

Leçon 3

Lecture d'erreurs

Pourquoi cette leçon ?

Tout le monde fait des erreurs en codant.

La différence entre débutant et confirmé : **savoir les comprendre et les corriger rapidement.**

Note — Typst a un avantage majeur sur LaTeX : ses messages d'erreur sont clairs et utiles.

Anatomie d'un message d'erreur

```
error: unclosed delimiter
```

```
└─ document.typ:15:3
```

```
15 | $ x = frac(a, b
```

```
    |   ^ unclosed delimiter
```

```
help: try closing the delimiter with `)``
```

Élément	Signification
---------	---------------

Ligne 1	Type d'erreur
---------	---------------

Ligne 2	Fichier, ligne, colonne
---------	-------------------------

Code	Le problème avec <code>^</code> pointant l'erreur
------	---

<code>help:</code>	Suggestion de correction
--------------------	--------------------------

Astuce — Lisez toujours jusqu'au `help:` — la solution est souvent là.

Erreur : Délimiteurs non fermés

Code

```
// Erreurs  
$ x = 1  
#align(center)[texte  
frac(a, b  
#link("texte)
```

Rendu

```
// Corrections  
$ x = 1 $  
#align(center)[texte]  
frac(a, b)  
#link("texte")
```

C'est l'erreur la plus fréquente.

Erreur : Expressions mal évaluées

Le `#` n'évalue qu'UN seul élément.

Code

```
#let a = 4
#let b = 5

// Erreur :
Résultat : #a + #b
// Affiche : 4 + 5
```

Rendu

```
#let a = 4
#let b = 5

// Correction :
Résultat : #(a + b)
// Affiche : 9
```

Attention — Dès qu'il y a une opération, utilisez `#(expression)`

Erreur : Imports mal formés

Code

```
// Erreur : manque les :  
#import "@preview/pkg:0.1.0" fn
```

Rendu

```
// Correct  
#import "@preview/pkg:0.1.0": fn
```

Erreur : Cases sans virgules

Code

```
// Erreur
$ f(x) = cases(
  x^2 & "si" x >= 0
  -x & "sinon"
) $
```

Rendu

```
// Correct
$ f(x) = cases(
  x^2 & "si" x >= 0,
  -x & "sinon",
) $
```

Erreur math : variables collées

En mode math, `xy` est lu comme un seul nom.

Code

```
// Rendu trompeur  
$ A = xy $
```

Rendu

```
// Correct : produit de x par y  
$ A = x y $
```

Attention — Pour multiplier deux variables, laissez un espace : `x y`, pas `xy`.

Méthode de débogage

1. **Lire** le message en entier (jusqu'au `help`)
2. **Localiser** l'erreur (aller à la ligne indiquée)
3. **Identifier** le type (délimiteur ? expression ? import ?)
4. **Corriger** une seule chose à la fois
5. **Recompiler** pour vérifier

Attention — Corrigez toujours la **première** erreur d'abord. Les suivantes disparaîtront peut-être.

Technique du commentaire

Si vous ne trouvez pas l'erreur, commentez du code suspect :

```
// Cette ligne pose peut-être problème :  
// #ma-fonction-suspecte()  
  
Le reste du document...
```

Technique « divide and conquer » : commentez la moitié, puis affinez.

Dictionnaire des erreurs

Message	Signification
unclosed delimiter	Oublié de fermer \$, [, (
unexpected token	Caractère inattendu
unknown variable	Variable non définie
cannot add	Opération impossible
unknown package	Package inexistant

Les IA trouvent très rapidement l'erreur.

Cas typst.app fréquents

Cas

Package not found

Import path broken

Function called in markup mode

PDF inchangé

Réflexe

Vérifier nom et version sur Universe

Vérifier `../`, dossier, majuscules

Ajouter `#` devant la fonction

S'assurer de compiler le bon fichier

Fonction appelée en markup

Une fonction en mode markup doit être précédée de #.

Code

```
// Incorrect  
v(0.5cm)  
align(center)[Texte]
```

Rendu

```
// Correct  
#v(0.5cm)  
#align(center)[Texte]
```

Attention — Sans #, `v(0.5cm)` s'affiche en toutes lettres au lieu d'agir.

Erreurs dans les formules

À vérifier dans `$ ... $` :

- parenthèses de `frac(...)`, `sqrt(...)`, `cases(...)`
- virgules entre les cas de `cases`
- variables multipliées séparées par un espace : `x y`, pas `xy`
- texte entre guillemets : `"si"`
- `\` en fin de ligne dans les alignements

Checklist

Quand ça ne compile pas :

- ☐ Tous les \$ sont-ils fermés ?
- ☐ Tous les crochets sont-ils appariés ?
- ☐ Toutes les parenthèses sont-elles fermées ?
- ☐ Les imports sont-ils bien formés ?
- ☐ Les expressions utilisent-elles des parenthèses ?
- ☐ Les cases ont-ils des virgules ?

Leçon suivante : images et packages