Le campus numérique in the Alps

DIGITAL GRENOBLE

Document Élève : Débug

Lors des premiers exercices permettant d'exécuter du code dans le navigateur vous avez certainement utilisé la console et "console.log" pour afficher la valeur de certaines expression dans cette console. Cette console est utilisable depuis les "Development Tools" (F12) sous Chrome par exemple.

Cela permet effectivement d'afficher des informations permettant de comprendre et/ou de mettre au point des programmes, ou d'interagir avec le code depuis cette console. Mais les possibilités apportées par cette approche est vite limitée et nécessite souvent de faire des modifications dans le code ("console.log" etc…) ce qui n'est pas toujours possible ou souhaitable.

Afin d'aller plus loin dans la compréhension du code s'exécutant, il est important de maîtriser les outils de "debug" ("débogueur" en français, mais on entend couramment "débugueur"). L'activité est proposée avec le navigateur Chrome, mais peut être faite sous Firefox par exemple (https://developer.mozilla.org/fr/docs/Outils/D%C3%A9bogueur).

Débug dans le navigateur

Suivre les informations de la page
https://developers.google.com/web/tools/chrome-devtools/javascript/

Débug depuis Visual Studio Code

Le navigateur peut exposer l'état de l'exécution à travers un interface de programmation (API, Application Programming Interface). Nous n'entrons pas dans les détails de ce fonctionnement ici, mais cela permet à une application externe de connaître l'état précis de l'exécution en cours (pile d'exécution, variables, etc).

C'est cette possibilité que nous allons explorer en débugant du code JavaScript depuis Visual Studio Code au lieu des outils de développement dans Chrome. Pour cela, installer dans Visual Studio Code l'extension "Debugger for Chrome" (plus de détails sur https://github.com/Microsoft/vscode-chrome-debug).

Note : cela est également possible avec Firefox : https://developer.mozilla.org/fr/docs/Outils/D%C3%A9bogueur

Premier essai

Écrire une fonction permute(a, b) qui permute (et oui !) les valeurs des deux variables. Appeler le script de cette fonction depuis une page HTML qui l'exécute avec deux valeurs prédéterminées et qui affiche le résultat.

Procéder ensuite à une exécution pas à pas depuis VS Code.

Si certains ont du temps, ils/elles peuvent chercher comment saisir les valeurs a et b depuis la page Web (voir le code du tutoriel du débogueur).

Deuxième essai : un petit goût de récursion

Le monde des listes

Le monde des structure de données "listes" est un monde fascinant et certains langages de programmation s'en sont fait une spécialité. Ce n'est pas le cas de Javascript (même si certaines librairies peuvent ajouter les listes). Ces listes vont nous permettre d'appréhender la notion de récursivité.

Une liste est un structure qui peut se définit de la façon suivante. Une liste est :

- soit la liste vide;
- soit une valeur, suivie d'une liste.

Comme vous le voyez, on définit une liste… à partir d'une liste. C'est un des aspects de la récursion.

En Javascript - du moins en JSON - nous pouvons décider de représenter la liste vide par “null” et un liste non vide par un objet de deux champs - la valeur et le reste de la liste. Par exemple, la liste (1; 2; 3; 5; 8) (aucun rapport avec une notation Javascript) peut être écrite en JSON :

```
{
  valeur: 1,
  reste:
      {
        valeur: 2,
        reste:
            {
              valeur: 3,
              reste:
                  {
                    valeur: 5,
                    reste:
                        {
                          valeur: 8,
                          reste: null
                        },
                  },
            },
      },
}
```

Ecrivez une fonction Javascript qui calcule la longueur d’une telle liste, c’est-à-dire le nombre de valeurs. Indice : de la même façon qu’une liste se définit avec une autre liste, la fonction longueur peut s’appeler elle-même.

Observez les appels des fonctions dans le débugueur.

Bonus : essayez de transformer la fonction récursive en fonction non récursive (indice : utilisez un compteur et un parcours par exemple avec une boucle *while*). À cause des appels dans des appels dans des appels…. Les fonctions récursives peuvent poser des problèmes de performance, voire générer des erreur (épuisement de la mémoire) : dans le cas de structures contenant beaucoup de données, il est préférable d’avoir des fonctions non récursives.

Les racines de la récursion

L’objectif est d’écrire une fonction qui détermine la racine carrée d’une valeur en utilisant la méthode de Newton. Cela consiste à déterminer une valeur approchante puis se rapprocher progressivement de la solution. On s’arrête quand l’erreur est inférieure à une valeur

pré-déterminée. La lecture de la page Wikipedia
https://github.com/Microsoft/vscode-chrome-debug n'est pas nécessaire !

Nous avons besoin de deux paramètres :

- La valeur x dont on veut calculer la racine carrée
- La précision attendue p (on peut également fixer une proportion de x, par exemple x/100)

Le point de départ est arbitraire, prenons r=0 pour démarrer (r sera le résultat).

L'algorithme est le suivant :

- Fixer x
- Fixer p
- r = 0
- Tant que |r*r - x| > p faire : // l'écart à la bonne valeur n'est pas satisfaisant
    - r = (r+x/x)/2
- r contient le résultat approximatif

Coder cet algorithme et suivre un calcul dans le débogueur.

L'algorithme précédent est itératif. Nous pouvons également supprimer la boucle "tant que" par une fonction qui se rappelle (que l'on appelle récursive), souvent plus proche des propriétés de la solution que de la façon d'y arriver :

- Fonction racine(x, r, p) :
    - Si |r*r - x| <= p
        - Alors renvoyer r
        - Sinon, renvoyer racine(x, (r+x/x)/2, p)

À nouveau, suivre l'exécution dans le débogueur.

Faisons le tri (et vite !)

Le choix d'un algorithme de trie est important : nous avons souvent à traiter - et à trier - un nombre important de données. Il est donc judicieux de choisir des algorithmes rapides.

Dans la suite, nous considérons que des tableaux de nombres. Vous pouvez suivre l'exécution de différents algorithmes de tris sur https://visualgo.net/bn/sorting?slide=1

Essayez d'écrire une fonction de tri d'un tableau de nombre en JavaScript, en utilisant l'algorithme Quick Sort.