ARTRAGIS

Zeste de savoir

Les macros google spreadsheet

12 août 2019

Table des matières

Google vous propose parmi tous les services de son *drive* un logiciel de tableur nommé « Google Spreadsheet ».

Comme beaucoup de tableurs, ce dernier permet d'automatiser les tâches via des macros. C'est ce que nous allons voir dans ce tutoriel.

i

J'ai tenté de rendre ce tutoriel le plus accessible possible aux utilisateurs de base de Spreadsheet : les personnes qui font de la bureautique ou des formulaires de sondage. Néanmoins, pour votre confort, avoir déjà vu un petit code et en avoir compris le fonctionnement *en gros* est conseillé.
L'objectif de ce tutoriel est qu'à la fin vous puissiez rapidement faire un petit script qui vous permette de vous faire gagner du temps lorsque vous utilisez ce tableur.
Pour obtenir de l'aide, le forum « Programmation » vous est ouvert, utilisez le tag « Google Spreadsheets » pour donner plus de visibilité à votre message.

1. Une macro, pour quoi faire ?

Lorsque vous utilisez un logiciel de tableur, qu'il s'agisse des classiques Excel ou Classeur[1], ou bien d'une application web telle que Google SpreadSheet ou encore Ethercalc, vous arriverez un jour face à cette problématique :

> Je suis en train de faire quelque chose de très répétitif mais il n'existe aucune formule pour le faire à ma place.

Cette situation n'est pas très agréable, et le sera encore moins lorsque votre feuille de calcul sera très longue (ou très large), qu'il faudra beaucoup *scroller* et que le moindre copier/coller sera sujet à des blocages de l'interface graphique.

1. Pour rappel, Excel est le tableur-grapheur édité par Microsoft dans sa suite Office (et office 365). Classeur est le nom francophone du tableur-grapheur de Libre Office, le *fork* le plus actif de Open Office. Je ne résiste pas à l'envie de vous donner ce lien de téléchargement .

Les développeurs sont souvent confrontés à ce problème et, dès lors, ils ont une solution : ils arrêtent de faire le travail eux-même pendant quelque temps et en profitent pour coder un programme –on parle aussi de *script* – qui fera la tâche à leur place.

FIGURE 1. – efficacité du programmeur en fonction du temps[2]

Cette idée a été reprise dans les grands logiciels de tableur via le concept de **macro**. Il s'agit d'un *script*, créé dans un langage souvent simplifié possédant plein d'outils et qui permet d'automatiser les tâches répétitives tout en les intégrant à l'interface graphique (ajout de bouton, de menu).

Google Spreadsheet ne coupe pas à cette logique. La seule différence c'est que les feuilles de calculs sont exécutées et gérées par votre navigateur. Du coup, pour créer ses *macros*, Google a choisi le langage **JavaScript**[3].

2. Manipuler la table de données

2.1. Les débuts de notre macro

Pour créer votre script, allez dans « Outils » puis « Éditeur de script ».

Vous arrivez sur une nouvelle page (attention, si vous fermez ou actualisez avec F5 votre feuille de calcul, il arrive que cette page se ferme automatiquement).

Un bout de code est écrit sur cette page :

```
1 function myFunction() {
2
3 }
```

Listing 1 – La fonction de base

Si vous avez déjà programmé dans votre vie, ce bout de code ne devrait pas vous effrayer. Sinon, imaginez simplement qu'il s'agit ici d'un modèle à respecter : le mot `function` est là pour dire que vous aller définir un ensemble d'actions à exécuter. Le mot `myFunction` est simplement le nom de votre macro. D'ailleurs, si vous regardez la barre d'outils de votre éditeur de script, l'interface vous propose de sélectionner cette fonction. Si vous changez le nom et que vous sauvegardez (CTRL + S) le nom changera.

2. Notons que ce graphique marche aussi bien en étant lu « nombre d'opérations réalisées en fonction du temps » que « litres de café ingurgités en fonction du temps ».

