

@import "style.less"

⚠ ce document est adapté pour être exporté avec Markdown Preview Enhanced ⚠

## Extensions VSCode

### Markdown all in one

- <https://marketplace.visualstudio.com/items?itemName=yzhang.markdown-all-in-one>
- TDM
  - `ctrl shift p : >Create table content`
  - va générer la table des matières (sans le titre)
  - Ajouter la balise `<!-- omit from toc -->` au titre pour ne pas l'afficher dans la table des matières
- De bon raccourcis claviers
- Prise en charge de Latex (formules); mermaid; et bien d'autres
- Prévisualisation `ctrl K, V` (**Sa ouvre juste le prévisualiseur de base de VSC**)
  - **Pas d'emoji de type** :code:
  - **2 ligne sans espace entre ==** à la suite du texte (fonctionnement normal de MD)
  - **UI simple mais efficace, on voit quel ligne est éditée**
- Export HTML
  - `ctrl shift p : >Print current document to HTML`
  - Passable, mais sans plus (manque un peu de colorisation comparé à la prévisualisation)
  - 1ère ligne du fichier markdown : `<!-- title: Guide Markdown -->` permet de définir le titre de la page HTML, sinon prend le 1er titre
  - **Ne supporte pas le front-matter (Markdown Preview Enhanced)**

### Markdown preview enhanced

- <https://marketplace.visualstudio.com/items?itemName=shd101wyy.markdown-preview-enhanced>
- Toutes les options sont bien expliquées dans la doc : <https://github.com/shd101wyy/markdown-preview-enhanced/tree/master/docs>
- Prévisualisation `ctrl K, V`
  - Rendu très bien
  - **2 ligne sans espace entre ==** comme si on avait un `<br>` (c'est une option) (fonctionnement normal de GitHub Flavored Markdown)
  - **gère les notes de bas de page** (`[^1]`)
  - **Checkbox modifiable dans la prévisu**
  - gère la visualisation Latex
  - Exports :
    - **Export "HTML", le rendu correspond à la prévisualisation** (thème de base)
    - Export "Puppeteer" est très puissant, mais demande de la configuration dans le **front-matter**
      - Gère les #page

- Malheureusement, le fichier CSS ne prend pas pour le "headerTemplate" et le "footerTemplate" (il faut configurer directement dans le front-formatter.
- les autres exports sont moyen (demande des installations supplémentaires, non testé)
- L'auto-génération de la TOC n'est pas fameuse // Mais en utilisant celle de Markdown ALL in One, le rendu est top

## Markdown PDF

- <https://marketplace.visualstudio.com/items?itemName=yzane.markdown-pdf>
- Pas testé plus que ça, car après configuration des précédentes extensions elle se suffisent. C'était l'extension que j'utilisais avant de tester les autres.
  - La configuration de l'entête et du pied de page se fait dans le fichier de configuration de l'extension (settings.json), ce qui n'est pas fameux pour des informations pour des fichiers spécifiques.
  - Pour afficher les #page c'est pas aisé

## Paste image

- <https://marketplace.visualstudio.com/items?itemName=mushan.vscode-paste-image>
- `ctrl shift v` : permet de coller une image dans le markdown (remapé, initialement `Ctrl Alt V`)

## Table des matières

- Extensions VSCode
  - [Markdown all in one](#)
  - [Markdown preview enhanced](#)
  - [Markdown PDF](#)
  - [Paste image](#)
- 1. Numéros de titres
  - 1.1. Titre 2
    - 1.1.1 Sous titre
- 2. Titre 3
- Styles
- Diagrammes
  - [PlantUML](#)
  - [Mermaid](#)
- [Nouvelle page \(HTML / PDF\)](#)
- [Markdown Preview Enhanced](#)
  - [CSS custom](#)
- Divers
- Image
- Notes de bas de page

## 1. Numéros de titres

Mise à jour des # titres auto avec `Markdown All in One` :

`ctrl shift p : >add/update section numbers` pour update  
`ctrl shift p : >remove section numbers` pour supprimer

