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	Styles passed to `#text` (fill, weight, etc.). You can also use a float (e.g., `0.5`) as a shortcut to inherit the active style with that opacity.
`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) => ...`.
`match-page-only`	bool	If true, considers a heading active if it is on the same page, regardless of its Y position. Useful for sidebars.
`filter`	func	A callback function `(heading) => bool` to programmatically include or exclude headings.
`marker`	content | dict | func	Content displayed before the item. Can be static, a dict by state, or a function `(state, level) => content`.
`clickable`	bool	Enables clickable links on headings. Defaults to `true`.

2 Navigation & Interactivity

By default, the outline is interactive: clicking on a section title navigates directly to the corresponding slide in the PDF.

```
progressive-outline()
```

In some print-focused or strict layout scenarios, you might want to disable this interactivity.

```
progressive-outline(
  clickable: false
)
```

3 Visibility

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

3.1 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

Function documentation

Navigation & Interactivity

Visibility

- **The ‘current-parent’ mode**
- Isolation via ‘current’ mode
- Deep nesting (Level 3)

Style Customization

Customizable Markers

The anti-jitter mechanism

Fine-grained spacing management

Numbering system

Filtering Content

Advanced Behavior

Additional information

3.2 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

Visibility

3.3 Deep nesting (Level 3)

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

3.3.1 Deep Component A

3.3.2 Deep navigation test

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


4 Style Customization

The function allows you to modify the appearance of headings based on their state (**completed**, **active**, or **inactive**).

4.1 The 3-state system

By default, headings can be in one of three states:

- **completed**: The heading has already been passed.
- **active**: This is the current heading.
- **inactive**: The heading is yet to come.

```
text-styles: (
  level-1: (
    active: (fill: eastern, weight:
'bold'),
    completed: (fill: gray.lighten(50%)),
    inactive: (fill: black)
  )
)
```

Past, Present, Future

Function documentation
Navigation & Interactivity
Visibility
Style Customization
Customizable Markers
The anti-jitter mechanism
Fine-grained spacing management
Numbering system
Filtering Content
Advanced Behavior
Additional information

4.2 Advanced Opacity & Inheritance

Instead of redefining the full style for `inactive` or `completed` states, you can use smart inheritance to adapt the `active` style.

4.2.1 The Float Shortcut (Clone & Fade)

Pass a number (0.0 to 1.0) to automatically clone the active style and apply transparency. `0.2` means 20% opacity (very faint), `1.0` means fully opaque.

```
text-styles: (
  level-1: (
    active: (fill: red, weight: 'black'),
    inactive: 0.2,   // Future: very faint
(20%)
    completed: 0.5   // Past: semi-
transparent (50%)
  )
)
```

Auto-Fade Shortcut

Function documentation
Navigation & Interactivity
Visibility
Style Customization
Customizable Markers
The anti-jitter mechanism
Fine-grained spacing management
Numbering system
Filtering Content
Advanced Behavior
Additional information

4.2.2 Partial Inheritance (Mix & Match)

You can also use a dictionary with an `opacity` key. This allows you to inherit the active color (faded) while overriding other properties (like weight).

```
text-styles: (
  level-1: (
    active: (fill: blue, weight: 'black'),
    inactive: (
      opacity: 0.5,      // 50% of active
color
      weight: 'regular'  // But force
regular weight
    )
  )
)
```

Fade + Weight Change

5 Customizable Markers

You can add visual indicators (icons, arrows, etc.) before each item using the `marker` parameter.

5.1 Static Marker

The simplest usage is to pass a single content element (like a symbol) that will be used for all items.

```
progressive-outline(
  marker: sym.triangle.filled.small
)
```

Static Symbol

5.2 State-based Markers (Dictionary)

You can define different markers for `active`, `inactive`, and `completed` states using a dictionary.

```
progressive-outline(
  marker: (
    active: sym.arrow.r,
    completed: sym.checkmark,
    inactive: sym.circle.small
  )
)
```

State Indicators

- ✓ Function documentation
- ✓ Navigation & Interactivity
- ✓ Visibility
- ✓ Style Customization
- → **Customizable Markers**
  - **The anti-jitter mechanism**
  - **Fine-grained spacing management**
  - **Numbering system**
  - **Filtering Content**
  - **Advanced Behavior**
  - **Additional information**

