



Décoder

1xx - Débogage en JavaScript (navigateur)

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision

Sommaire

A Introduction.....	1
A.1 Présentation.....	1
A.2 Prérequis.....	1
B Utiliser les outils de débogage du navigateur.....	2
B.1 Activer le débogueur.....	2
B.1.1 Erreurs classiques JavaScript.....	3
B.1.1.a Variable ou fonction non définie.....	3
B.1.1.b Undefined, NaN, Infinity.....	4
B.1.1.c Erreur de type.....	4
C Utiliser les points d'arrêt.....	5
C.1.1 Gestion point d'arrêt.....	6
C.1.1.a Via le débogueur.....	6
C.1.1.b Via le code.....	8
C.1.1.c Expressions espionnes.....	8
D Gérer les performances.....	9
D.1 Accès à un site.....	9
D.2 Accès à une API.....	10
E Envoyer des messages dans la console.....	11
E.1 Messages classiques.....	11
E.2 Les autres messages.....	12
E.2.1 Afficher un tableau.....	12
E.2.2 Afficher une durée.....	12
E.2.3 Afficher les fonctions.....	13
E.2.4 Autres fonctions.....	13
F Annexes.....	14
F.1 Sources.....	14

Nomenclature à supprimer :

- **Assimilation** : cours pur. Explication théorique et détaillée (globalement supérieur à 4 pages).
- **Décoder** : fiche de cours, généralement inférieure à 5 pages.
- **Découverte** : Travaux dirigés. Faisable sans matériel.
- **Explorer** : Travaux pratiques. Nécessite du matériel ou des logiciels.
- **Mission** : Projet encadré ou partie d'un projet.
- **Projet** : Projet en autonomie totale. Environnement ouvert.

A Introduction

Une des plus grandes difficultés dans un code, est son débogage : trouver une erreur peut parfois tourner au cauchemar.

C'est d'ailleurs celui des développeurs débutants qui souvent ne savent pas où trouver les outils pour déboguer leurs propres codes.

Témoignages :

Jérémy : "ça ne marche pas !"

Lucie : "y'a rien qui se passe, pourtant j'ai tapé la même chose que vous !"

Bénédicte : "C'est où qu'on trouve pourquoi ça ne marche pas ?"

Wahil : "Mais j'hallucine, pourquoi y'a rien qui s'affiche ???"

A.1 Présentation

Ce décodage donne un rappel rapide des éléments à mettre en œuvre pour pouvoir se dépanner seul.

