Guide: Progressive Outline

1 Function documentation

This section details all the parameters available for the `progressive-outline` function.

Option	Type	Effect & Expected Values
`level-X-mode`	string	Defines the visibility of level X (1, 2, or 3). Values: `"all"`, `"current"`, `"current-parent"`, `"none"`.
`text-styles`	dict	Dictionary of Typst `text(...)` styles for each level. Structure: `(level-X: (active: (...), inactive: (...)))`.
`spacing`	dict	Controls vertical space (`v-between-X-Y`) and horizontal indentation (`indent-X`) between elements.
`show-numbering`	bool	Enables or disables the display of heading numbering.
`numbering-format`	str | func	Typst numbering format (e.g., `"1.1"`) or custom function `(..n) => ...`.

2 Visibility

This section covers the `level-X-mode` parameters.

2.1 The ‘current-parent’ mode

The `current-parent` mode is the most powerful: it only displays the “siblings” of the current element. This allows you to see the plan of the current section without being distracted by other chapters.

```
progressive-outline(
  level-1-mode: 'all',
  level-2-mode: 'current-parent'
)
```

Visibility Demonstration H2

2.2 Isolation via ‘current’ mode

If you want an ultra-minimalist rendering, the `current` mode hides everything except the exact entry where you are located.

```
progressive-outline(
  level-1-mode: 'current',
  level-2-mode: 'none'
)
```

Isolated Visibility Demonstration

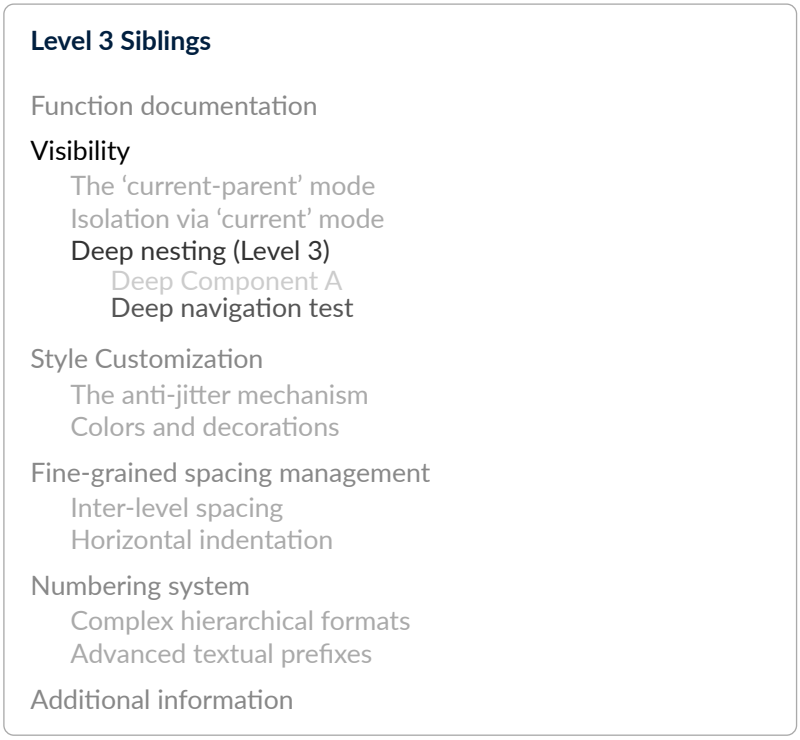
2.3 Deep nesting (Level 3)

For complex structures, you can enable Level 3. Using `current-parent` will show siblings at the current depth.

2.3.1 Deep Component A

2.3.2 Deep navigation test

```
progressive-outline(
  level-2-mode: 'all',
  level-3-mode: 'current-parent'
)
```



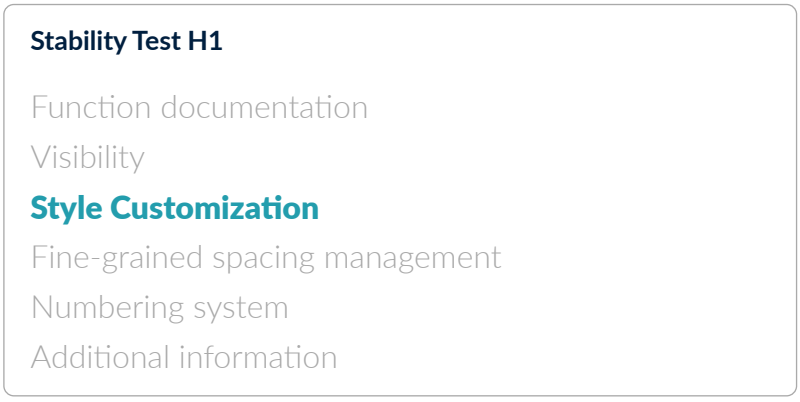
3 Style Customization

The function allows you to modify the appearance of headings based on their state (active/inactive).

3.1 The anti-jitter mechanism

Anti-jitter ensures that switching from a thin font to a bold one doesn’t move the text. We use a ghost box to reserve the maximum space required.

```
text-styles: (
  level-1: (
    active: (weight: 'black', fill:
eastern, size: 1.2em),
    inactive: (weight: 'light', fill:
gray, size: 1.2em)
  )
)
```



3.2 Colors and decorations

Each level can have its own rules for colors, italics, or bold.

```
text-styles: (
  level-2: (
    active: (style: 'italic', fill: blue,
weight: 'bold'),
    inactive: (fill: luma(200))
  )
)
```

Creative Style H2

4 Fine-grained spacing management

The `spacing` dictionary sculpts the rhythm.

4.1 Inter-level spacing

You can define the exact space between an H1 heading and an H2 heading, or between two headings of the same level.

```
spacing: (
  v-between-1-1: 2em,
  v-between-1-2: 1.2em,
  v-between-2-2: 0.8em,
  v-between-2-1: 1.5em
)
```

Airy Vertical Rhythm

4.2 Horizontal indentation

Indentation defines the offset to the right for each depth level.

```
spacing: (
  indent-2: 3em,
  indent-3: 6em
)
```

Marked Indentation

5 Numbering system

The function relies on Typst’s native engine.

5.1 Complex hierarchical formats

The `numbering-format` parameter accepts all standard Typst models (1, a, i, I, A).

```
show-numbering: true,
numbering-format: 'I.a.1. '
```

Legal Format

5.2 Advanced textual prefixes

To use long words like “Chapter” without errors, pass a function. This prevents Typst from interpreting letters like ‘a’ or ‘i’ as numbering models.

```
show-numbering: true,
numbering-format: (..n) => 'Chapter ' +
numbering('1', ..n) + ' : '
```

Secure 'Chapter' Prefix

6 Additional information

It is optimized to work within presentation themes (like `progressive-outline`), but can be used in any standard Typst document.