BNF

```
; ============================================================
; SlideDeckML — Concrete Syntax (BNF-like) with @deck/@slide
; ============================================================

<deck> ::= <deck_header>? <slide> (<slide_sep> <slide>)*

<slide_sep> ::= NL '---' NL

; --------------------------
; DECK HEADER (metadata globale)
; --------------------------
<deck_header> ::= '@deck' NL <deck_meta_line>* '@enddeck' NL

<deck_meta_line> ::=
      'title:' WS <string> NL
    | 'author:' WS <string> NL
    | 'offlineExport:' WS <bool> NL
    | 'template:' WS <string> NL
    | 'cssPath:' WS <path> NL
    | 'assetsDir:' WS <path> NL

; --------------------------
; SLIDE
; --------------------------
<slide> ::= <slide_header>? <slide_body>

<slide_header> ::= '@slide' NL <slide_meta_line>* '@endslide' NL

<slide_meta_line> ::=
      'id:' WS <id> NL
    | 'layout:' WS <layout_name> NL
    | 'transition:' WS <transition_type> (WS <int>)? NL
    | 'notes:' WS '|' NL <indented_lines>

<layout_name> ::= <id>        ; ex: TITLE_CONTENT, TWO_COLUMNS, FREE (ou nom
custom)
<transition_type> ::= 'NONE' | 'FADE' | 'SLIDE' | 'ZOOM'

<slide_body> ::= (<block> NL?)*

; --------------------------
; BLOCKS (contenu)
```

```
; Chaque block peut être "révélé" à un step avec le préfixe @n
; --------------------------
<block> ::= <reveal_prefix>? (
      <heading>
    | <paragraph>
    | <list>
    | <image>
    | <video>
    | <code_block>
    | <equation_block>
    | <zone_block>
    | <slot_block>
    | <live_code_block>
    | <interactive_block>
    | <step_action>
)

<reveal_prefix> ::= '@' <int> WS   ; apparaît à l'étape <int>

; --------------------------
; TEXTE / LISTES
; --------------------------
<heading> ::= ('#' | '##' | '###') WS <textline>

<paragraph> ::= <textline> (NL <textline>)*

<list> ::= <unordered_list> | <ordered_list>
<unordered_list> ::= ('-' WS <textline> NL)+
<ordered_list> ::= (<int> '.' WS <textline> NL)+

; --------------------------
; MEDIA
; --------------------------
<image> ::= '!' '[' <alt_text> ']' '(' <path> ')' <image_opts>?
<image_opts> ::= WS '{' 'fit=' <image_fit> '}'
<image_fit> ::= 'CONTAIN' | 'COVER' | 'STRETCH'

<video> ::= '!video' '(' <path> ')' <video_opts>?
<video_opts> ::= WS '{' <video_flag> (WS <video_flag>)* '}'
<video_flag> ::= 'autoplay' | 'loop' | 'muted' | 'controls'

; --------------------------
; CODE + REVEAL LIGNES
; {1|2-3|4} => étapes successives de révélation des lignes
; --------------------------
<code_block> ::= '```' <lang> <code_reveal_spec>? NL <code_lines> '```'
```

```
<lang> ::= <id>

<code_reveal_spec> ::= WS '{' <reveal_range> ('|' <reveal_range>)* '}'
<reveal_range> ::= <int> | <int> '-' <int>

<code_lines> ::= (<textline> NL)*

; --------------------------
; EQUATIONS (extension)
; --------------------------
<equation_block> ::= <equation_inline> | <equation_display>
<equation_inline> ::= '$' <latex> '$'
<equation_display> ::= '$$' NL <latex_lines> '$$'

<latex_lines> ::= (<textline> NL)*

; --------------------------
; ZONES (layout zones explicites)
; Exemple:
; ::zone LEFT::
;   @1 - item
; ::endzone::
; --------------------------
<zone_block> ::= '::zone' WS <area> '::' NL <slide_body> '::endzone::'
<area> ::= 'HEADER' | 'CONTENT' | 'LEFT' | 'RIGHT' | 'FOOTER' | 'CANVAS'

; --------------------------
; SLOTS + SWITCH (remplacement contrôlé)
; Un slot définit des variantes affichées selon le step.
; Exemple:
; :::slot viz
;   @1 ![a](a.png)
;   @2 ![b](b.png)
; :::
; --------------------------
<slot_block> ::= ':::slot' WS <slot_id> NL <slot_entry>+ ':::'
<slot_id> ::= <id>

<slot_entry> ::= '@' <int> WS <slot_variant_block> NL?
<slot_variant_block> ::= <image> | <video> | <paragraph> | <zone_block>

; --------------------------
; LIVE PROGRAMMING (extension)
; --------------------------
<live_code_block> ::= '```live' WS <lang> <live_opts>? NL <code_lines> '````'
```

```
<live_opts> ::= WS '{' <live_opt> (WS <live_opt>)* '}'

<live_opt> ::=
      'editable=' <bool>
    | 'runtime=' <runtime_kind>
    | 'endpoint=' <string>
    | 'timeoutMs=' <int>
    | 'showOutput=' <bool>
    | 'plot=' <chart_type>

<runtime_kind> ::= 'LOCAL' | 'REMOTE'
<chart_type> ::= 'BAR' | 'LINE' | 'PIE' | 'WORD_CLOUD'

; --------------------------
; INTERACTIVE QUIZ / POLL (extension)
; --------------------------
<interactive_block> ::= ':::interactive' WS <interactive_kind> WS
<interactive_id> NL
                        <interactive_header>
                        <interactive_body>
                        ':::'

<interactive_kind> ::= 'QUIZ' | 'POLL'
<interactive_id> ::= <id>

<interactive_header> ::= ('question:' WS <textline> NL)
                         ('results:' WS <results_visibility> NL)?
                         ('view:' WS <result_view_type> NL)?
                         ('allowMultiple:' WS <bool> NL)?

<results_visibility> ::= 'ALWAYS' | 'ON_DEMAND'
<result_view_type> ::= 'CHART' | 'WORD_CLOUD' | 'SUMMARY'

<interactive_body> ::= <choices_block> | <short_answer_block>

<choices_block> ::= 'choices:' NL (<choice_line>)+
<choice_line> ::= '-' WS <textline> (<choice_correct>)? NL
<choice_correct> ::= WS '[correct]'

<short_answer_block> ::= 'short-answer:' NL <indented_lines>

; --------------------------
; STEP ACTIONS (extension)
; Used for ON_DEMAND results
; --------------------------
<step_action> ::= '@step' WS <int> ':' WS <action_call>
```

```
<action_call> ::= 'showResults(' <interactive_id> ')'

; --------------------------
; LEXICAL
; --------------------------
<bool> ::= 'true' | 'false'
<id> ::= <letter> (<letter> | <digit> | '_' | '-')*
<int> ::= <digit>+
<path> ::= <string>
<string> ::= '"' <chars>* '"' | <bare_string>
<bare_string> ::= <chars_no_nl>+
<alt_text> ::= <chars_no_bracket>*
<textline> ::= <chars_no_nl>+
<indented_lines> ::= (WS WS <textline> NL)*

<letter> ::= 'A'..'Z' | 'a'..'z'
<digit> ::= '0'..'9'
<latex> ::= <chars_no_dollar>+
<chars>* ::= (any char)*
WS ::= ' '+
NL ::= '\n'
```