

## TP2 : partie 2 - Vérification des TODO

### Table des matières

Étape 1 - Créer le plateau de jeu.....	2
1A - init .....	2
1B - afficherParametresAJour .....	2
1C - genererJeu.....	2
1D - genererJeuFacile .....	3
1E - genererJeuMoyen .....	3
1F - genererJeuDifficile.....	4
1G - diminuerLargeur .....	5
1H - augmenterLargeur .....	5
1I - diminuerHauteur .....	6
1J - augmenterHauteur .....	6
1K - diminuerMines .....	7
1L - augmenterMines .....	7
1M - verifierNbMines .....	7
1N - ajouterEcouteursAuxCellules.....	7
1O - creerObjetsCelulles.....	8
1P - placerMines.....	9
Étape 2 - Lancer le chronomètre.....	10
2A - lancerChronometre.....	10
Étape 3 - Clic gauche .....	10
3A - verifierCellule .....	10
3B - gameOver.....	11
3C - dévoilerCellule.....	11
Étape 4 - Clic droit .....	12
4A - placerDrapeau.....	12

## Étape 1 - Créer le plateau de jeu

### 1A - init

Il n'y a rien à vérifier immédiatement. Toutefois, lorsque vous allez vérifier les **TODO 1B à 1L** progressivement, si cliquer sur un **bouton** ne fonctionne pas, vous pouvez réviser vos **écouteurs d'événements** dans **init** !

### 1B - afficherParametresAJour

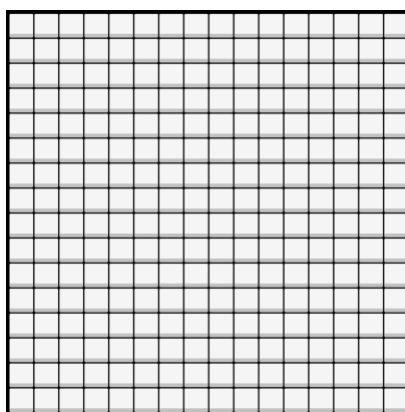
À partir de maintenant, lorsqu'on actualise la page, on devrait voir ces informations. (16, 16 et 40 pour le moment)



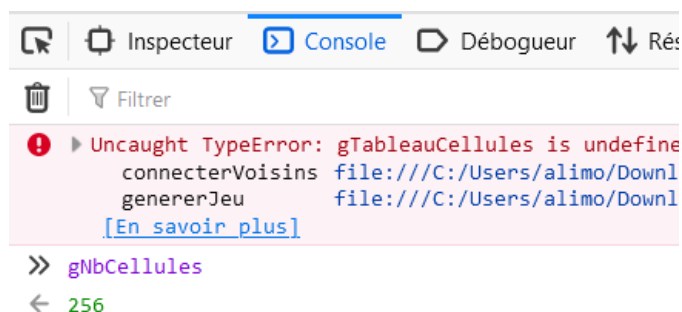
Autrement, on ne peut pas interagir avec la page présentement. (Ça ne fait rien)

### 1C - genererJeu

À partir de maintenant, on peut cliquer sur le bouton **Personnalisée** et cela fait apparaître une **grande grille** qui possède 16 colonnes et 16 rangées dans le bas de la page.



Si vous demandez à la **console** combien vaut la variable **gNbCellules**, elle est censée vous afficher **256**. (Il y aura une erreur dans la **console** pour le moment, mais c'est normal)



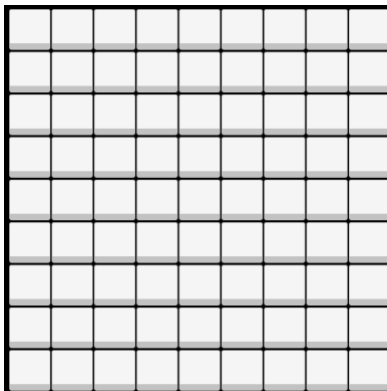
## 1D - genererJeuFacile

Désormais, on peut appuyer sur le bouton « **Facile** ».

Facile

Cela fait apparaître une **grille** avec 9

rangées et 9 colonnes.