===== 1.1. Titre 2 =====

===== 1.1.1 Sous titre =====

1.1.1.1 beaucoup de 1


===== 2. Titre 3 =====

===== Styles =====

- puces
- 😊
  - 😎
- 1. abc
- 2. def
  - ghi
    - jkl

citation

- ☐ tache
- ☒ Faite

style	code	raccourci (Markdown all in one)
<b>gras</b>	<code>**gras**</code>	<code>ctrl b</code>
<i>italique</i>	<code>*italique*</code>	<code>ctrl i</code>
<u>souligné</u>	<code>&lt;u&gt;souligné&lt;/u&gt;</code>	
<del>barré</del>	<code>~~barré~~</code>	<code>alt s</code>
code	<code>`code`</code>	
bloc de code (voir plus bas)	<code>```type code ```</code>	
lien	<code>[lien](...)</code>	<code>ctrl v</code> un lien en sélectionnant du texte
 image	<code>![image](...)</code>	
note de bas de page <sup>[^1]</sup> [^nom]	<code>[^1] [^nom]</code>	
$\varphi(n) \rArr x$	<code>\$...\$</code>	

**bloc de code :**

multiline

```
a = "hello world"
print(a)
```

```
var a = "hello world, avec une très longue ligne de code qui dépasse la largeur de
la fenêtre lors de l'exportation en PDF de ce document"
system.out.println(a)
```

Dans la preview, un slider s'affiche. Pour l'export avec Puppeteer ([Markdown Preview Enhanced](#)), le texte est tronqué si on ajoute pas cette option dans le css (ajout expliqué [ici](#)) :

```
@media print {
  pre[class*="language-"] {
    white-space: pre-wrap;
  }
}
```

le `@media print` permet de conserver le slider dans la preview`

les blocs de code pour les types `plantuml` et `mermaid` ne peuvent être affichés tel quel, ils sont convertis en diagrammes, voir [Diagrammes](#)

[^1]: note de bas de page #1

[^nom]: bloc de code

avec plusieurs lignes

---

## Diagrammes

---

Si les blocs de code prennent le type `plantuml` ou `mermaid`, ils sont automatiquement interprétés et affichés.

Du moins pour l'extension [Markdown Preview Enhanced](#).

---

## PlantUML

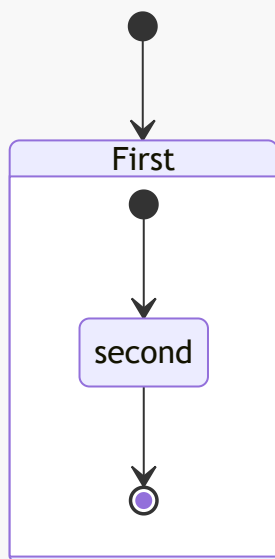
---

```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```

```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```

## Mermaid

```
stateDiagram
    [*] --> First
    state First {
        [*] --> second
        second --> [*]
    }
```



## Nouvelle page (HTML / PDF)

Faire un saut de page au format pdf ou impression navigateur :

- `<div style="page-break-after: always;"></div>`, // syntaxe pure HTML
- `<div class="page"></div>` // Simplification dans notre css, l'extension *markdown-pdf* la contient de base

// même si *page-break-after* est marqué comme *déprécié*, *puppeteer* ne gère pas le *break-after*

# Markdown Preview Enhanced

En 1ere ligne du fichier Markdown, possibilité d'utiliser le **front-matter** pour configurer le rendu HTML

```
---
config1: value1
config2: value2
---
```

Configuration possible (*testé*) :

config	description	valeur par défaut	type
title	titre de la page	nom du fichier sans l'extension	texte
print_background	utiliser le style du prévisualiseur	false	booléen
puppeteer	Configuration de l'export "Chrome (Puppeteer)"	-	objet
puppeteer.format	Taille de la feuille	A4	A4, A3, Letter, etc
puppeteer.margin	Marge de la page	-	objet
puppeteer.printBackground	Afficher les couleurs de background	false	booléen
puppeteer.displayHeaderFooter	Afficher les headers et footers	false	booléen
puppeteer.headerTemplate	Template HTML du header	-	HTML
puppeteer.footerTemplate	Template HTML du footer	-	HTML

## CSS custom

C:\Users\[utilisateur]\.mume\style.less sera la feuille de style par défaut, mais il est possible d'en ajouter de cette manière (ici relatif à ce fichier md) : `@import "css.less"`  
 .css est aussi valide, mais non visible dans la prévisu.