3. C'est un des langages du web qui a permis de dynamiser les sites. Si vous voulez voir l'étendue de ses possibilités en dehors de Google SpreadSheet, vous pouvez jeter un coup d'œil ici

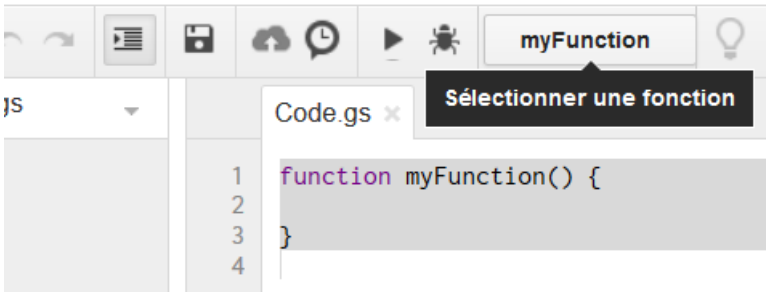
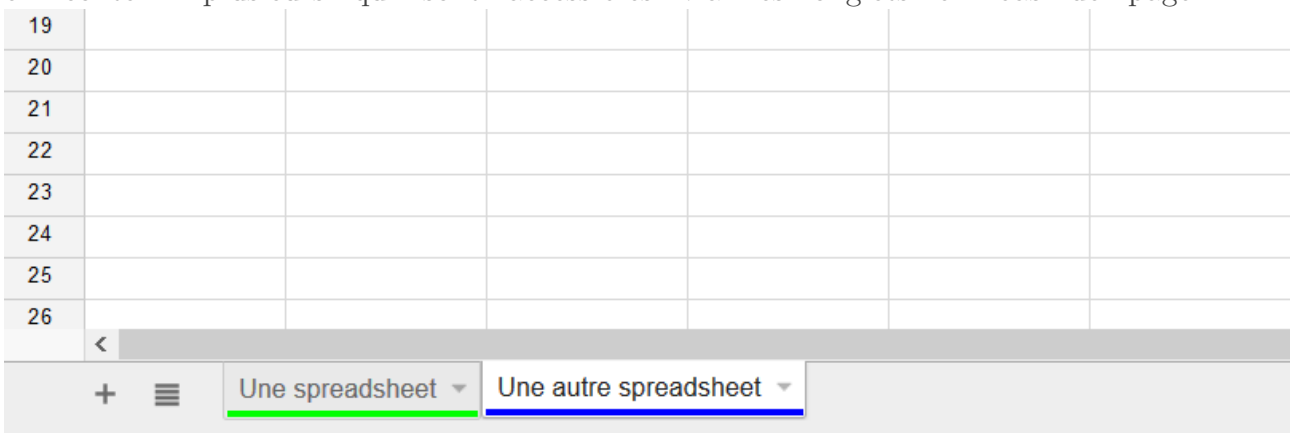


FIGURE 2. – Sélectionnez votre macro.

Dans vos fonctions, vous avez droit à plusieurs objets globaux dont `Browser` et `SpreadsheetApp`. Le premier permet d'interagir avec l'utilisateur (avec une `inputBox` par exemple), le second de contrôler ce que vous faites dans votre feuille de calcul.
Notons juste que du point de vue de Google, tout est organisé selon trois niveaux de hiérarchie :

— La `SpreadsheetApp` qui contient l'ensemble des objets que vous manipulez. En fait, vous pouvez contrôler toutes les `SpreadsheetApp` des documents tableur qui sont dans votre *drive*.

— La `Spreadsheet` : c'est le document de tableur dans sa globalité. Il possède par défaut une feuille de calcul (`Sheet`). Chaque document peut en contenir plusieurs qui sont accessibles via les onglets en bas de page.



— Le `Range` : c'est une plage de cellules (une cellule est vue par Google comme un `Range` de 1 sur 1).

Exécuter une macro demande certaines autorisations. Cette politique de sécurité implique qu'il est impossible de mettre des scripts sur des feuilles publiques. Préférez un partage privé pour cela.

2.2. Ecrire et lancer une macro

2.2.1. Jouons avec nos spreadsheets

La première chose à faire, est donc se savoir quelle est la `SpreadsheetSheet` que nous allons manipuler.
Pour cela, il faut piocher dans les *fonctions* de `SpreadsheetApp` et aller trouver `getActiveSpreadsheet`.
Notre premier défi consistera à afficher son nom. C'est dans `Browser` que nous trouverons la solution.

En effet, comme nous l'avons précisé auparavant, ce dernier permet d'interagir avec l'utilisateur. Commençons par simplement afficher une boîte à message contenant le titre du document courant.

```
function myFunction() {
  var document = SpreadsheetApp.getActive();
  Browser.msgBox(document.getName());
}
```

Listing 2 – Afficher le nom dans une `msgbox`.