5.3 Dynamic Markers (Function)

For total control, pass a function `(state, level) => content`. This allows you to vary markers based on depth level and status.

```
progressive-outline(
  marker: (state, level) => {
    if level == 1 { sym.star.filled }
    else if state ==
'active' { sym.arrow.r }
    else { sym.circle.filled.tiny }
  }
)
```

Advanced Logic

★ Function documentation

★ Navigation & Interactivity

★ Visibility
- The 'current-parent' mode
- Isolation via 'current' mode
- Deep nesting (Level 3)

★ Style Customization
- The 3-state system
- Advanced Opacity & Inheritance

★ **Customizable Markers**
- Static Marker
- State-based Markers (Dictionary)

→ **Dynamic Markers (Function)**
- Marker Alignment

★ **The anti-jitter mechanism**
- Colors and decorations

★ **Fine-grained spacing management**
- Inter-level spacing
- Horizontal indentation

★ **Numbering system**
- Complex hierarchical formats
- Advanced textual prefixes

★ **Filtering Content**
- Label-based filtering
- Logic-based filtering
- Recursive filtering

★ **Advanced Behavior**
- Page-based matching

★ **Additional information**

5.4 Marker Alignment

Use the `spacing` parameter to fine-tune layout:

- `marker-gap`: Space between marker and text (default `0.5em`).
- `marker-width`: Fixed width for the marker container (useful for alignment).

```
progressive-outline(
  marker: (active: sym.arrow.r),
  spacing: (
    marker-gap: 1em,
    marker-width: 1.5em
  )
)
```

Aligned Markers

Function documentation
Navigation & Interactivity
Visibility
Style Customization
→ **Customizable Markers**
The anti-jitter mechanism
Fine-grained spacing management
Numbering system
Filtering Content
Advanced Behavior
Additional information

6 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)
  )
)
```

6.1 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))
  )
)
```


7 Fine-grained spacing management

The `spacing` dictionary sculpts the rhythm.

7.1 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
)
```


7.2 Horizontal indentation

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

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

Marked Indentation

8 Numbering system

The function relies on Typst’s native engine.

8.1 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

8.2 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

9 Filtering Content

The `filter` parameter allows you to programmatically include or exclude headings from the outline. It expects a callback function `(heading) => boolean`.

The `heading` object passed to the filter contains standard properties (`level`, `body`, `label`, `counter`) as well as context properties: `parent-h1` and `parent-h2`.

9.1 Label-based filtering

In this document, the current section “Filtering Content” has been tagged with the label `<hidden>`.

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

Standard Outline (No Filter)

```
progressive-outline(
  level-2-mode: 'none',
  filter: h => h.label != <hidden>
)
```

Filtered Outline (Label)

9.2 Logic-based filtering

You can also filter based on any heading property. Here, we filter the list to **keep only** the section named “Visibility”.

```
progressive-outline(
  level-2-mode: 'none',
  // Keep only the heading named
'Visibility'
  filter: h => h.body == [Visibility]
)
```

Filtered Outline (Content)

Here, we create a custom rule: show all Level 1 headings, but show Level 2 headings **only** if they belong to the “Visibility” section.

```
progressive-outline(
  level-2-mode: 'all',
  filter: h => h.level == 1 or
    (h.level == 2 and h.parent-h1.body ==
[Visibility])
)
```

Conditional Depth

9.3 Recursive filtering

The filtering logic is recursive: if a parent heading (e.g., a Section) is excluded by the filter, all its children (Subsections and Sub-subsections) are automatically hidden as well, even if they would have passed the filter individually.

```
// Hiding a parent automatically hides its
children
progressive-outline(
  level-2-mode: 'all',
  filter: h => h.label != <hidden>
)
```


10 Advanced Behavior

10.1 Page-based matching

In contexts like sidebars, the outline is rendered in the page margin or background before the slide content. This can cause the active heading detection to fail because the content is technically “after” the sidebar in the document flow.

Setting `match-page-only: true` solves this by considering any heading on the current page as “active”, ignoring precise vertical positioning.

```
match-page-only: true
```

Sidebar Logic

11 Additional information

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