Si on appuie d'abord sur **Facile**, puis qu'on appuie sur **Personnalisée**, cela ne fait rien. (Par contre, cela génère des erreurs dans la **console**)

Si on appuie d'abord sur **Personnalisée**, puis qu'on appuie sur **Facile**, la grille va changer de taille pour devenir plus petite. (Mais cela provoque également des erreurs dans la **console**)

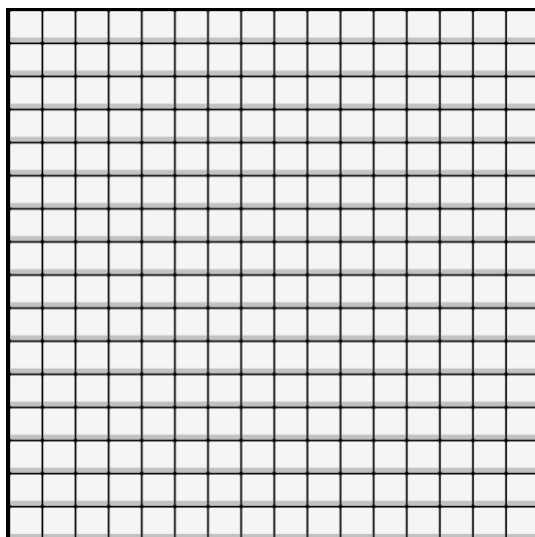
## 1E - genererJeuMoyen

Désormais, on peut appuyer sur le bouton « **Normal** ».

Normal

Cela fait apparaître une **grille** avec

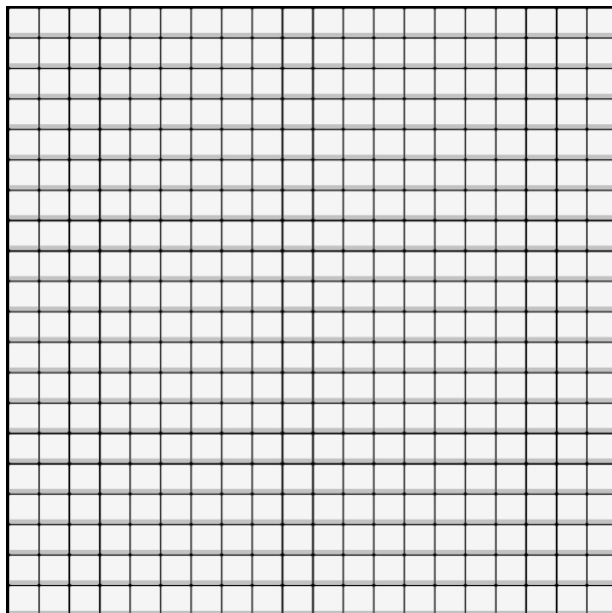
16 rangées et 16 colonnes. (Identique à celle personnalisée, pour le moment)



On peut alterner entre une **grille** « **Facile** » et une **grille** « **Normal** », mais encore une fois, cela génère des erreurs dans la **console**.

## 1F - genererJeuDifficile

Désormais, on peut appuyer sur le bouton « **Difficile** ».  Cela fait apparaître une **grille** avec 20 rangées et 20 colonnes.



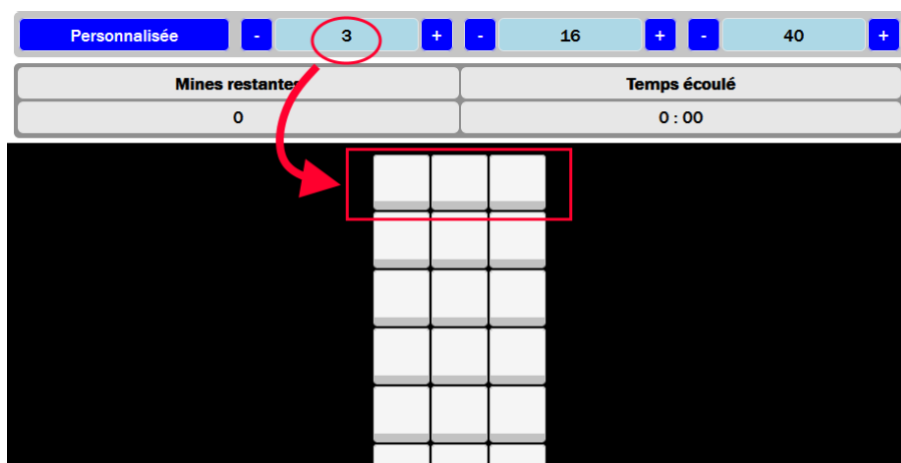
On peut alterner entre une **grille** « **Facile** », une **grille** « **Normal** » et une **grille** « **Difficile** », mais encore une fois, cela génère des erreurs dans la console.