A.2 Prérequis

Cours JavaScript

B Utiliser les outils de débogage du navigateur

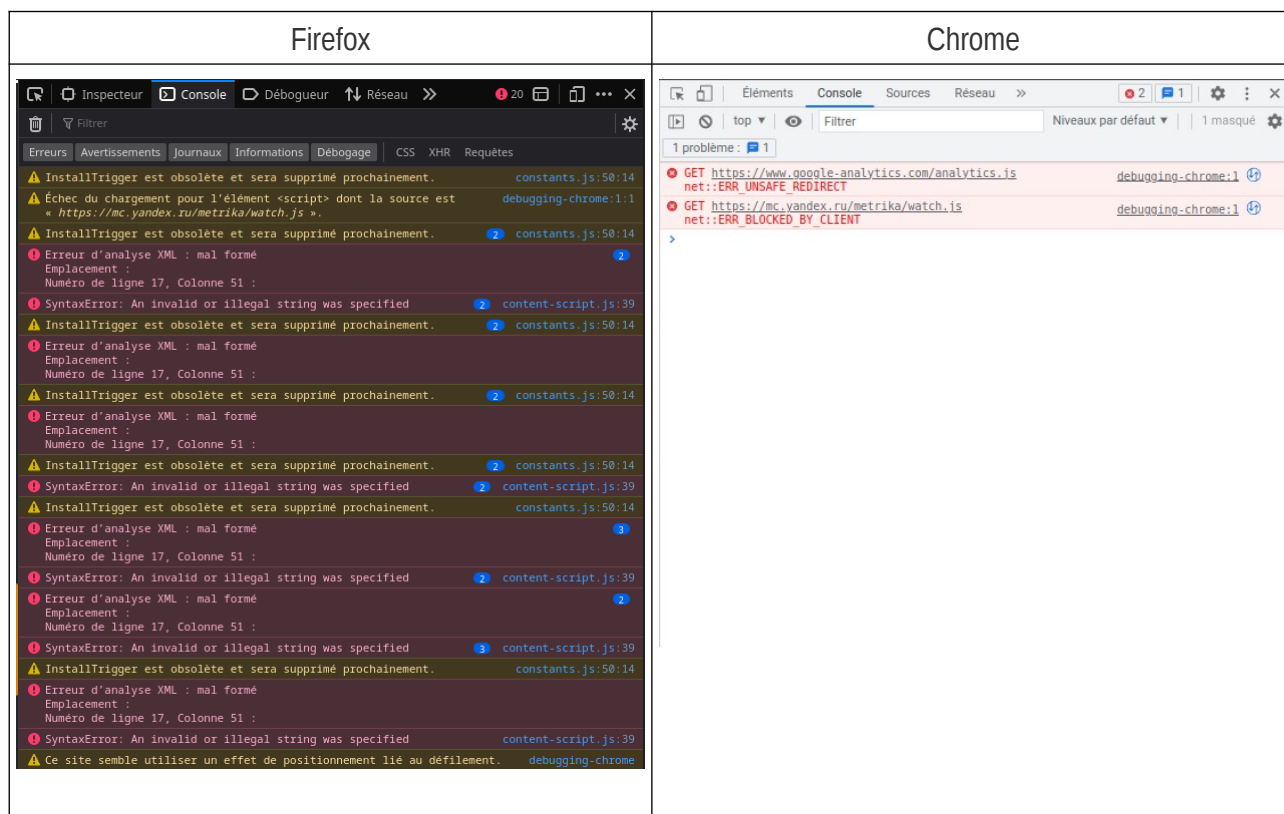
Tous les navigateurs intègrent des outils pour analyser le fonctionnement des requêtes HTTP et l'exécution des scripts JavaScript. C'est ce dernier outil qui nous intéresse.

B.1 Activer le débogueur

Dans la plupart des navigateurs, la combinaison de touches est [CTRL]+[shift]+[i], mais l'accès est possible via la souris également :

- Clic droit sur la page et choisir
 - Firefox : `inspecter`
 - Chrome : `outils de développement` → `Inspecter`
- Menu :
 - Firefox : `outils supplémentaires` → `outils de développement`
 - Chrome : `Menu outils` → `outils de développement`

Dans tous les cas, vous devriez obtenir l'affichage d'un panneau contenant au minimum un onglet [Console].



Les erreurs peuvent donc s'afficher dans cette partie du navigateur : il faut toujours ouvrir ce débogueur pour trouver le message d'erreur et déterminer où intervenir dans le code.

Les couleurs des messages indiquent leur gravité :

information	<pre>Cuisine 26.26 les_fonctions_gr2.html:50 Salon 29.87104912367834 les_fonctions_gr2.html:50 Chambre 71.87°F les_fonctions_gr2.html:50 SdB -3.19°C les_fonctions_gr2.html:50 Buanderie 22.76°C les_fonctions_gr2.html:50</pre>
avertissement	<pre>⚠ Ce site semble utiliser un effet de positionnement lié au défilement. debugging-chrome Cet effet pourrait ne pas fonctionner correctement avec le défilement asynchrone. Consultez https://firefox-source-docs.mozilla.org /performance/scroll-linked_effects.html pour obtenir davantage de détails ou discuter des outils et des fonctionnalités liés.</pre>
blocage	<pre>✖ ▶ Uncaught ReferenceError: toto is not defined les_fonctions_gr2.html:52 at les_fonctions_gr2.html:52:21</pre>

B.1.1 Erreurs classiques JavaScript

B.1.1.a Variable ou fonction non définie

```
Uncaught ReferenceError : xxxxxx is not defined at nomsript:ligne:colonne
```

```
❗ ▶ Uncaught ReferenceError: toto is not defined les_fonctions_gr2.html:52:9
<anonymous> ...Spaces/Javascript/les_fonctions_gr2.html:52
[En savoir plus]
```

Le débogueur signale simplement que la variable ou la fonction appelée/utilisée n'existe pas :

- Vérifier que l'élément existe déjà avant
- Vérifier que les caractères sont les mêmes (majuscules et minuscules)
- Vérifier que la déclaration correspond à la portée de l'élément (déclaration de variable dans un bloc avec let ou const)

Notez que le débogueur donne le **numéro de la ligne** après le nom du fichier (à droite).

