LYCÉE ARISTIDE BERGÈS
SIO

Exploration

101 Intérêt de la programmation

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision
Juillet 2022	Création

Sommaire

A Introduction

Savoir programmer est un moyen de gagner du temps. La plupart des actions qui sont programmables, peuvent être réalisées manuellement. Il y a plusieurs intérêts à utiliser un programme :

- éviter la perte de temps : rédiger un code peut faire gagner du temps (il faut alors évaluer combien de temps nous allons économiser)
- éviter les erreurs : un programme effectuera des calculs de manière fiable, tandis que les humains peuvent faire des erreurs de saisies ou de calculs.

A.1 Présentation

Cette exploration va prouver que la programmation peut rendre les tâches plus faciles.

A.2 Prérequis

Aucun

B Contexte

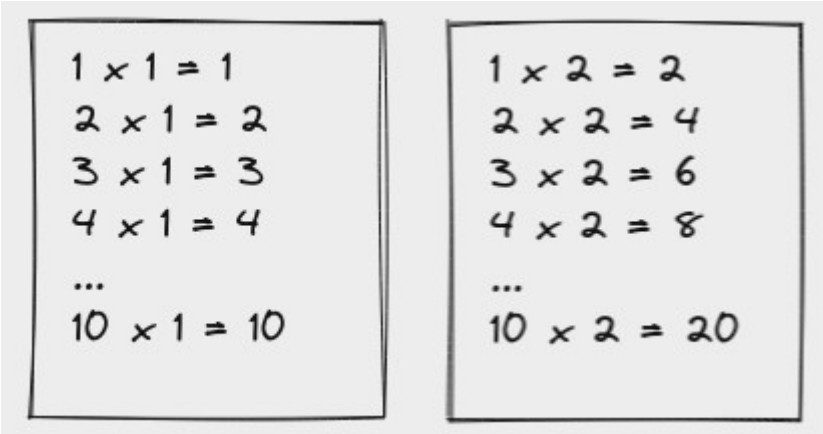
Votre voisin a un enfant de 7 ans qui a du mal a apprendre les tables de multiplication.

Son père sait que vous faite des études en informatique et il connaît l'intérêt pédagogique des jeux et applications sur ordinateur.

Il vous demande si vous pourriez faire un programme pour aider son fils Mattéo à apprendre les tables de multiplication.

B.1 Cahier des charges

Le père de Mattéo a une idée précise d'une première approche : il vous fait le dessin suivant :



a) Il vous demande de créer une page web avec un affichage similaire.

B.2 Évolutions

b) Il vous demande s'il serait possible d'écrire les tables de multiplication jusqu'à 10 ?

c) Il vous demande s'il serait possible de créer des tables de 1 à 14 par exemple ?

C Réalisation

À l'aide de votre éditeur favori, vous allez créer plusieurs versions pour les présenter à Mattéo et son père.

C.1 Multiplications en HTML

Dans un premier temps, vous décidez de créer les tables de multiplication en HTML. Comme ce n'est pas un langage de programmation, vous devrez écrire toutes les tables.

Voici votre premier code :

TableMultiplication_1.html

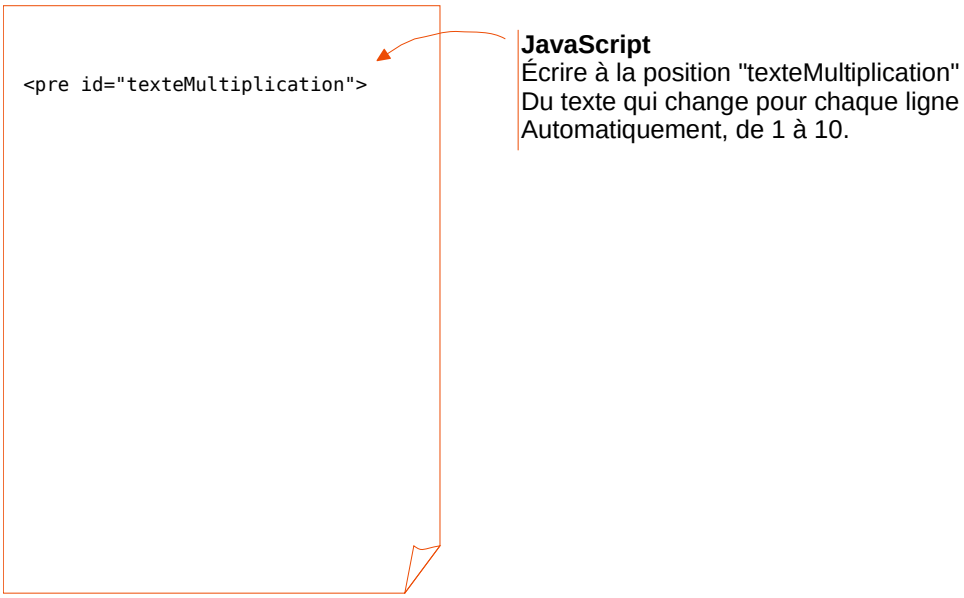
```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>01 - Table de multiplication</title>
</head>
<body>
    <h1>01 - Table de multiplication</h1>
    <p>Il suffit de compléter le paragraphe suivant (par des copier/coller) et modifier les
valeurs de 3 à 10</p>
    <pre>
2 x 1 = 2
2 x 2 = 4
    </pre>
</body>
</html>
```

