

Résumé - Leçon 5 : Personnalisation

Objectifs

Utiliser le template avec beautitled, beautiframe et exercise-bank. Personnaliser l'apparence du document.

1. Structure du template

Import du template

```
#import "../shared/template.typ": *
```

```
#show: math-course-template.with(
  course: "IMA - EG - 2025-2026",
  license: "CC BY-NC-SA",
)
```

Éléments fournis par le template

Élément	Description
definition	Bloc de définition
theorem	Bloc de théorème
remark	Bloc de remarque
example	Bloc d'exemple
objectifs	Bloc d'objectifs
propriete	Bloc de propriété (français)
methode	Bloc de méthode (français)
formule	Bloc de formule (français)
exo	Exercice avec solution
plot	Graphique de fonction
tasks	Questions en colonnes

2. Environnements beautiframe

Utilisation de base

```
#definition(name: "Titre optionnel")[
  Contenu de la définition.
]

#theorem(name: "Pythagore")[
  Dans un triangle rectangle,  $a^2 + b^2 = c^2$ .
]

#remark[
  Une remarque sans titre.
]
```

Environnement de cours

Code Typst

```
#definition(name: "Discriminant") [
  Pour $a x^2 + b x + c = 0$,
  on pose $\Delta = b^2 - 4 a c$.
]
```

Rendu

Définition 1 — Discriminant

Pour $ax^2 + bx + c = 0$, on pose $\Delta = b^2 - 4ac$.

Personnalisation (à savoir situer)

Dans cette formation, `shared/template.typ` configure déjà `beautifram`. Dans un document d'exercice, utilisez d'abord `#definition`, `#theorem`, `#remark`, etc. Ne recopiez pas ce bloc dans votre fichier sauf si vous voulez vraiment modifier le template partagé.

```
#beautifram-setup(
  style: "modern", // ou "classic", "minimal"
  theorem-variant: "prominent",
  definition-variant: "standard",
  remark-variant: "minimal",
  accent-color: rgb("#00695c"),
)
```

3. Exercices avec exercise-bank

Structure de base

```
#exo(
  exercise: [
    Énoncé de l'exercice avec  $x^2 + 1 = 0$ .
  ],
  solution: [
    Pas de solution réelle car  $\Delta < 0$ .
  ],
)
```

Configurer puis réutiliser avec `.with()`

Code Typst

```
#let my-tasks = tasks.with(columns: 2,
  label-format: "a)")

#my-tasks [
  + Première réponse
  + Deuxième réponse
]
```

Rendu

a) Première
réponse

b) Deuxième
réponse

Configuration globale

```
#exo-setup(
  badge-style: "box", // Style du badge
  display: "both", // Afficher exercice et solution
  corr-display: "correction", // Affichage des corrections
  solution-label: "Solution", // Étiquette
  exercise-label: "Exercice", // Étiquette
  label-font-size: 11pt,
)
```

4. Tableaux de signes avec sign-table

Le helper sign-table est fourni par le template du cours pour générer des tableaux de signes cohérents.

```
#sign-table(
  factors: (
    (expr: $x - 2$,
      zeros: ((value: $2$, approx: 2)),
      signs: ("-", "+")),
    (expr: $x + 1$,
      zeros: ((value: $-1$, approx: -1)),
      signs: ("-", "+")),
  ),
  summary-label: $f(x)$,
  variation: true,
)
```

5. Personnalisation des titres avec beautitled

```
#beautitled-setup(
  style: "titled", // ou "schoolbook", "modern"
  chapter-prefix: "Chapitre",
  section-prefix: "Section",
)
```

6. Règles show / set (à reconnaître)

Les règles set changent un réglage pour la suite du document. Les règles show transforment automatiquement certains éléments. Pour le moment, l'objectif est surtout de les reconnaître dans le template, pas d'en écrire beaucoup.

Exemple utile et simple

```
#set page(margin: 2cm)
#set text(size: 11pt, lang: "fr")
```

Exemple avancé dans un template

```
// Tous les titres en bleu
#show heading: set text(fill: blue)
```

La syntaxe heading.where(...) existe, mais elle est plus avancée. Gardez-la pour une personnalisation de template, pas pour les premiers documents.

7. Couleurs

Couleurs prédéfinies

red, blue, green, yellow, orange, purple, black, white, gray

Couleurs personnalisées

```
rgb("#ff0000") // Rouge en hex
rgb(255, 0, 0) // Rouge en RGB
luma(50%)      // Gris 50%
blue.lighten(30%) // Bleu éclairci
red.darken(20%) // Rouge assombri
```

```
// Définir sa propre couleur
#let ma-couleur = rgb("#00695c")
```

Application aux éléments

```
#text(fill: blue)[Texte bleu]
#rect(fill: ma-couleur.lighten(70%), stroke: ma-couleur)
```

8. Changer la police

Typst peut utiliser les polices installées sur le système. Sur typst.app, utilisez les polices disponibles dans le projet.

Police globale du document

```
#set text(font: "Lora", size: 11pt)
```

Police locale

```
#text(font: "Fira Sans")[Texte en sans-serif]
```

Police avec fallback

```
#set text(font: ("Lora", "serif"))
```

Si « Lora » n'est pas disponible, Typst utilise la police serif par défaut.

Polices recommandées

Type	Polices
Serif	Lora, Libertinus Serif, EB Garamond, Linux Libertine
Sans-serif	Fira Sans, Inter, Source Sans Pro, Roboto
Monospace	Fira Code, JetBrains Mono, Inconsolata

Typst.app

Sur typst.app, utilisez de préférence les polices proposées dans l'interface du projet. Si une police n'est pas disponible, Typst utilise un fallback : le document compile, mais le rendu peut changer.

9. Checklist d'auto-vérification

- ☐ J'ai importé le template correctement
- ☐ Mes définitions/théorèmes utilisent `beautiframe`
- ☐ Mes exercices ont des solutions avec `exo`
- ☐ Mes graphiques illustrent les exercices
- ☐ Le document compile sans erreur

10. Liens utiles

- Universe `beautitiled` : <https://typst.app/universe/package/beautitiled>
- Universe `beautiframe` : <https://typst.app/universe/package/beautiframe>
- Universe `exercise-bank` : <https://typst.app/universe/package/exercise-bank>
- Fonction `text` : <https://typst.app/docs/reference/text/text>
- Show rules : <https://typst.app/docs/reference/styling/>