B.1.1.b Undefined, NaN, Infinity...

```
undefined                               | les_fonctions_gr2.html:52:17
```

Bien que non-bloquant, le débogueur affiche un message inattendu pour une des raisons suivantes :

- `undefined` : la variable existe mais ne contient rien du tout
- `NaN` : le calcul ne peut être fait, car ce n'est pas un nombre (Not a Number)
- `Infinity` : le résultat est infini, car il s'agit d'une division par zéro

B.1.1.c Erreur de type

```
Uncaught TypeError: tes.toFixed is not a function
```

On applique une fonction sur un mauvais type de données :

- Vérifier que la variable contient le bon type de données (par exemple ici, un nombre).

C Utiliser les points d'arrêt

Le point d'arrêt permet de suspendre le programme à un endroit et ainsi, lire les valeurs des variables à ce moment-là. C'est très utile dans les fonctions utilisant des boucles.

Pour utiliser cette fonction nous avons besoin d'un petit code. Créez un fichier HTML contenant ceci :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Les fonctions</title>
</head>
<body>
  <script>

    function Capteur(isCelsius=true, isText=true, isDecimal=false) {
      let temp = Math.random()*40-10 //calcul en °C
      let degre = "C"
      if (isCelsius == false) {
        temp = temp * 1.8 + 32
        degre = "F"
      }
      if (isDecimal == false) {
        temp = temp.toFixed(2)*1
      }
      if (isText == true) {
        return temp+"°"+degre
      } else {
        return temp
      }
    }

    // reçoit une chaîne de caractères et retourne un nombre en degré Celsius
    function DetecterDegre(valeur) {
      let nouvelleValeur
      if (valeur.charAt(valeur.length-1)=="F") {
        nouvelleValeur = valeur.substring(0, valeur.length-2)
        nouvelleValeur = (nouvelleValeur-32)/1.8
      } else {
        nouvelleValeur = valeur.substring(0, valeur.length-2)*1
      }
      return nouvelleValeur
    }

    let pieces = ["Cuisine", "Salon", "Chambre", "SdB", "Buanderie"]
    let temperatures = []
    temperatures[0] = Capteur(true, false, false)
    temperatures[1] = DetecterDegre(Capteur(false, true, true))
    temperatures[2] = Capteur(false, true, false)
    temperatures[3] = Capteur(true, true)
    temperatures[4] = Capteur()

    for (let t=0; t < pieces.length; t++) {
      document.write(temperatures[t]+" dans "+pieces[t]+"<br>")
      console.log(pieces[t], temperatures[t])
    }

