

Préambule

Nous avons sélectionné la balise `<button>` pour représenter les éléments de notre calculette.

`<button>` a un comportement en ligne et permet de centrer facilement le contenu.

[en action](#)

Analyse



La phase la plus importante est l'analyse ! Elle consiste à trouver des affinités de comportement entre les différents éléments de notre calculette.



Par exemple, les boutons AC et DEL sont des éléments pour effacer les données.



Ces boutons peuvent être regroupés dans un bloc parent que l'on identifie par la classe "reset"¹.

définition des éléments	Définition d'un parent
<pre><button>AC</button> <button>DEL</button></pre>	<pre><div class="reset"> <button>AC</button> <button>DEL</button> </div></pre>

Pour dissocier dans ce groupe les différents éléments de même fonctionnalité on ajoute un data-set².

L'attribut data est utilisé pour stocker des données utilisables par le CSS et le code JS.



Voici le code HTML complet pour les deux boutons d'effacement

1. <div class="reset">
2. <button data-all-clear>AC</button>
3. <button data-delete>DEL</button>
4. </div>

¹ C'est mieux qu'un identifiant.

² <https://developer.mozilla.org/fr/docs/Web/API/HTMLElement/dataset>

Lig. 1 : Encapsulation pour regrouper les fonctionnalités d'effacement

Lig. 2-3 : Ajout d'un data-* pour caractériser la fonctionnalité d'effacement. *All-clear* efface toutes les valeurs. *Delete* efface un seul chiffre.

Passons à l'implémentation en CSS. Et continuons avec nos deux boutons d'effacement.



Le code indiquera tout d'abord la zone associée dans la grille principale³.

```
1. .reset {
2.     grid-area: reset;
3. }
```

Lig. 2 : On indique que la zone d'effacement est associée dans la grille principale à la zone "reset".



Le code définit ensuite la grille en elle-même.

```
1. .reset {
2.     display: grid;
3.     grid-auto-flow: column;
4. }
```

³ La grille principale est composée de plusieurs zones.

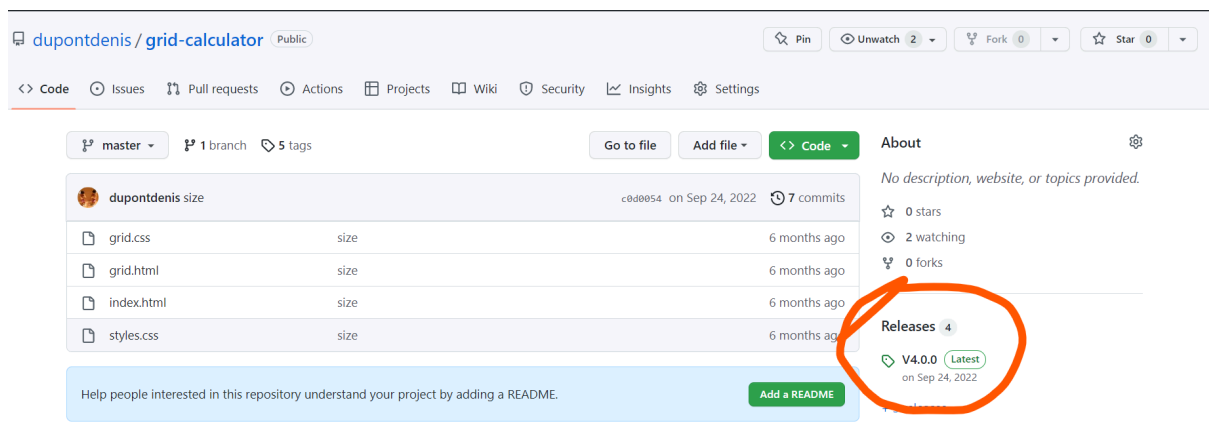
Lig.3 : On indique le flux en colonne.

Clonage de la correction sur GitHub

Vous allez découvrir au moins 5 versions différentes.

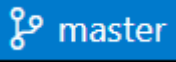
Dans github, vous pouvez voir les différentes versions sous