En cas de modification dans le fichier importé, il est nécessaire de recharger la prévisu pour prendre en compte les changements.

## Divers

Ligne séparatrice : au moins 3 \* ou - ou \_ sur une ligne

---

# Image

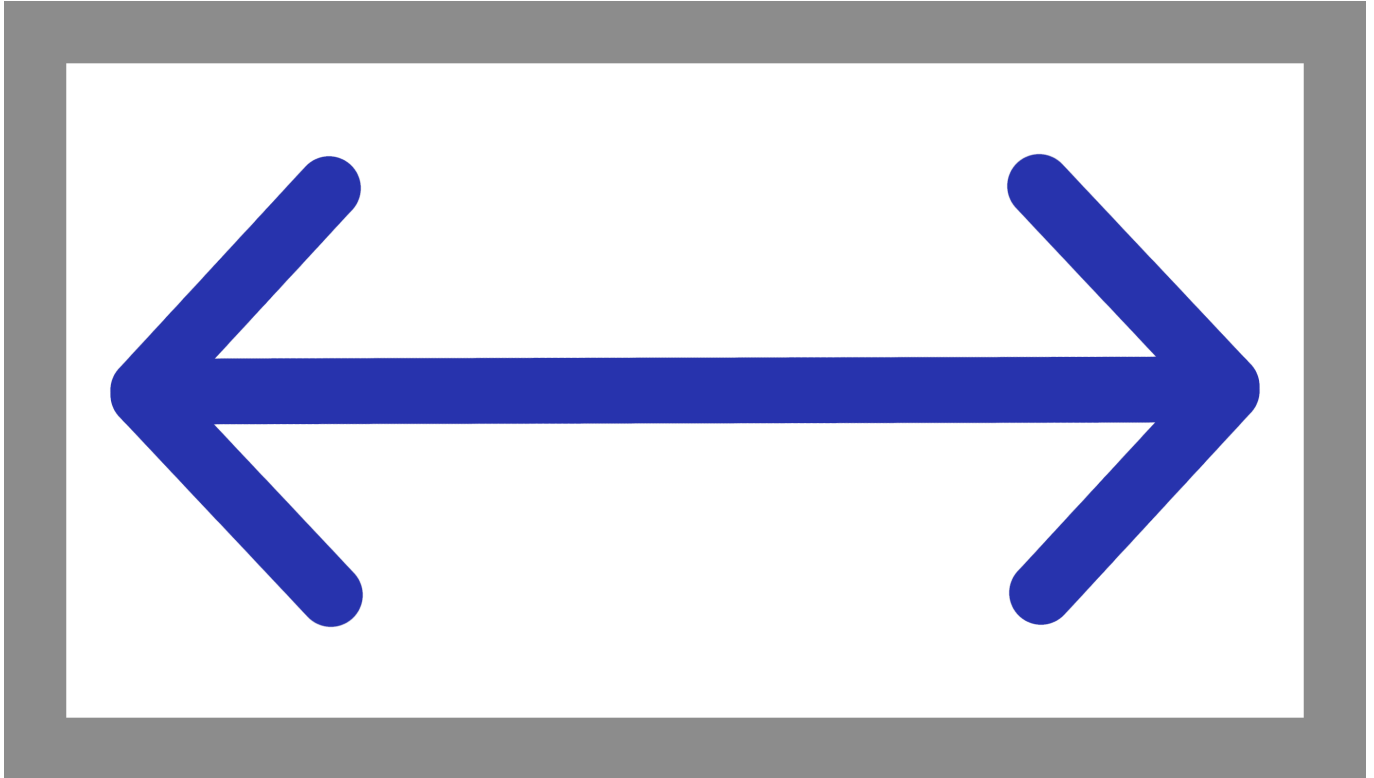
---

La taille des deux images ci-après est exactement la même !