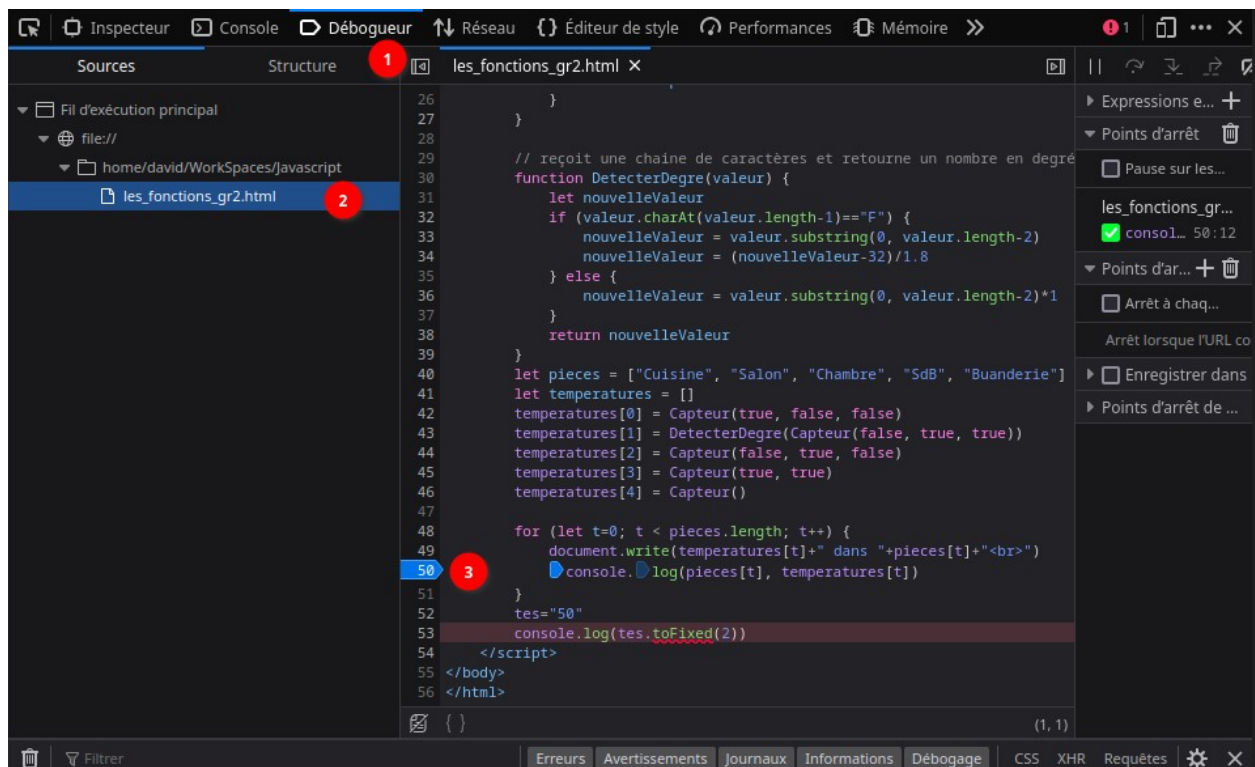
  </script>
</body>
</html>
```

C.1.1 Gestion point d'arrêt

C.1.1.a Via le débogueur

L'accès au point d'arrêt (breakpoint) se fait dans le débogueur :

1. Cliquer sur l'onglet [Débogueur]
2. Dans la zone de gauche, il faut sélectionner le fichier JavaScript (ou le fichier HTML contenant le code JavaScript).
3. Cliquer sur la ligne choisie pour stopper l'exécution du code.



Le point d'arrêt est visible, cela signifie que la boucle sera interrompue par le débogueur. En rafraîchissant la page, celle-ci exécute le code rencontré, s'arrête à l'endroit du point d'arrêt et un bouton permet de continuer le programme (un peu comme une Pause / Reprendre sur une vidéo).

-8.43 dans Cuisine

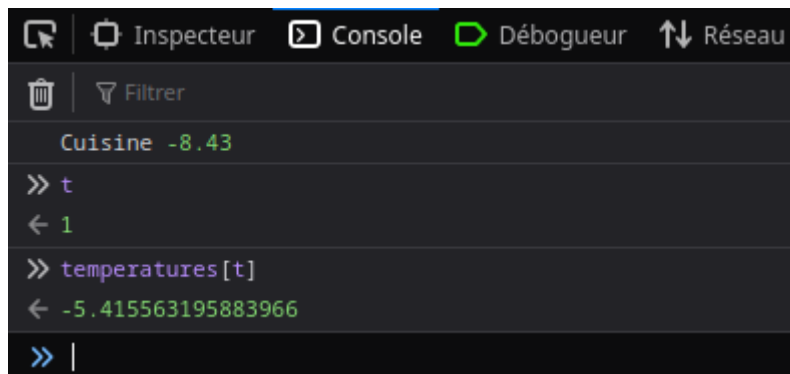
Mis en pause par un point d'arrêt