## 1G - diminuerLargeur

Le **bouton** qui permet de diminuer la largeur devrait fonctionner. Par contre, il faut que descendre sous 1 soit impossible !



Lorsqu'on génère une **grille** personnalisée, le nombre de colonnes correspond à la largeur choisie.



## 1H - augmenterLargeur

On peut maintenant augmenter la largeur. (Il n'y a pas de limite maximale, on peut augmenter à l'infini)  
Lorsqu'on génère une grille personnalisée, le nombre de colonnes correspond encore bel et bien à la largeur choisie.

## 1I - diminuerHauteur

Il est possible de réduire la hauteur. (Elle ne peut pas aller plus bas que 1, par contre)

Hauteur

9

16

20

- 11 +

Quand on génère une **grille** personnalisée, la largeur (nombre de colonnes) et la hauteur (nombre de rangées) correspondent encore bel et bien aux valeurs choisies.

Largeur

9

16

20

- 5 +

Hauteur

9

16

20

- 6 +

estapes

0

Temps

0 : 0

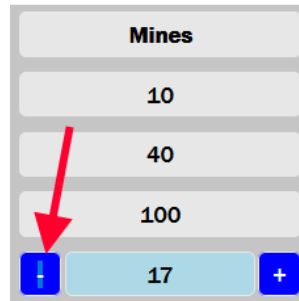
5x6 grid

## 1J - augmenterHauteur

On peut maintenant augmenter la hauteur. (Il n'y a pas de limite maximale, on peut augmenter à l'infini)  
Lorsqu'on génère une grille personnalisée, le nombre de rangées correspond encore à la hauteur choisie.

## 1K - diminuerMines

Il est possible de diminuer le nombre de mines. (Par contre, on ne peut pas aller sous 0)



## 1L - augmenterMines

On peut maintenant aussi augmenter le nombre de mines. Par contre, impossible de dépasser le nombre qui correspond à **largeur x hauteur**. (Par exemple, si **hauteur** vaut 4 et **largeur** vaut 4, impossible de mettre plus de 16 mines !)

Pour le moment, quand on diminue la largeur et la hauteur, cela n'impacte pas le **nombre de mines**. (Donc la règle citée plus haut peut être contournée indirectement...)

## 1M - verifierNbMines

Maintenant, quand on diminue la largeur ou la hauteur, le nombre de mines est réduit automatiquement si jamais la règle du nombre de mines est brisée.

## 1N - ajouterEcouteursAuxCellules

Pour le moment, on ne peut pas vérifier ce **TODO**. Toutefois, s'il ne se passe rien quand vous allez cliquer sur les cellules pour tester les **TODO 3A** et **4A**, c'est signe qu'il y a peut-être un problème ici !

## 10 - creerObjetsCellules

Générez une **grille** « Facile ». Dans la console, après avoir généré la grille, nous allons vérifier que `gTableauCellules` n'est pas vide. Il faut vérifier à l'index 0 et à l'index 9.



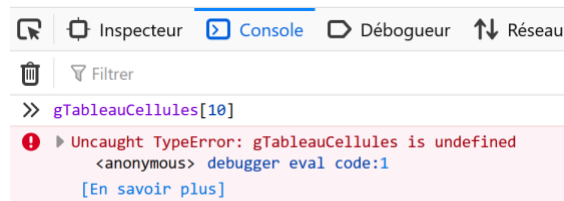
```

>> gTableauCellules[0]
← ▶ Object { voisins: (3) [...], nbMinesVoisines: 0, contientMine: false, contientDrapeau: false, dévoilee: false }

>> gTableauCellules[9]
← ▶ Object { voisins: (5) [...], nbMinesVoisines: 0, contientMine: false, contientDrapeau: false, dévoilee: false }