Vous sauvegardez le document et vérifiez dans un navigateur (Firefox, Chrome, Edge ou autre) que le résultat correspond à la demande du père de Mattéo.

Complétez le code pour la table de 2.

C.2 Multiplications en JavaScript

JavaScript est capable d'agir dans la page HTML et d'y afficher du contenu. Votre deuxième programme va utiliser cette propriété pour demander à JavaScript de calculer pour vous une table.

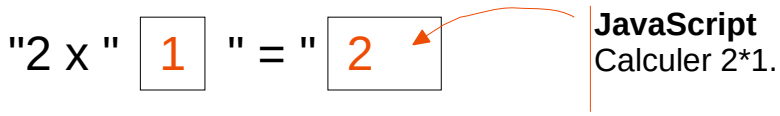


Il faut donc préciser une zone dans laquelle JavaScript va pouvoir écrire : c'est la balise HTML qui contiendra la propriété `id="texteMultiplication"` qui servira.

Ensuite, il faut écrire chaque ligne de la multiplication avec deux changements :

- Le chiffre multiplié évolue de 1 à 10
- Le résultat est un calcul sur le chiffre multiplié

En programmation, nous allons utiliser une itération (ou répétition), c'est-à-dire une suite de code qui effectue le même travail mais avec une variation.



Il faut donc créer une chaîne de caractères qui contient :

- du texte fixe, comme "2 x " ou " = "
- une zone variable, dont le contenu évoluera de 1 à 10
- une zone variable calculée

Voici le programme :

TableMultiplication_2.html

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>01 - Table de multiplication</title>
</head>
<body>
    <h1>01 - Table de multiplication (semi-automatique)</h1>
    <p>Il suffit de changer la valeur de 'maximum' dans le code</p>
<pre id="texteMultiplication">
</pre>
    <script>
        let zoneEcrire = document.getElementById("texteMultiplication")
        let maximum = 10
        for (let nombre = 1; nombre <= maximum; nombre = nombre + 1){
            zoneEcrire.innerHTML = zoneEcrire.innerHTML + "2 x "+nombre+" = "+2*nombre+"\n"
        }
    </script>
</body>
</html>
```

Attention, le code JavaScript est intégré dans la page HTML : c'est pratique mais ce n'est pas recommandé. Les développeurs préfèrent séparer le code HTML et le code JavaScript dans des fichiers séparés, nous verrons comment faire plus tard…

Voici le résultat :

01 - Table de multiplication (semi-automatique)

Il suffit de changer la valeur de 'maximum' dans le code

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Essayez de changer la valeur de la variable `maximum` en changeant la valeur 10 par un 20.

C.3 Automatisation table de multiplication

Pour permettre de changer le nombre maximum de multiplication, il faut aller lire un champ HTML, prévu à cet effet. Dès qu'un changement intervient sur celui-ci, JavaScript va exécuter un morceau de code, toujours identique (à chaque changement, c'est le même code qui s'exécute) : il faut donc créer une fonction avec ce code réutilisable.

Il faut également "écouter" les changements du champ de saisie. Enfin, il faut appeler la fonction lors du premier affichage de la page. Voici le code :

TableMultiplication_3.html

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>01 - Table de multiplication</title>
</head>
<body>
    <h1>01 - Table de multiplication (automatique)</h1>
    <p>Le 'maximum' est saisie par l'utilisateur sur la page (plus d'accès au code)</p>
    Saisir le maximum : <input type="number" id="saisie" name="" placeholder="10" value="10"
oninput="multiplication()">
<pre id="texteMultiplication">
</pre>
    <script>
        // création d'une fonction pour exécution à chaque changement de valeur
        function multiplication() {
            let zoneEcrire = document.getElementById("texteMultiplication")
            let maximum = document.getElementById("saisie").value
            zoneEcrire.innerHTML = ""
            for (let nombre = 1; nombre <= maximum; nombre = nombre + 1){
               zoneEcrire.innerHTML = zoneEcrire.innerHTML + "2 x "+nombre+" = "+2*nombre+"\n"
            }
        }
        multiplication()
    </script>
</body>
</html>
```