Il ne reste plus qu'à lancer notre macro. Pour cela : appuyez sur le petit triangle « play » dans l'éditeur de script. Vous devriez obtenir un message lorsque vous affichez votre feuille de calcul.

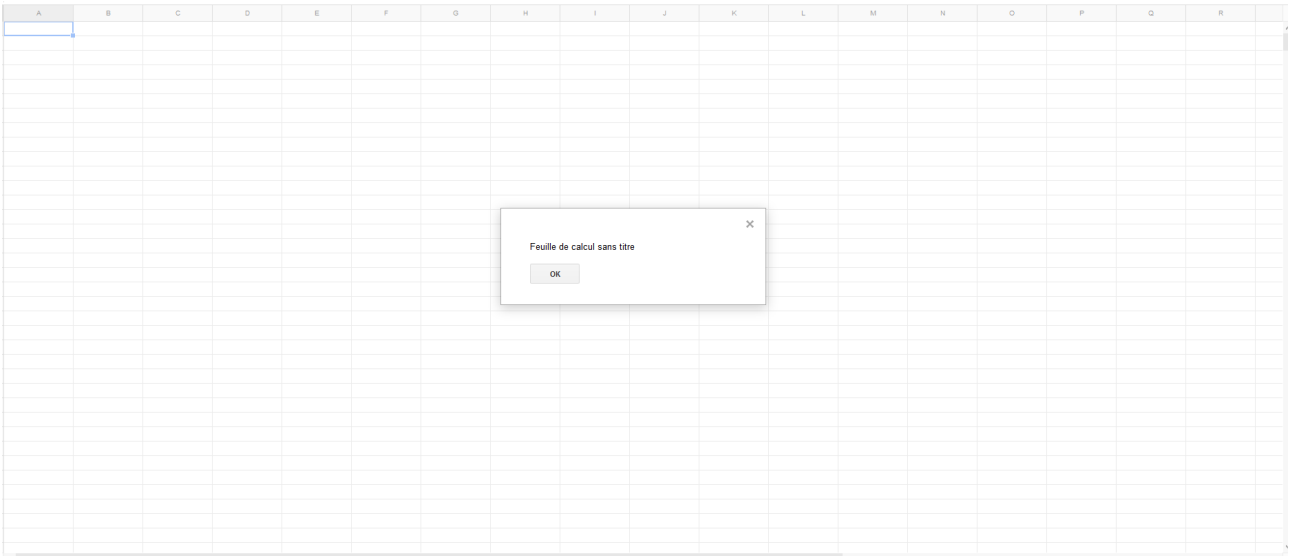


FIGURE 2. – Affichons notre message.

i

On peut observer que c'est le titre du document (`Spreadsheet`) qui est affiché et non le titre de la feuille (`Sheet` qui s'appelle *Feuille 1* par défaut). Pour obtenir le nom de la feuille courante, il faut faire `getSheetName` à la place de `getName`.

Maintenant, créons une seconde feuille dans notre document. Pour pouvoir naviguer entre les feuilles, il vous suffira de connaître leur nom.

```
function myFunction() {
  var document = SpreadsheetApp.getActive();
  var sheet = document.getSheetByName("Feuille 1");
  document.setActiveSheet(sheet);
  Browser.msgBox(document.getSheetName());
  sheet = document.getSheetByName("Feuille 2");
  document.setActiveSheet(sheet);
  Browser.msgBox(document.getSheetName());
```

```
9 }
```

Listing 3 – Le code change de page et affiche leur nom.

2.2.2. Obtenir les données d'une sélection

Pour que votre macro travaille sur des données, vous pouvez
- renseigner en dur la plage d'action ;
- sélectionner la plage à la main avant d'activer la macro.

Dans le premier cas, la fonction à aller chercher est `SpreadsheetApp.getRange("plage de cellule")`. Vous utilisez les plages comme d'habitude (par exemple `A1:B2` est une plage carrée de quatre cases en partant d'en haut à gauche). Vous pouvez aussi définir la plage avec des entiers (sachant que les indices commencent à 1, vous auriez donc `SpreadsheetApp.getRange(1, 1, 2, 2)`). Une fois que vous avez obtenu votre plage, il faut aller chercher les valeurs grâce à la fonction `getValues()` qui renverra un tableau Javascript (indicé à partir de 0 !) à deux dimensions, le premier indice correspondant à la ligne, le second à la colonne.