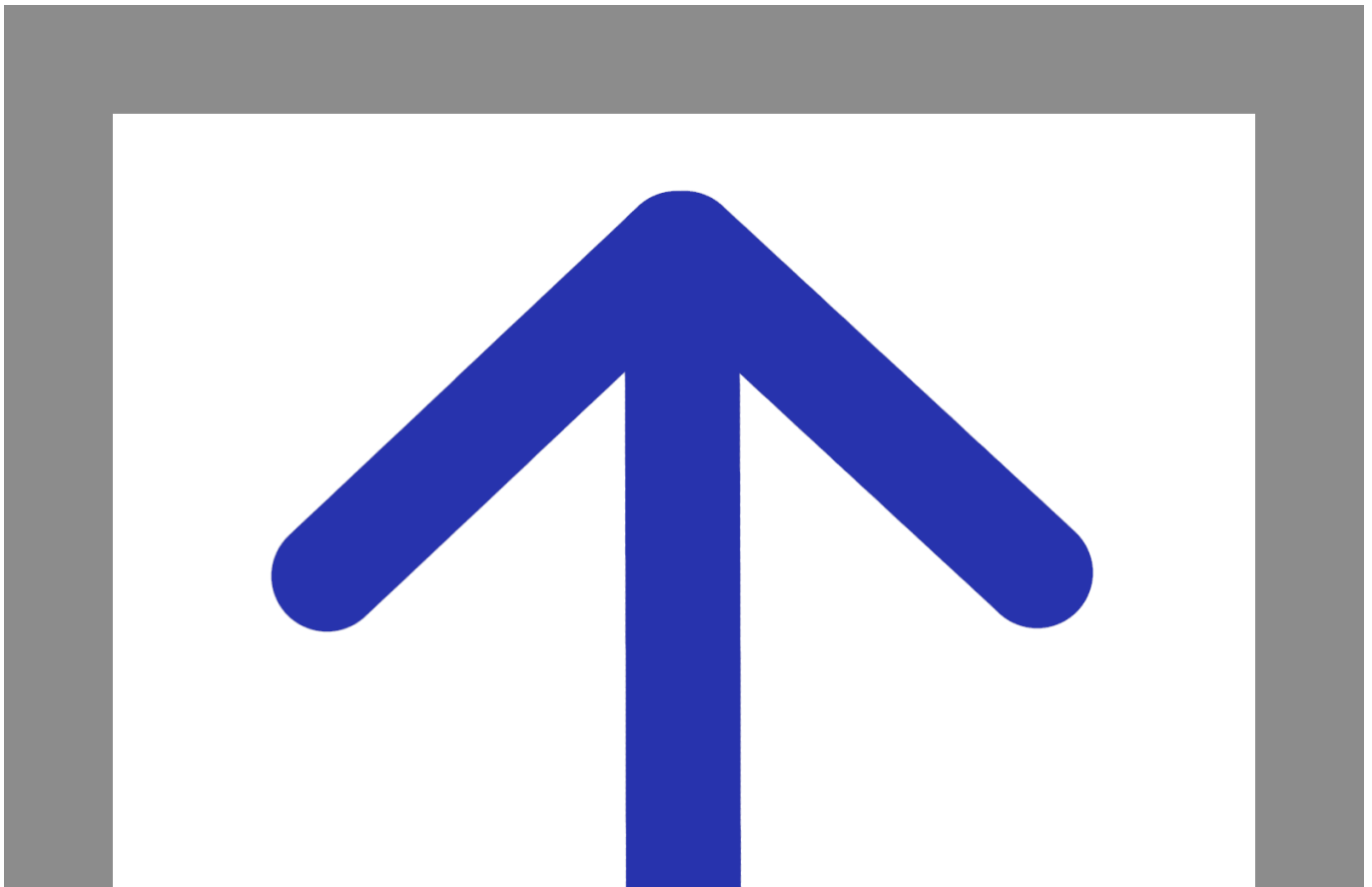
Sauf l'une est horizontale et l'autre verticale (respectivement 2800x1600 et 1600x2800).