Pour le moment, l'écoute est gérée côté HTML avec la propriété `oninput="multiplication()"` mais plus tard, HTML ne servira qu'à l'affichage et nous créerons l'écoute du champ avec JavaScript.

Le code à droite du trait bleu, contient la fonction : elle n'est exécutée par JavaScript, que lorsqu'on l'appelle par son nom, ici `multiplication()`. Il s'agit donc de sa définition ou déclaration.

En HTML, pour créer un champ, on utilise une balise `<input type="####">` et en remplaçant #### par le type de données à saisir (text, number, password…)

Ce code est très pratique et il n'est pas nécessaire de l'éditer pour pouvoir changer le maximum. L'utilisateur n'a pas besoin de notions en programmation pour utiliser la page, mais nous n'avons qu'une seule table de multiplication dans la page.

C.4 Amélioration du rendu (code CSS)

Le père de Mattéo a précisé dans son dessin, que la table devait être encadrée. Pour cela, nous allons utiliser un autre langage, appelé CSS. Ce langage permet de séparer le fond (texte et balises HTML) et la forme (couleurs, polices, cadres, etc.)

Comme pour JavaScript, les développeurs préfèrent mettre le code CSS dans un fichier séparé, mais ici, nous allons continuer à tout conserver dans le même fichier que voici :