```

Si votre vérification ressemble à l'image ci-dessus (vérifiez que dans les deux cas, les **propriétés** sont exactement les mêmes pour vous !), c'est bon ! Par contre, si vous avez quelque chose comme ça pour l'index 0 ou l'index 9 :



```

>> gTableauCellules[10]
! ▶ Uncaught TypeError: gTableauCellules is undefined
  <anonymous> debugger eval code:1
  [En savoir plus]

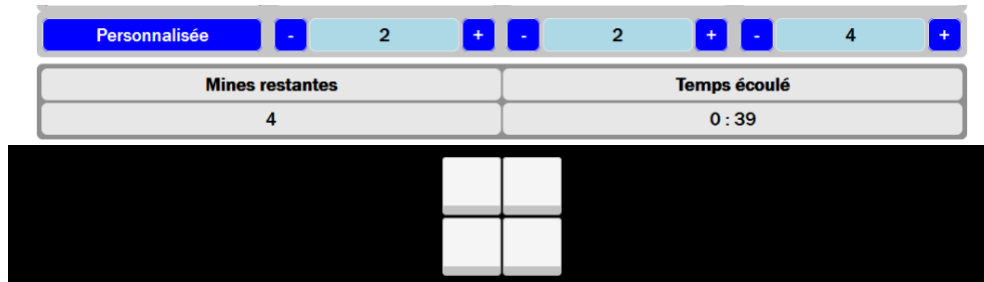
```

Alors il y a un problème !



## 1P - placerMines

Pour vérifier ce **TODO**, nous allons faire une opération très spécifique. D'abord, générez une **grille personnalisée** avec 2 de largeur et 2 de hauteur. (Et donc avec 4 mines)



Ensuite, nous allons faire les appels suivants dans la **console** :

```

>> gTableauCellules[0]
< ▶ Object { voisins: (3) [...], nbMinesVoisines: 3, contientMine: true, contientDrapeau: false, devoilee: false }
>> gTableauCellules[1]
< ▶ Object { voisins: (3) [...], nbMinesVoisines: 3, contientMine: true, contientDrapeau: false, devoilee: false }
>> gTableauCellules[2]
< ▶ Object { voisins: (3) [...], nbMinesVoisines: 3, contientMine: true, contientDrapeau: false, devoilee: false }
>> gTableauCellules[3]
< ▶ Object { voisins: (3) [...], nbMinesVoisines: 3, contientMine: true, contientDrapeau: false, devoilee: false }
>>

```

Vérifiez que « **contientMine** » est **true** pour TOUTES ces cellules !

## Étape 2 - Lancer le chronomètre

### 2A - lancerChronometre

Désormais, quand on génère une grille, le chronomètre fonctionne.



De plus, si on génère une nouvelle grille, le chronomètre retourne à 0 : 00 et redémarre.

## Étape 3 - Clic gauche

### 3A - verifierCellule

Pour tester ce **TODO**, il faut écrire, temporairement, les lignes de code suivantes :

- ◆ Dans **gameOver()** : **alert("BOOM") ;**
- ◆ Dans **devoilerCellule()** : **alert("FIOU") ;**

Vous pouvez maintenant essayer de faire des clics-gauches sur plein de cellules. La plupart du temps, vous avez une alerte "FIOU", et quelques rares fois, vous aurez l'alerte "BOOM".

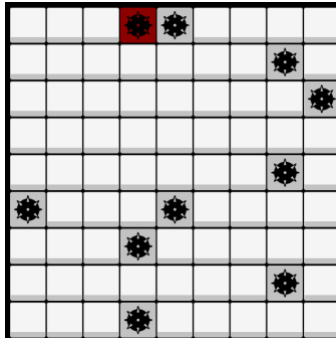
Si cliquer ne fait absolument rien, le problème peut être lié à **3A** ou à **1N**.

N'oubliez pas de retirer les alertes !



### 3B - gameOver

Générez une **grille Facile** dans la page. Ensuite, faites un **clic-gauche** sur toutes les cellules. Pour la plupart ça ne fera rien, mais vous devriez finir par tomber sur une cellule qui fait apparaître plein de **mines** dans la grille.