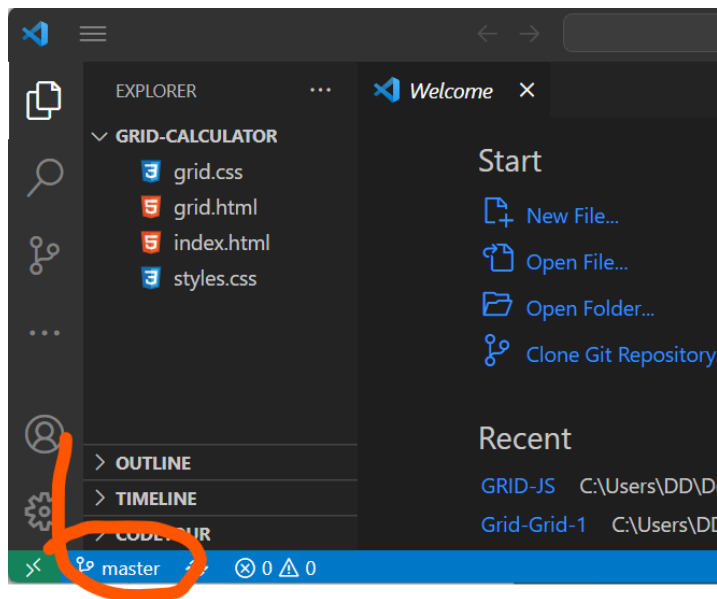
l'onglet **Releases** 4



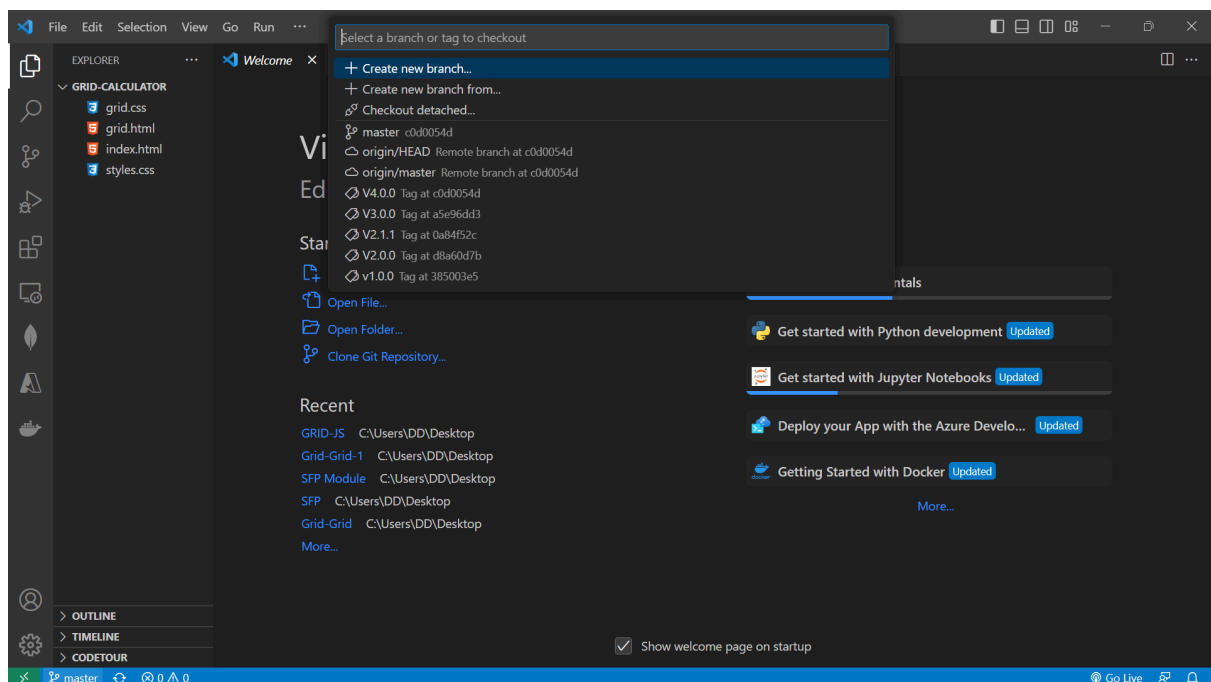
Commencez par clonez le code dans VisualStudio :

<https://github.com/dupontdenis/grid-calculator>

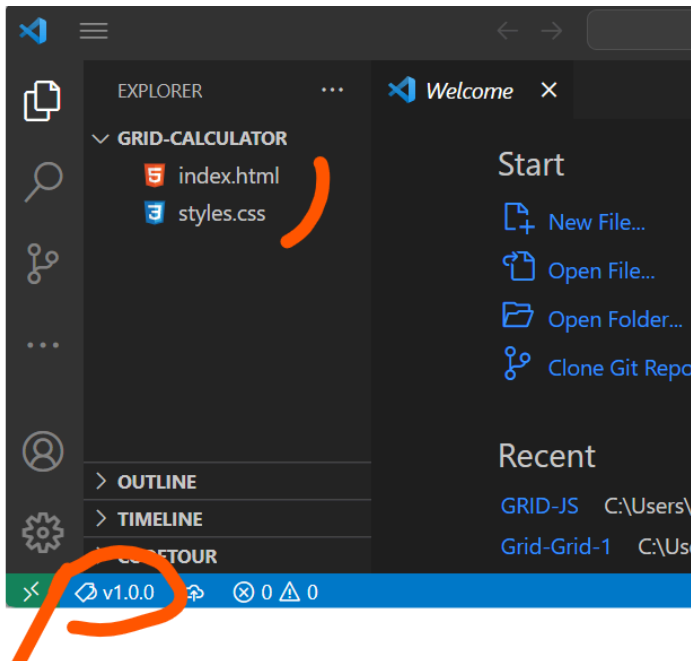
Cliquez sur l'onglet  master Pour voir les différentes versions



Vous voyez apparaître les versions sous le nom VX.X.X



En cliquant sur v1.0.0 vous voyez apparaître le code associé.



Cette première version définit une simple grille de 4 colonnes et 6 lignes.

1. .calculator-grid {
2. grid-template-columns: repeat(4, 100px);
3. grid-template-rows: minmax(120px, auto) repeat(5, 100px);
4. }

Toute la difficulté est reportée sur le flux du code HTML.

En effet, pour remplir la grille, le code HTML doit définir le flux avec un ordre précis.

Voici un extrait du code



```
<button data-operation>÷</button>
<button data-number>1</button>
<button data-number>2</button>
<button data-number>3</button>
```

```
<button data-operation>*</button>
```

```
<button data-number>4</button>
```

Ce code n'a rien de naturel, il vient compenser la définition basique de la grille.

Je vous laisse découvrir les différents codes HTML/CSS cherchant une solution "idéale".