openclassrooms.com

# Servez-vous des outils dédiés au débogage

6–8 minutes

---

Maintenant que vous avez découvert tous les types de bugs côté DOM, il est temps d'aller plus loin avec la **console** et le **débogueur**. Dans ce chapitre, vous allez apprendre à vous servir de ces outils pour déboguer votre code.

## Utilisez la console pour déboguer votre DOM

Le premier outil que je vais vous présenter est la console. Normalement, vous connaissez déjà cet outil : nous nous en sommes déjà servi pour déboguer dans les chapitres précédents. Cela dit, dans ce chapitre, nous allons aller plus loin et découvrir de nombreuses fonctionnalités de la console.

### Initialisez et accédez à des variables dans votre console.

L'une des principales fonctionnalités est de pouvoir créer et récupérer des variables. Cela dit, vous pouvez le faire uniquement si cette variable est accessible globalement, autrement dit si elle est déclarée en dehors d'une fonction.

Mais, on parle-t-on de variables stockées dans le code source du

projet ou on peut le faire dans la console ?

Bonne question !

En fait, la console nous permet de faire les deux :

- il est tout à fait possible d'initialiser de nouvelles variables dans la console.
- il est aussi possible de récupérer une variable déjà existante et qui a été déclarée dans le code.

Dans l'exemple ci-dessous, j'utilise ma console pour créer une constante firstName qui contiendra mon prénom. Je peux ensuite directement l'afficher.

```
>> const firstName = "Thomas"
<- undefined
>> firstName
<- "Thomas"
```

Extrait de la console

L'un des principaux intérêts de cette fonctionnalité est de pouvoir :

- tester rapidement un bout de code.
- connaître l'état d'une variable durant l'exécution d'un programme.

**Analysez des erreurs JavaScript**

Comme vous avez pu le voir dans les chapitres précédents, je me suis déjà servi plusieurs fois de la console pour avoir des informations comme par exemple un message d'erreur :


Ici, nous sommes face à une erreur de Reference. Autrement dit, la variable appelée n'existe pas.

C'est une méthode bien pratique pour comprendre où une erreur se produit et pourquoi. Ici, l'importance va être de bien comprendre les

messages d'erreur et à quelle ligne ils surviennent.

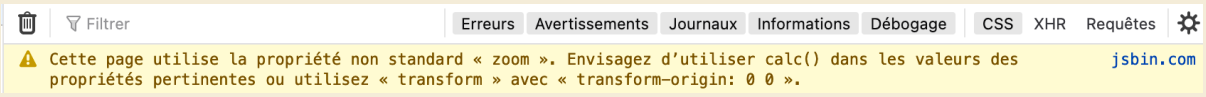
Mais faut-il vraiment connaître tous les messages d'erreur ?

C'est une question d'habitude.

Plus vous allez être confronté à une erreur, plus vous serez capable de comprendre d'où elle provient.

### Affichez et filtrez des informations dans votre console

En plus d'initialiser des variables, vous pouvez aussi vous servir de la console pour afficher des informations (via le célèbre console.log et ses déclinaisons) ainsi que pour filtrer les différents niveaux d'information.




Ici j'ai activé la possibilité de recevoir des mises en garde côté CSS.

Il peut être intéressant de filtrer vos informations si vous souhaitez afficher uniquement des informations relatives aux erreurs sur votre page ou aux requêtes HTTP par exemple.

Je me sers très souvent du console.log pour déboguer mon code : j'affiche chaque variable qui me semble poser problème et je me pose la question du type de variable que je manipule. Il me permet aussi de voir si le code passe aux endroits requis pour exécuter le code.


### Écrivez puis exécutez des fonctions

La console peut aussi vous permettre d'écrire et d'exécuter des fonctions. Cela peut être utile si vous souhaitez tester un bout de code rapidement et voir si ce dernier fonctionne.

```
const helloWorld = () => console.log('Hello, world')
undefined
helloWorld()
```

```
Hello, world
undefined
```

Extrait de la console

**À vous de jouer !**

Vous vous souvenez de notre bug du projet fil rouge, Façadia, dans le chapitre précédent ? Celui sur le bouton “Voir les détails” ? Nous allons utiliser le débogueur pour le résoudre.


En plus du console.log classique, vous pouvez aussi parfois passer le console.group et le console.groupEnd. Cela va vous permettre d’afficher des blocks de message ensemble. Voici ci-dessous un exemple d’utilisation :

```
console.group()

console.log("Une variable")

console.log("Une autre variable")

console.groupEnd() // J'ai besoin d'ajouter le groupEnd ici pour faciliter la lecture des éléments
```

**Tirez parti du débogueur**

En plus de la console, nous avons vu ensemble au début de ce cours qu’il y avait un autre outil à votre disposition : le **débogueur**.

**Prenez en main le débogueur (sans l’extension VsCode)**

Nous allons ici nous concentrer sur cette issue : https://github.com/tdimnet/debuggez-l-interface-de-votre-site/issues/2. Vous pouvez vous mettre sur la branche `partie -3/chapitre-1/section-5` .

Avant de regarder cette vidéo, essayez de prendre en main le débogueur de votre côté et réfléchissez à comment vous pourriez l'utiliser pour déboguer votre page.

Comme vous pouvez le voir, le débogueur permet d'aller un peu plus vite que les console.log. C'est un outil vraiment puissant et qui est souvent assez peu utilisé.

**Prenez en main le débogueur (avec l'extension VsCode)**

Petit bonus avant de passer à la prochaine partie : vous allez découvrir comment déboguer avec l'extension VsCode que je vous ai présentée en début de chapitre.

Après cette vidéo, vous n'aurez plus aucune excuse pour mal vous servir du débogueur !

Pour cette vidéo, vous aurez besoin de vous mettre sur la branche `partie-3/chapitre-1/section-5-extension`.

**À vous de jouer !**

Vous connaissez et savez maintenant utiliser la console et le débogueur pour résoudre vos bugs. Nous allons donc en profiter pour travailler sur un exercice pratique.  Voici le bug sur lequel nous allons travailler : https://github.com/tdimnet/debuggez-l-interface-de-votre-site/issues/3

Comme toujours avant de regarder la vidéo ci-dessous, essayez de

le trouver et de le résoudre de votre côté. Tirez parti de la console et du débogueur pour le résoudre. Cela va vous permettre de continuer à bien prendre en main ces deux outils.


**En résumé**

- La console est un outil très pratique côté front-end. Vous pouvez déclarer des variables, exécuter du code JavaScript et appeler des fonctions.
- Quand on débogue via la console, on va souvent ajouter des console.log aux endroits possibles du bug. Cela nous permettra de remonter le fil pour découvrir l'origine de ce dernier.
- Le débogueur est un outil complémentaire de la console. Vous pouvez l'utiliser pour connaître l'état des variables.

  *Vous connaissez désormais les sources de bug possibles côté DOM et les outils à votre disposition pour les traquer et les résoudre, il est temps de nous intéresser aux bugs d'API.*