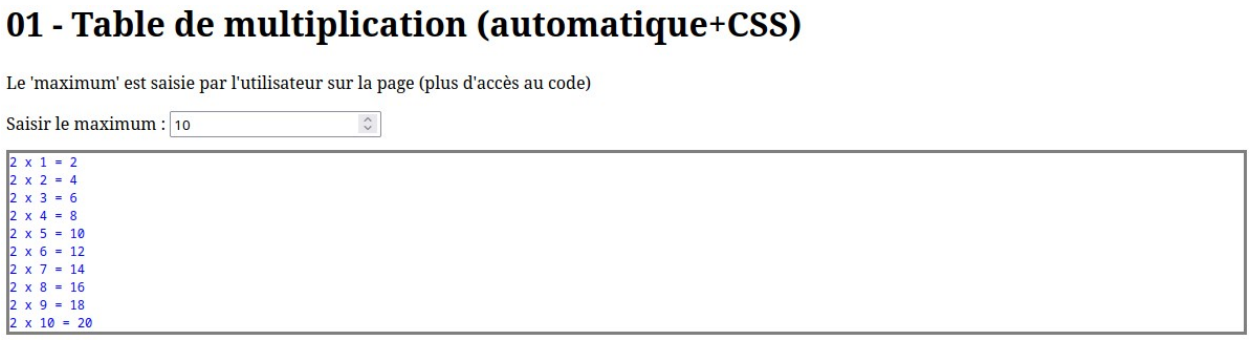
TableMultiplication_4.html

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>01 - Table de multiplication</title>
    <style>
        #texteMultiplication {
            display: flex;
            border-style:solid;
            border-color:gray;
            color:blue;
        }
    </style>
</head>
<body>
    <h1>01 - Table de multiplication (automatique+CSS)</h1>
    <p>Le 'maximum' est saisie par l'utilisateur sur la page (plus d'accès au code)</p>
    Saisir le maximum : <input type="number" id="saisie" name="" placeholder="10" value="10"
oninput="multiplication()">
<pre id="texteMultiplication">
</pre>
    <script>
        // création d'une fonction pour exécution à chaque changement de valeur
        function multiplication() {
            let zoneEcrire = document.getElementById("texteMultiplication")
            let maximum = document.getElementById("saisie").value
            zoneEcrire.innerHTML = ""
            for (let nombre = 1; nombre <= maximum; nombre = nombre + 1){
                zoneEcrire.innerHTML = zoneEcrire.innerHTML + "2 x "+nombre+" = "+2*nombre+"\
n"
            }
        }
        multiplication()
    </script>
</body>
</html>
```

Le code à droite du trait violet est en CSS : il indique que dès qu'on trouve la balise `id="texteMultiplication"` on appliquera le style bordure solide, grise et écriture bleue.

Grâce à CSS, on améliore facilement l'affichage d'une page web : CSS s'applique aux balises HTML, comme `<body>`, `<div>` ou bien aux balises contenant les propriétés `id=""` ou `class=""`.

Le résultat est le suivant :



Dans l'idéal, il faudrait encore avoir plusieurs tables côte à côte, diminuer la largeur du cadre et pour réussir cela, il faut "casser" notre code existant :

C'est un concept important en informatique, comme dans tout travail créatif ! Il faut être capable de détruire une partie du travail, pour l'améliorer. Les programmeurs experts ont tendance à réfléchir à un code très évolutif, permettant de tout faire… mais avant d'y parvenir, ils ont d'abord rédigé beaucoup de codes qu'ils ont dû modifier, changer, effacer, refaire, etc.

C.5 Code final

Voici le code définitif.

TableMultiplication_5.html

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>01 - Table de multiplication</title>
    <style>
        #texteMultiplication {
            display: flex;
        }
        .cadre {
            border-style:solid;
            border-color:gray;
            color:blue;
            font-family: courrier;
            padding: 3px;
        }
    </style>
</head>
<body>
    <h1>01 - Table de multiplication (complet)</h1>
    <p>Le 'maximum' et le 'nombre de table' sont saisis par l'utilisateur sur la page (plus
d'accès au code)</p>
    Saisir nombre de tables : <input type="number" id="nbTable" name="nbTable" placeholder="10"
value="5" >
    Saisir le maximum : <input type="number" id="saisie" name="maximum" placeholder="10"
value="10" ></br>
<div id="texteMultiplication">
</div>
    <script>
        // initialisation globale des champs de saisie
        let maximum = document.getElementById("saisie")
        let nbTable = document.querySelector("#nbTable")
        // création d'une fonction pour exécution à chaque changement de valeur
        function multiplication() {
            let zoneEcrire = document.getElementById("texteMultiplication")
            let texte = ""
            zoneEcrire.innerHTML = ""
            // Dessiner des tables
            for (let table = 1; table <= nbTable.value; table = table + 1) {
                texte = texte + "<div class='cadre'><pre>"
                // Dessiner une table de multiplication de 1 à Maximum
                for (let nombre = 1; nombre <= maximum.value; nombre = nombre + 1){
                    texte = texte + table + " x "+nombre+" = "+table*nombre+"\n"
                }
                texte = texte + "</pre></div>"
            }
            zoneEcrire.innerHTML = texte
        }
        // Création des événements
        maximum.addEventListener("input", multiplication)
        nbTable.addEventListener("input", multiplication)
        multiplication()
    </script>
</body>
</html>
```

Attention, certaines lignes étant trop longues, elles passent à la ligne : si vous rencontrez des problèmes d'exécution du code, vérifiez que ce n'est pas un retour chariot en trop (regardez l'indentation des lignes, plus on s'éloigne du bord et moins on devrait revenir en début de ligne).

Dans ce code, nous avons trois parties principales :

- Le code CSS (écriture violette) contenu entre les balises HTML `<style>` et `</style>`,
- Le code JavaScript (écriture verte) contenu entre `<script>` et `</script>`,
- Le reste est du HTML

Nous avons désormais deux champs de saisie (un pour la longueur de la table et un pour le nombre de table), ce qui explique qu'il y a deux boucles (itérations) dans le code JavaScript.

Les lignes JavaScript qui commence par `//` sont des commentaires (le navigateur ne les lit pas, elles ne servent que pour le développeur).

Ensuite, la gestion des événements n'est plus dans la partie HTML, mais est gérée par JavaScript, grâce à l'instruction `addEventListener(evenement, fonction)` qui s'applique sur un objet : oui, JavaScript est un langage orienté objet.

Enfin, on a créé une variable `texte` qui va contenir toutes les tables à afficher : on stocke tout le travail fait par Javascript pour ne l'afficher qu'une seule fois avec `zoneEcrire.innerHTML = texte`.