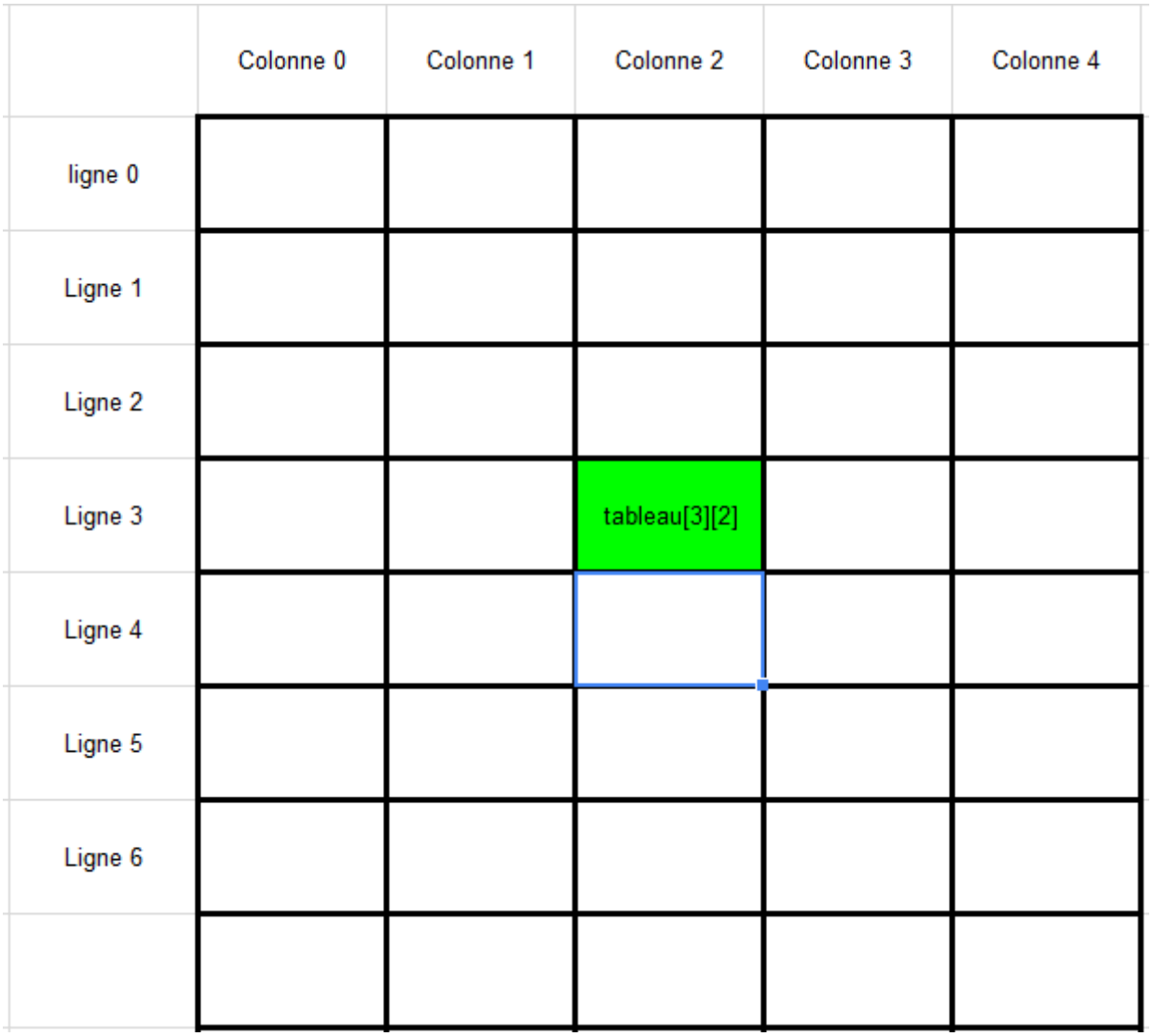


FIGURE 2. – Les tableau de javascript sont indicés à partir de 0.

```
function maMacro(){
    var spreadsheet = SpreadsheetApp.getActiveSpreadsheet();
    var range = spreadsheet.getRange("A1:B10");
    var values = range.getValues();
    var s = 0;
    for(var i=0; i<values.length; i++){
         s += values[i][1];
    }
    Browser.msgBox("le premier titre est " + values[0][0] +
        "la somme est de " + s);
}
```

Listing 4 – Fait la somme de la seconde colonne d'une plage.

