
- Quel parcours précis je fais pour générer le bug ? Quel support j'utilise ?

  Pour en revenir à notre comparaison avec l'assassinat du Colonel Moutarde, observer un bug, c'est un peu comme observer une scène de crime.

  > En effet, sur une scène de crime, l'enquêteur se pose des questions et observe, par exemple :
  >
  > - dans quel sens se tient le cadavre ? Tient-il quelque chose à la main ?
  > - s'il y a des éclats de verre sur le sol ? Peut-être que le coupable est entré dans la pièce par la porte ? Ou par la fenêtre ?
  > - l'arme du crime. Est-elle visible ? Est-ce qu'elle correspond à la blessure ?
  > - Qui sont les témoins ? Où se trouvaient-ils au moment du meurtre ?

  À cette étape, il est important d'aller du **macroscopique vers le microscopique** : regardez et observez le bug dans son ensemble puis essayez de zoomer sur certaines parties de votre code pour voir si ces dernières sont en relation avec ce dernier.

  Dans la vidéo ci-dessous, nous allons nous intéresser à un bug présent sur la page d'accueil de notre projet Façadia. Voici le code source correspondant à cette vidéo. Il semblerait que le bouton se connecter ne fonctionne plus.

  Vous en savez désormais un peu plus sur le bug, il est temps d'apprendre à créer un **test répétable**.

  Écrivez un test répétable

Écrire un test répétable est souvent la **partie la plus complexe** car reproduire un bug n'est pas toujours chose aisée. En effet, le bug peut se produire dans des **circonstances particulières**, sur l'ordinateur d'un certain type d'utilisateur mais pas sur le vôtre.

Votre première mission pour cette étape va donc être de reproduire le bug sur votre ordinateur et dans votre environnement de développement. Être en mesure de le répéter va vous permettre de comprendre son contexte d'exécution.

Si le bug se produit sur votre ordinateur, essayez de noter le parcours d'exécution de ce dernier : quelles sont les étapes qui amènent à ce bug et quel est le parcours utilisateur ?

Si le bug se produit sur l'ordinateur d'un collègue, ou pire, d'un utilisateur, il est important que vous ayez les informations suivantes :

- Quel type d'appareil votre utilisateur a-t-il ? Utilise-t-il un ordinateur portable, une tablette, un téléphone ?
- Quel système d'exploitation (Windows, Mac, Linux, iOS, Android) votre utilisateur a-t-il au moment où survient le bug ? De quelle version dispose-t-il ?
- Quel navigateur est utilisé ? Quelle est sa version ?

Pour finir, vous devrez parfois écrire du code pour répliquer les conditions du bug. Cela vous permettra de tester automatiquement la fonctionnalité buggée et de travailler dessus.

Dans notre comparaison avec la scène de crime, cela pourrait se rapprocher du moment où Sherlock est dans son labo et analyse les pièces à sa disposition :

- il va regrouper les témoignages ;
- il va essayer de collecter des indices sur les pièces à conviction ;
- et il va reproduire dans son laboratoire la scène de crime !

Grâce à la vidéo précédente, vous avez vu comment et où se produisait le bug. Nous allons donc maintenant essayer de

reproduire ce dernier et tester des cas de figure. Essayez de tester de votre côté avant de regarder la vidéo

.

Élaborez une théorie concernant le bug et testez-la

> C’est maintenant l’heure d’émettre des hypothèses quant à l’assassinat du Colonel Moutarde !

Si vous résumez les étapes précédentes, vous pouvez voir qu’on commence à avoir une idée assez précise du bug. Vous savez notamment :

- d’où il peut venir ; du CSS ou du JavaScript par exemple ;
- comment le reproduire ;
- à quel moment ce dernier survient.

Toutes ces informations vont vous permettre d’**élaborer une théorie** et de la **tester**. Cela dit, attention, tester une théorie, et plus particulièrement, si cette dernière fonctionne **ne veut pas dire que le bug est résolu** : cela veut surtout dire que vous avez compris d’où il venait.

Si, par contre, votre théorie échoue, pas de panique, **ce n’est pas grave** : ces choses-là arrivent constamment ! Cela veut peut-être dire que vous êtes passé à côté de quelque chose d’important. Essayez alors de regarder les indices un à un et de comprendre ce qui n’a pas marché. N’hésitez surtout pas à demander à un collègue, il pourra aussi vous apporter son éclairage.

Dans la vidéo ci-dessous, nous allons repartir de notre bug du projet fil rouge et essayer de tester la théorie. Avant de vous lancer dans la vidéo, essayez de vous servir d’un des outils que nous avons vus ensemble dans le chapitre précédent pour tester votre théorie, cela pourra peut-être vous aider.

Comme vous pouvez le voir, il est important de se servir des outils à votre disposition. Ici, grâce au débogueur, j’ai pu séquencer

l'exécution de mon code.

À vous de jouer !

La résolution du bug va souvent être la seule étape où vous allez devoir **écrire du code**. Une fois le code écrit, il est très important :

- d'essayer de reproduire à nouveau le bug pour voir si ce dernier survient. Essayez de changer de navigateur, de taille d'écran, etc.
- de tester les autres fonctionnalités de votre site et plus particulièrement celles qui auraient pu être impactées par votre solution.

J'insiste particulièrement sur le deuxième point : il est très fréquent que les développeurs manquent de rigueur à cette étape. Le plus souvent, ils voient que leur code fonctionne, que le bug est résolu, mais ils ne prennent pas le temps d'aller regarder si autre chose a cassé.

Dans la vidéo ci-dessous, vous allez voir comment le bug de notre projet Façadia est résolu. Avant de vous lancer dans la vidéo, essayez de résoudre le bug de votre côté et de tester tous les cas de figure.

Comme je vous le disais, cette étape vous permet d'implémenter la solution mais pas que ! Il est important que vous testiez votre code et les différents cas de figure pour voir si tout fonctionne comme prévu.

En résumé

- La première étape lors de la résolution d'un bug est son observation : essayez de comprendre où le bug survient, côté CSS ? Côté JavaScript ? Etc.
- La deuxième étape est d'écrire un test répétable. Autrement dit, vous allez devoir créer les conditions pour pouvoir reproduire ce bug autant que possible.
- Ensuite, vous pouvez élaborer et tester votre théorie. C'est le moment de voir ce qui marche et ce qui ne marche pas.
- Enfin, vous pouvez rédiger votre code en proposant une solution au bug.

  Vous connaissez maintenant les principaux outils du débogage ainsi qu'une méthodologie à mettre en pratique. Il est maintenant temps de pratiquer !

  Testez vos premières compétences dans le quiz qui suit, et retrouvez-moi dans la partie 2, où nous apprendrons concrètement à déboguer du code HTML/CSS.

- #

  Prenez en main les outils de débogage Quiz : Identifiez les différents types de bug de votre interface Front-End