Le premier symbole force l'exécution en pas à pas (faire une pause à chaque instruction).

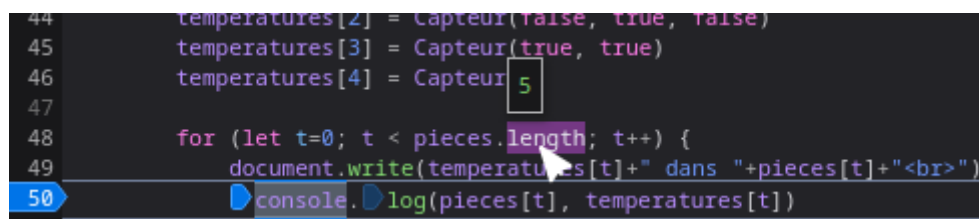
Le second symbole reprend l'exécution jusqu'à la rencontre d'un prochain point d'arrêt.

En allant dans l'onglet [Console] il devient possible d'afficher les variables au moment du point d'arrêt :



On peut ainsi voir chaque étape de la boucle for du code, et le contenu des variables locales pendant la boucle.

Il y a une fonction plus puissante encore pour voir l'état du programme : passer la souris sur les lignes du programme dans le débogueur :



Le débogueur est en pause, car son icône est colorée.

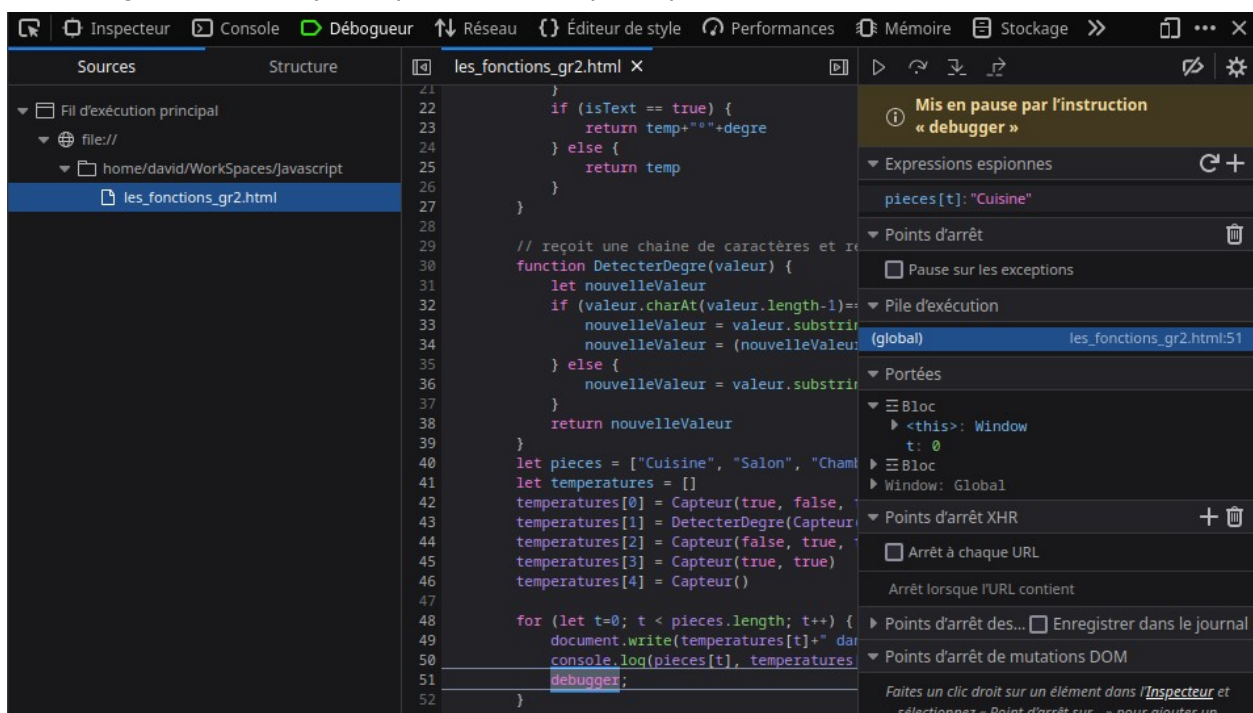
C.1.1.b Via le code

Il est aussi possible de forcer un arrêt dans le code JavaScript avec l'instruction debugger. Pour cela, l'outil de débogage doit être ouvert (sinon, le navigateur l'ignore simplement).

Exemple :

