
TP0 : Outils de développement JavaScript

Ayoub KARINE (ayoub.karine@isen-ouest.yncrea.fr)

1 - Préparation

1. Sous VS code, créez un répertoire tp0
1. Créez deux fichiers factorial.html et factorial.js dans ce répertoire.
2. Créez un titre dans factorial.html et établissez une liaison avec factorial.js
3. Remplissez factorial.js avec le contenu suivant :

```
function factorial(n){  
    let result = 1;  
    for(let i=1; i <=n; i++){  
        result *= i;  
    }  
    return result;  
}  
  
function main(){  
    let n = 6;  
    let f = factorial(6);  
    console.log("factorial(" + n + ") = " + f);  
}  
  
main();
```

4. Ouvrez factorial.html avec Chrome.

5. Ouvrez les outils de développement avec CTRL+Maj+I ou en faisant un clic droit sur la page et en sélectionnant "Inspecter".

2 - Console

1. Cliquez sur "Console" et vérifiez si le résultat est bien
`"factorial(6)=720"`

Sur la droite est précisée la ligne de code ayant abouti à cet affichage. En plus, cet élément est un lien : vous pouvez cliquer dessus, vous basculez alors dans l'onglet "Sources" à la bonne ligne.

2. La console n'est pas qu'un espace d'affichage. Il s'agit également d'un interpréteur interactif dans lequel vous pouvez [évaluer des expressions](#).

Evaluer l'expression `1+1` en la tapant dans la console.

3. Une fois les scripts chargés, il est possible d'utiliser les variables ou fonctions ayant une portée globale.
Par exemple, commencez à écrire "factorial", Chrome vous propose alors tous les symboles contenant les lettres saisies. C'est ce qu'on appelle l'[auto-complétion](#).

- a. Vous pouvez alors valider la saisie avec Tab ou Entrée.

Terminez l'écriture de l'appel de fonction en écrivant par exemple `factorial(6)`, vous verrez alors l'aperçu du résultat dès que la parenthèse fermante est saisie.

- b. Essayez les appels suivants et observez le résultat (Utilisez l'auto-complétion pour sélectionner rapidement "console" puis la bonne fonction d'affichage):

```
console.warn("warning");  
console.error("error");
```

4. Il est possible de voir l'[historique](#) des commandes précédemment écrites en appuyant sur les flèches haut et bas. Essayez par exemple de ré-exécuter la dernière commande.

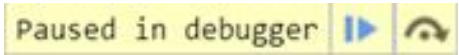
3 - Débogueur

[Point d'arrêt \(breakpoint\)](#)

Un point d'arrêt permet d'interrompre l'exécution du code au niveau d'une ligne. Pour placer un point d'arrêt, il suffit de cliquer sur le numéro de ligne.

1. Dans source, placer un point d'arrêt au niveau de la ligne 10. Il apparaîtra à deux endroits :
 - a. au niveau des numéros de ligne (matérialisé par un indicateur bleu)
 - b. au niveau de la liste des breakpoints (sur la droite)


[Exécution en mode debug](#)


1. Rafraîchissez la page en appuyant sur F5. L'exécution du script est alors bloquée au niveau du point d'arrêt que vous venez de placer. Ceci se matérialise par :
 - a. Un encadré  ajouté dans la zone de rendu de la page Web
 - b. Une ligne bleue indiquant la prochaine ligne à exécuter

Sur la droite sont affichées toutes sortes d'informations sur le contexte d'exécution courant. On y trouve notamment :

- la pile d'exécution (partie "Call stack")
- les variables (partie "Scope")

Pas à pas (step over)

Vous pouvez alors exécuter votre programme pas à pas (un pas correspondant à une ligne) en cliquant sur l'icône  (dans la barre d'outils à droite de l'onglet "Sources" ou dans l'encadré jaune sur la page Web) ou en appuyant sur F10.

1. Exécutez chacune des lignes en regardant évoluer la valeur des variables (n et f) dans le code (au niveau des déclarations) et dans la partie "Scope".
2. Placer un point d'arrêt à l'intérieur du for dans la fonction factorial(). Anticiper combien de fois vous devrez cliquer sur  pour exécuter le programme jusqu'à la fin.

Espions (watch)


Tout comme dans la console, on peut évaluer des expressions qui sont des combinaisons de symboles définis dans le contexte courant.

1. Ajouter un espion (watch) valant $2*n$

Entrée / sortie d'une fonction (step in / step out)


Il est possible d'avoir une vue détaillée de ce qui se passe à l'intérieur des fonctions en utilisant step in.

1. Rafraîchir la page web

2. Faites des step over jusqu'à la ligne contenant l'appel à `factorial()`. Quand vous arrivez sur cette ligne, cliquez sur  (step in) ou appuyez sur F11.


Vous entrez alors à l'intérieur de la fonction et constatez que :

- l'appel à la fonction `factorial()` s'est ajoutée à la pile d'exécution (call stack)
- les variables de la partie "Scope" ont changé : il s'agit maintenant des variables locales de la fonction `factorial()`

3. Faites quelques step over dans la fonction, vous voyez alors les variables évoluer au fur et à mesure des itérations du `for`. Quand vous en aurez assez, vous pourrez sortir de la fonction en cliquant sur  (step out) ou en appuyant sur SHIFT+F11. Vous reviendrez alors dans votre programme principal, juste après l'exécution de `factorial()`.

Navigation entre les points d'arrêt


Il est possible d'aller directement d'un point d'arrêt à un autre.

1. Placez un point d'arrêt sur la ligne du `return` de la fonction `factorial()` et rafraîchissez la page
2. Cliquer sur , le programme va être exécuté jusqu'au prochain point d'arrêt (ici en fin de fonction `factorial()`). Le prochain clic terminera l'exécution du code, car aucun autre point d'arrêt n'est présent dans le programme.

Débogage d'une fonction récursive

1. Remplacer la version itérative de la fonction factorial() par la version réursive ci-après :

```
function factorial(n) {  
  
    if(n <= 0) {  
        return 1;  
    }  
    return n * factorial(n - 1);  
}
```

2. Placez un point d'arrêt au niveau du if et observez l'évolution de la pile d'exécution à chaque fois que vous cliquez sur  .

Pour aller plus loin

[Get Started with Debugging JavaScript in Chrome DevTools](#)