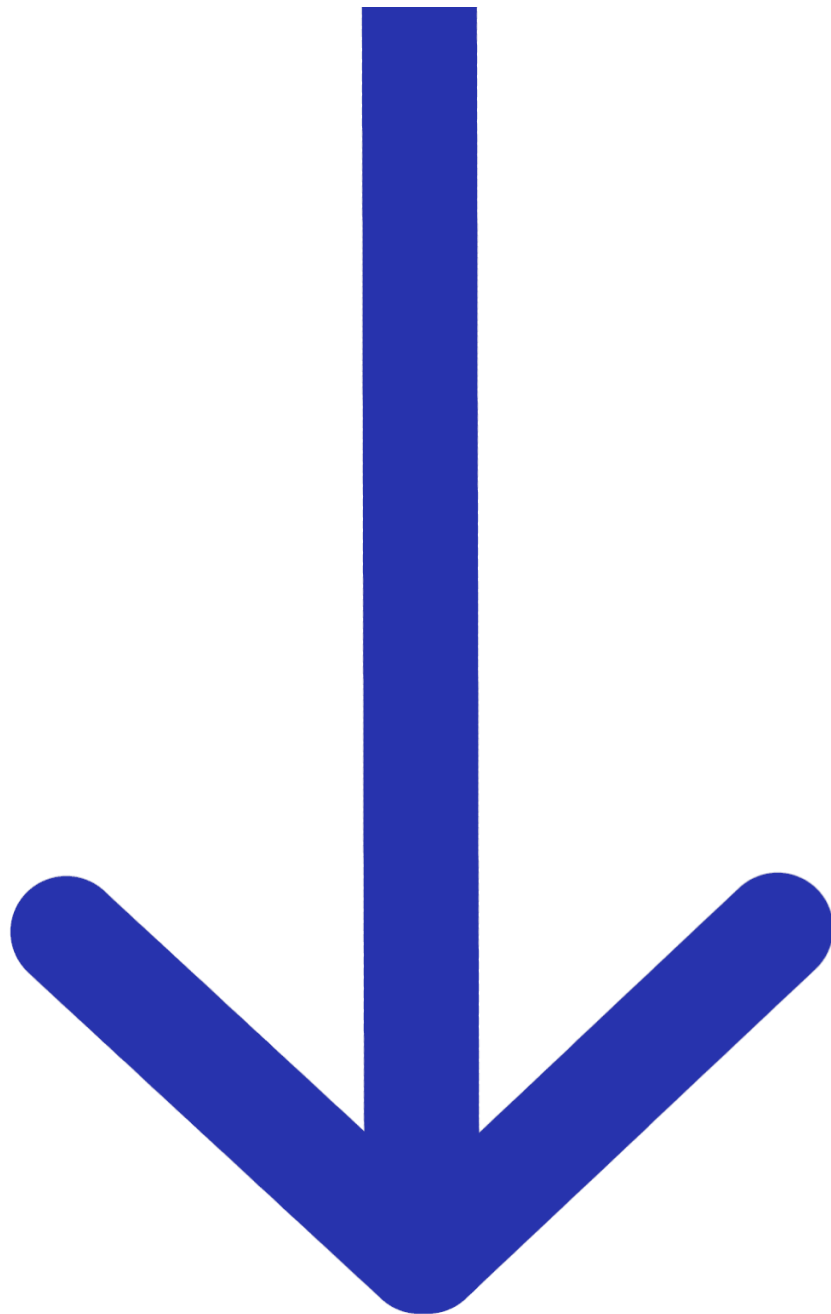
La taille de l'image est adaptée à la largeur de la fenêtre de prévisualisation, ou la taille de l'export PDF.

Une large image :

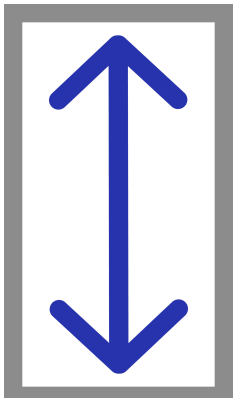


Une longue image :





Pour redimensionner une image, passer par html plutôt que markdown :





```

```

---

---

## Notes de bas de page

---

---

(Markdown Preview Enhanced)

Même si elle sont créé avant, elle seront affiché en fin de document

Nommé une note permet de la réutiliser plus facilement, mais son index sera visible dans l'output (HTML, prévisu)