```
for (let t=0; t < pieces.length; t++) {
  document.write(temperatures[t]+" dans "+pieces[t]+"<br>")
  console.log(pieces[t], temperatures[t])
  debugger
}
```

Le débogueur affiche en jaune qu'il a été mis en pause par une instruction :

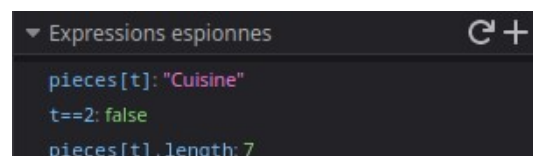


C.1.1.c Expressions espionnes

Une autre fonctionnalité est de surveiller automatiquement certaines variables (c'est notamment nécessaire pour afficher le contenu d'une cellule de tableau, par exemple, là où le survol à la souris afficherait tout le contenu du tableau).

Il suffit d'ajouter le nom de l'expression espionne dans la sous-fenêtre prévue à cet effet, comme dans l'exemple ci-dessus (juste en dessous des symboles de reprise des points d'arrêt, en haut à droite).

On peut y inscrire des variables, mais aussi le résultat de certaines fonctions ou conditions.



D Gérer les performances

Le navigateur prenant en charge toutes les opérations concernant la page et les différents téléchargements, il est possible de faire des mesures de performances.

D.1 Accès à un site

L'exemple suivant permet de constater comment fonctionne un moteur de recherche : DuckDuckGo.

Dans la barre de lien du navigateur, tapez l'adresse <https://duckduckgo.com/>

Puis dans le débogueur, cliquez sur l'onglet [Réseau] et sur le bouton [Recharger] :

État	Méthode	Domaine	Fichier	Initiateur	Type	Transfert	Taille	0 ms
304	GET	duckduckgo.com	/	document	html	mis en cache	6,01 Ko	194 ms
200	GET	duckduckgo.com	b125.js	script	js	mis en cache	0 o	0 ms
200	GET	duckduckgo.com	l132.js	script	js	mis en cache	0 o	0 ms
200	GET	duckduckgo.com	duckduckgo86.js	script	js	mis en cache	0 o	0 ms
200	GET	duckduckgo.com	u668.js	script	js	mis en cache	0 o	0 ms
200	GET	duckduckgo.com	d3149.js	script	js	mis en cache	0 o	0 ms
200	GET	duckduckgo.com	ProximaNova-Reg-webfont.woff2	font	html	mis en cache	18,08 Ko	0 ms
200	GET	duckduckgo.com	ProximaNova-Sbold-webfont.woff2	font	html	mis en cache	18,16 Ko	0 ms
200	GET	duckduckgo.com	ProximaNova-ExtraBold-webfont.woff2	font	html	mis en cache	21,03 Ko	0 ms

Vous obtenez toutes les requêtes pour l'affichage de la page de DuckDuckGo. Mais vous pouvez maintenant taper dans le champ de recherche les mots "BTS SIO" : vous pouvez aussi constater que le site échange des données en temps réels (permettant la complétion de recherche) :

État	Mét...	Domaine	Fichier	Initiateur	Type	Transfert	Taille	En-têtes	Cookies	Requête	Répc
200	GET	duckduckgo.com	/	document	html	mis en cache	6,01 Ko				
200	GET	duckduckgo.com	b125.js	script	js	mis en cache	0 o				
200	GET	duckduckgo.com	l132.js	script	js	mis en cache	0 o				
200	GET	duckduckgo.com	duckduckgo86.js	script	js	mis en cache	0 o				
200	GET	duckduckgo.com	u668.js	script	js	mis en cache	0 o				
200	GET	duckduckgo.com	d3149.js	script	js	mis en cache	0 o				
200	GET	duckduckgo.com	ProximaNova-Reg-webfont.woff2	font	html	mis en cache	18,08 Ko				
200	GET	duckduckgo.com	ProximaNova-Sbold-webfont.woff2	font	html	mis en cache	18,16 Ko				
200	GET	duckduckgo.com	ProximaNova-ExtraBold-webfont.woff2	font	html	mis en cache	21,03 Ko				
200	POST	improving.duckduckgo.com	atbhl_firefox_v359-774204679&va=_&at	u668.js:1 (beac...	scripts	Bloquée par ...					
200	POST	improving.duckduckgo.com	hi77273001&b=firefox&atbi=true&ei=tr	u668.js:1 (beac...	scripts	Bloquée par ...					