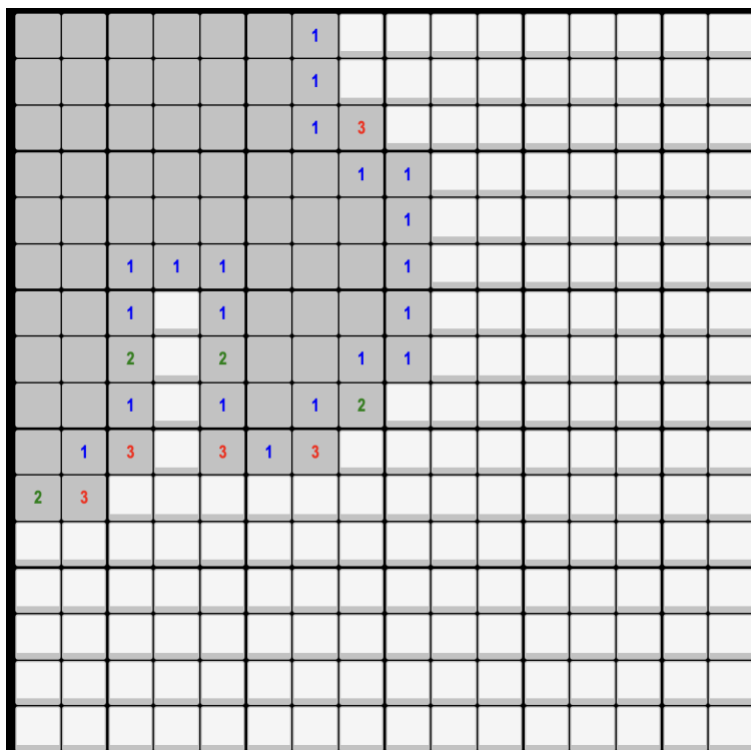
La cellule sur laquelle vous aurez appuyé sera une **mine rouge**, et toutes les autres seront **grises**. Notez également que le chronomètre s'arrête une fois que vous avez trouvé une **mine**.

Une fois qu'une mine a été découverte de cette manière, on peut cliquer sur les autres mines sans qu'il n'y ait de changement. (Si, en cliquant sur une autre mine, elle devient rouge elle aussi, il y a un problème !)

### 3C - dévoilerCellule

Générez une **grille facile**. Maintenant, faire un **clic-gauche** sur une cellule génère toujours une réaction ! Soit cela dévoile une **mine** et nous fait perdre, comme avant, ou encore cela dévoile un **nombre coloré**. (0, 1, 2, 3, etc.)

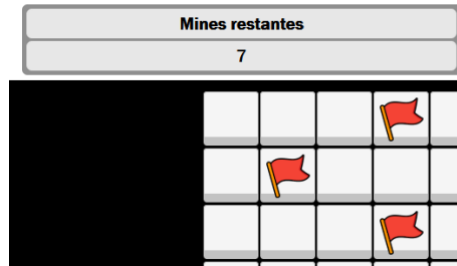
Notez que si on trouve un « 0 », cela dévoile une grande zone automatiquement.



## Étape 4 - Clic droit

### 4A - placerDrapeau

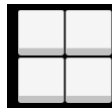
Générez une **grille** de votre choix. Le **clic-droit** fonctionne maintenant. Si on fait un **clic-droit** sur une cellule non dévoilée, un **drapeau** apparaît et le nombre affiché de « **Mines restantes** » diminue. On peut faire un **clic-droit** sur un drapeau déjà présent pour le retirer, ce qui augmente le nombre affiché de « **Mines restantes** ».



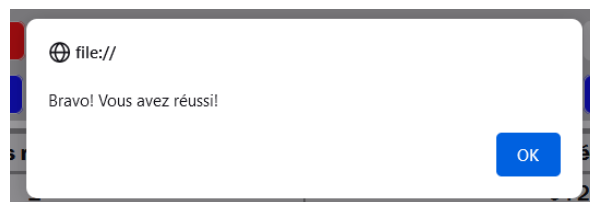
Notez que faire un **clic-gauche** sur un **drapeau** qu'on a placé ne doit rien faire. Par contre, si on retire le **drapeau**, on pourra faire un **clic-gauche** sur la cellule sans problème.

Si faire un **clic-droit** ne fait rien OU que vous avez l'impression que cela fait la même chose qu'un **clic-gauche**, il pourrait y avoir une erreur avec votre **TODO 1N**.

Finalement, générez une **grille personnalisée** avec 2 rangées, 2 colonnes et 4 mines.



Placez 4 drapeaux dans la grille. Vous devriez avoir une alerte qui vous indique que vous avez gagné. S'il n'y a pas de message, vous devrez vérifier votre code pour **4A**.



D'ailleurs, ce n'est pas votre seule victoire. Vous avez terminé le TP !