Voici le résultat :

01 - Table de multiplication (complet)

Le 'maximum' et le 'nombre de table' sont saisis par l'utilisateur sur la page (plus d'accès au code)

Saisir nombre de tables : 10 Saisir le maximum : 13

1 x 1 = 1	2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5	6 x 1 = 6	7 x 1 = 7	8 x 1 = 8	9 x 1 = 9	10 x 1 = 10
1 x 2 = 2	2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10	6 x 2 = 12	7 x 2 = 14	8 x 2 = 16	9 x 2 = 18	10 x 2 = 20
1 x 3 = 3	2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15	6 x 3 = 18	7 x 3 = 21	8 x 3 = 24	9 x 3 = 27	10 x 3 = 30
1 x 4 = 4	2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20	6 x 4 = 24	7 x 4 = 28	8 x 4 = 32	9 x 4 = 36	10 x 4 = 40
1 x 5 = 5	2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25	6 x 5 = 30	7 x 5 = 35	8 x 5 = 40	9 x 5 = 45	10 x 5 = 50
1 x 6 = 6	2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30	6 x 6 = 36	7 x 6 = 42	8 x 6 = 48	9 x 6 = 54	10 x 6 = 60
1 x 7 = 7	2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35	6 x 7 = 42	7 x 7 = 49	8 x 7 = 56	9 x 7 = 63	10 x 7 = 70
1 x 8 = 8	2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40	6 x 8 = 48	7 x 8 = 56	8 x 8 = 64	9 x 8 = 72	10 x 8 = 80
1 x 9 = 9	2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45	6 x 9 = 54	7 x 9 = 63	8 x 9 = 72	9 x 9 = 81	10 x 9 = 90
1 x 10 = 10	2 x 10 = 20	3 x 10 = 30	4 x 10 = 40	5 x 10 = 50	6 x 10 = 60	7 x 10 = 70	8 x 10 = 80	9 x 10 = 90	10 x 10 = 100
1 x 11 = 11	2 x 11 = 22	3 x 11 = 33	4 x 11 = 44	5 x 11 = 55	6 x 11 = 66	7 x 11 = 77	8 x 11 = 88	9 x 11 = 99	10 x 11 = 110
1 x 12 = 12	2 x 12 = 24	3 x 12 = 36	4 x 12 = 48	5 x 12 = 60	6 x 12 = 72	7 x 12 = 84	8 x 12 = 96	9 x 12 = 108	10 x 12 = 120
1 x 13 = 13	2 x 13 = 26	3 x 13 = 39	4 x 13 = 52	5 x 13 = 65	6 x 13 = 78	7 x 13 = 91	8 x 13 = 104	9 x 13 = 117	10 x 13 = 130

essayez de mettre des valeurs négatives, de très grandes valeurs, ne rien mettre, mettre du texte (exemple : 'bob') et regardez comment se comporte le programme.

D À retenir

Cette exploration est terminée, il est probable que tout le code ne soit pas encore totalement compris et ce n'est pas l'objectif.

Il s'agit d'une première approche pour comprendre l'intérêt de la programmation.

D.1 Contenu global

Le fichier HTML contient beaucoup de codes mélangés entre-eux :

- HTML
- CSS
- JavaScript

D.1.1 Contenu HTML

On trouve les principales balises expliquées dans les supports de cours HTML, notamment `<HTML>`, `<head>` et `<body>`.

La balise `<div>` (comme division) est certainement une des plus importantes pour placer des cadres.

La balise `<input>` permet de placer des champs de saisies pour laisser l'utilisateur interagir avec la page.

Les balises `<style>` et `<script>` permettent de créer un bloc de code CSS ou JavaScript.

D.1.2 Contenu Javascript

Le fichier contient un code JavaScript complet qui utilise des concepts importants de la programmation :

Les **variables** qui sont des zones mémoires de stockage, comme *maximum*, *nbTable* ou *texte* mais aussi *zoneEcrire* ou *nombre*. Elles sont précédées du mot clé `let` pour leur déclaration.

On utilise les **itérations** grâce à l'instruction `for` : on peut ainsi répéter les instructions contenues dans le bloc (délimité par `{` et `}`) un nombre défini.

On utilise une **fonction** définie par le mot-clé `function nom()` pour exécuter une partie de code sur commande (par exemple, lors d'un changement d'un champ de saisie). Le bloc est également délimité par `{` et `}`. On appelle (exécute) la fonction en utilisant son nom.

D.1.3 Contenu CSS

La partie CSS utilise les ID HTML en commençant avec le symbole `#` ou les classes HTML avec le symbole `.` ce qui permet de personnaliser certaines balises et pas toutes.

On peut définir le type de cadre, l'épaisseur des bordures, la couleur d'écriture, etc.