200	GET	duckduckgo.com	/ac/?q=b&kl=fr-fr	l132.js:45 (xhr)	js	2,62 Ko	177 o	État 200 OK		GET https://duckduckgo.com/ac/?q=bts&kl=fr-f	
200	POST	improving.duckduckgo.com	acp_home?9509342&g=_&biaexp=b&i	u668.js:1 (beac...	scripts	Bloquée par ...		Version HTTP/2			
200	GET	duckduckgo.com	/ac/?q=bt&kl=fr-fr	l132.js:45 (xhr)	js	2,58 Ko	157 o	Transfert 2,44 Ko (taille 168 o)			
200	GET	duckduckgo.com	/ac/?q=bts&kl=fr-fr	l132.js:45 (xhr)	js	2,44 Ko	168 o	Politique de référent origin			
200	GET	duckduckgo.com	/ac/?q=bts+&kl=fr-fr	l132.js:45 (xhr)	js	2,51 Ko	172 o	Priorité de la requête Highest			
200	GET	duckduckgo.com	/ac/?q=bts+s&kl=fr-fr	l132.js:45 (xhr)	js	2,47 Ko	188 o	En-têtes de la réponse (2,37... Texte brut			
200	GET	duckduckgo.com	/ac/?q=bts+si&kl=fr-fr	l132.js:45 (xhr)	js	2,51 Ko	227 o	cache-control: no-cache			
200	GET	duckduckgo.com	/ac/?q=bts+sio&kl=fr-fr	l132.js:45 (xhr)	js	2,44 Ko	225 o	content-disposition: attachment; filename			
200	GET	duckduckgo.com	/ac/?q=bts+sio+&kl=fr-fr	l132.js:45 (xhr)	js	2,48 Ko	235 o	content-encoding: br			
200	GET	duckduckgo.com	/ac/?q=bts+sio++&kl=fr-fr	l132.js:45 (xhr)	js	2,58 Ko	235 o	content-security-policy: default-src 'none' ;			
200	GET	duckduckgo.com	/ac/?q=bts+sio+++&kl=fr-fr	l132.js:45 (xhr)	js	2,60 Ko	235 o	connect-src https://duckduckgo.com https://			

Cliquez sur la requête contenant juste le mot "BTS" et vous constaterez que le serveur a émis une réponse contenant la liste des réponses :

État	Mét...	Domaine	Fichier	Initiateur	Type	Transfert	Taille	En-têtes	Cookies	Requête	Rénc
200	GET	duckduckgo.com	/ac?q=b&kl=fr-fr	1132.js:45 (xhr)	js	2,62 Ko	177 o				2
200	GET	duckduckgo.com	/ac?q=bts&kl=fr-fr	1132.js:45 (xhr)	js	2,44 Ko	168 o				1
200	GET	duckduckgo.com	/ac?q=bts+&kl=fr-fr	1132.js:45 (xhr)	js	2,51 Ko	172 o				
200	GET	duckduckgo.com	/ac?q=bts+s&kl=fr-fr	1132.js:45 (xhr)	js	2,47 Ko	188 o				

JSON
0: Object { phrase: "bts" }
1: Object { phrase: "bts mco" }
2: Object { phrase: "bts ndrc" }
3: Object { phrase: "bts sam" }
4: Object { phrase: "bts sp3s" }
5: Object { phrase: "bts sio" }
6: Object { phrase: "bts gpme" }
7: Object { phrase: "bts esf" }

Testez maintenant l'adresse suivante : <https://html.duckduckgo.com/html>

Cette fois, il n'y a aucune donnée échangée, car il s'agit de la version statique du moteur de recherche.

D.2 Accès à une API

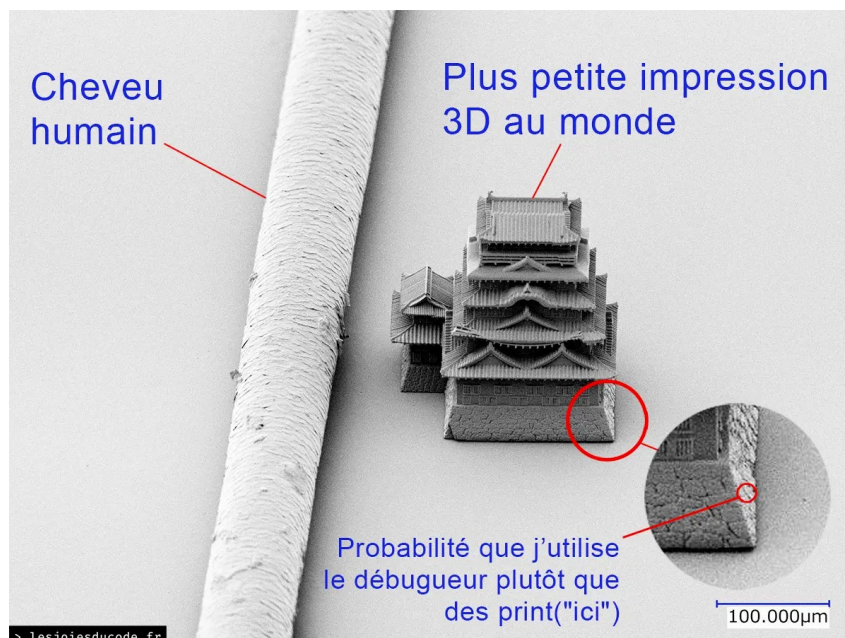
Si vous codez un serveur RESTful, vous pouvez également récupérer les données et les afficher dans le débogueur.

Testez avec l'URL suivante :

<http://restapi.adequateshop.com/api/Tourist/>

E Envoyer des messages dans la console

Si malgré les outils précédents, vous n'arrivez pas à déboguer votre programme, il existe une méthode universelle (mais malheureusement, officiellement moins reconnue).






La partie classique n'est effectivement pas la plus glorieuse, mais quelques autres fonctions sont utiles.

E.1 Messages classiques

L'usage de `console.log()` est fortement répandu est c'est effectivement un moyen d'afficher des informations rapidement, sans avoir à gérer de points d'arrêt ou de variables espions. La commande accepte plusieurs paramètres, évitant ainsi d'avoir à créer une chaîne de texte par concaténation.