Dans le second cas, la procédure est la même, mais avec `SpreadsheetApp.getActiveRange()`.
Exécutez la macro. Retournez sur votre feuille, et s'il y avait de bonnes données, vous devriez avoir un beau message qui apparaît.

2.2.3. Plus loin : plusieurs tableurs, partage...

Un cas qui pourra vous permettre d'utiliser au maximum les possibilités des scripts est le suivant :

1. Vous avez une feuille avec beaucoup de données détaillées.
2. Tous les mois, vous aimeriez qu'une synthèse vous soit envoyée sans les détails trop précis.

La macro devra, comme précédemment, aller chercher toutes les données qui l'intéressent. Puis, il vous faudra créer la nouvelle feuille grâce à la méthode `create`.
Tout de suite après cela, vous voudrez sûrement partager la feuille, les fonctions `addEditors` et `addViewers` sont là pour ça.

```
var sheet = SpreadsheetApp.create("Nouvelle synthèse"); //on crée
sheet.addEditors(["monAmi@gmail.com",
    "mongooglegroup@googlegroups.com"]); //on partage
sheet.appendLine(["titre1", "titre2"]); //on met des titres
```

Listing 5 – Partager un nouveau tableur

Enfin, vous allez placer les données. Et là mon conseil, c'est d'y aller ligne par ligne. D'une part parce que l'API est plus utilisable dans ce sens (`appendRow` permet d'ajouter des données, alors que `insertColumn` ne fait que mettre des cellules vides) mais aussi parce que ça vous permettra de mieux séquencer votre script.

3. Lancer une macro automatiquement

Nous avons vu comment lancer notre macro/script directement via l'éditeur. Mais avouons que cela n'est pas pratique si nous devons toujours ouvrir un deuxième onglet pour pouvoir exécuter le code qu'on a produit.

3.1. Les déclencheurs

Pour cela Google vous fournit un nouvel outil : les déclencheurs (*trigger* en anglais). Ces outils réagissent à un événement et *déclenchent* votre macro.
Les événements disponibles sont assez basiques :

- régulièrement dans le temps (toutes les heures, tous les jours...),
- lorsque la page s'ouvre,
- lorsque la feuille est modifiée,
- lorsqu'un utilisateur valide le Google Form qui est associé à la feuille (uniquement lorsque vous utilisez Google Form)

Pour ajouter un déclencheur, il vous suffit d'aller dans le menu « Édition » puis « Tous les déclencheurs ». Dans la fenêtre qui apparaît, cliquez sur le lien « Ajouter un nouveau déclencheur » et configurez-le comme vous le désirez.

FIGURE 3. – Ajouter un nouveau déclencheur

3.2. Mettre une macro dans un menu

Comme nous avons pu le remarquer, les déclencheurs ne permettent pas nativement de dire « lorsque j'appuie sur un bouton, lance ma macro ». Une technique existe néanmoins qui vous permettra d'arriver à ce résultat.
Elle consiste à ajouter un menu à la barre d'outils et une fois le menu ajouté le lien entre la macro et le clique sur le menu est automatique.
Pour ajouter le menu, nous allons faire une macro qui appelle certaines fonctions de `SpreadsheetApp`, puis nous allons simplement lier cette macro au déclencheur « À partir de la feuille de calcul » => « À l'ouverture ».
C'est pour ça que par convention on aime bien appeler la macro qui génère le menu `onOpen`.
Dans la fonction `onOpen()`, nous allons demander à l'interface graphique de notre `SpreadsheetApp` de créer un menu et d'y insérer notre macro.

```
1  function laMacroQuiVeutUnBouton(){
2
3  }
4  function onOpen() {
5    var ui = SpreadsheetApp.getUi();
6
7    ui.createMenu('Macros') // ce qui est vu dans la barre d'outils
```

```
8       .addItem('Appliquer ma macro', 'laMacroQuiVeutUnBouton') //
            le deuxième paramètre fait le lien entre le clique sur le
            menu et la macro.
9       .addToUi();
10  }
```

Et voilà, maintenant vous êtes capable de créer votre script. S'il vous manque des fonctions, je vous donne une dernière astuce : à partir de l'objet le plus logique pour la fonction recherchée (par exemple, pour gérer les titres, `SpreadsheetApp` est indiqué alors que pour gérer les valeurs, c'est sur un `Range` qu'il faut travailler) mettez un « . » puis appuyez sur CTRL + ESPACE et la liste des fonctions disponibles apparaîtra.