```
console.log(variable1, "texte de test", variable2, ...)
```

Il existe aussi d'autres fonctions d'affichage utilisable avec console :

<code>.log()</code> ou <code>.debug()</code>	Affichage traditionnel du contenu
<code>.info()</code>	Affiche une icône devant le contenu :  Hello
<code>.warn()</code>	Affiche le contenu comme étant un avertissement :  Hello
<code>.error()</code>	Affiche le contenu comme étant en erreur :  Hello

E.2 Les autres messages

E.2.1 Afficher un tableau

La commande `console.table(variableTableau)` permet d'afficher le contenu du tableau dans un format humain :

```
>> console.table(pieces)
console.table()           ...bugger eval code:1:9
┌(index)┐ ┌Valeurs┐
├───┬───┤ ───┬───┤
0   │     │ cuisine
1   │     │ salon
2   │     │ chambre
3   │     │ salle d'eau
```

E.2.2 Afficher une durée

JavaScript permet d'afficher la durée écoulée entre une instruction `console.time(label)`, `console.timeLog(label)` et `console.timeEnd(label)` comme dans l'exemple ci-dessous :

```
function longue() {
  let temp = 0
  for (let t=0; t<100000000; t++) {
    temp = Math.acos(t)*Math.sqrt(t)
  }
}

console.time("fonctionLongue")
longue()
console.timeLog("fonctionLongue")
longue()
console.timeEnd("fonctionLongue")
```

Affichera :

```
fonctionLongue : 101 ms
fonctionLongue : 198 ms - chronomètre arrêté
```

Note : le label est optionnel (il n'existe dans ce cas, qu'un seul timer).

E.2.3 Afficher les fonctions

La fonction `console.trace()` permet de tracer le passage par une fonction ou un point de passage :

```
function longue() {
  let temp = 0
  for (let t = 0; t < 10; t++) {
    for (let j = 0; j < 100000; j++) {
      temp = Math.acos(t) * Math.sqrt(t)
    }
    maxVal(temp)
  }
}

function maxVal(x) {
  if (x > compteur) compteur = x
  console.trace()
}

let compteur = 0
console.time("fonctionLongue")
longue()
console.timeEnd("fonctionLongue")
```

Affichera :

```
console.trace()
maxVal    file:///home/david/WorkSpaces/Javascript/debug functions.html:25
longue    file:///home/david/WorkSpaces/Javascript/debug functions.html:19
<anonyme> file:///home/david/WorkSpaces/Javascript/debug functions.html:29

fonctionLongue : 2 ms - chronomètre arrêté
```

E.2.4 Autres fonctions

Quelques autres fonctions peuvent encore servir, comme `console.group()` mais cela devient très spécifique et rare d'utilisation.

On notera particulièrement `console.group()` et `console.groupEnd()` qui appliquent et retirent une indentation aux commentaires affichés.

```
15:34:57.676 | This is the outer level
15:34:57.677 |
15:34:57.679 |   Level 2
15:34:57.680 |
15:34:57.682 |     Level 3
15:34:57.683 |     ▲ More of level 3
15:34:57.685 |     Back to level 2
15:34:57.687 |   Back to the outer level
15:34:57.689 | ► undefined
```

Ou bien `console.count(label)` qui peut compter le nombre de passage dans une fonction.

Vous trouverez d'autres informations sur la page suivante :

<https://developer.mozilla.org/fr/docs/Web/API/Console/group>

F Annexes

F.1 Sources

<https://fr.javascript.info/debugging-chrome>

<https://developer.mozilla.org/fr/docs/Web/API/Console/group>