Prism Quarto Extension

Quarto Extension

Mickaël CANOUIL, *Ph.D.*

The `prism` filter lets a single Div, Span, or CodeBlock carry **conditional attributes**: each output format picks its own values from format-prefixed keys, so the same prose or code can be styled differently in HTML, revealjs, and Typst without duplication.

Attributes are keyed `format:name="value"`. At render time, attributes whose `format` matches the active output are re-emitted as `name="value"`; the rest are dropped. Unprefixed attributes pass through unchanged.

How prefixes resolve

Prism matches each prefix against the **target format** of the current render, resolved from `quarto.format.format_identifier()["target-format"]`. Matching is **exact**, so `html:` targets HTML output without affecting revealjs.

This document is rendered to three formats:

Active format	Matching prefix
`html`	`html:`
`revealjs`	`revealjs:`
`typst`	`typst:`

Format-scoped attributes on Divs

Each prefix is independent: `html:style` is applied only under HTML, `revealjs:style` only under revealjs, and `typst:fill` only under Typst. The unprefixed `note` attribute is preserved everywhere.

```
::: {
  note="kept everywhere"
  html:style="color: hotpink;"
  revealjs:style="color: deepskyblue;"
  typst:fill='rgb("#b22222")'
}
This paragraph is pink in HTML, blue in revealjs, and unstyled in Typst.
:::
```

This paragraph is pink in HTML, blue in revealjs, and unstyled in Typst.

Override behaviour

A format-scoped attribute overrides a static attribute of the same name. Under HTML the `html:` value wins; under revealjs the `revealjs:` value wins; under Typst no prefix matches, so the static `gray` is kept.

```
::: {
  style="color: gray;"
  html:style="color: rebeccapurple;"
  revealjs:style="color: forestgreen;"
}
This paragraph reads as purple in HTML, green in revealjs, and gray in Typst.
:::
```

This paragraph reads as purple in HTML, green in revealjs, and gray in Typst.

Format-scoped attributes on Spans

Inline content carries format-scoped attributes the same way.

```
The next sentence styles a
[single    span]{html:style="background:    yellow;    padding:    0    0.2em;"
revealjs:style="background: lightyellow; padding: 0 0.2em;"}
only under the matching format.
```

The next sentence styles a single span only under the matching format.

Format-scoped attributes on CodeBlocks

CodeBlocks accept the same syntax.

````
```{
  .r
  html:style="background: #f6f8fa; padding: 0.5em;"
  revealjs:style="background: #1f2937; color: #e5e7eb; padding: 0.5em;"
}
1 + 1
```
````

```r
1 + 1
```

Unmatched prefixes are dropped

Prefixes that do not resolve to the active target format are removed entirely. The next Div carries only `revealjs:` and `typst:` attributes; under HTML it therefore renders as plain content with no `style` attribute.

```
::: {revealjs:style="color: indigo;" typst:fill='rgb("#0d9488")'}
Plain content in HTML; styled in revealjs and Typst.
:::
```

Plain content in HTML; styled in revealjs and Typst.

Format-group aliases

The `slide` alias matches every HTML slide format: `revealjs`, `slidy`, `s5`, `dzslides`, and `slideous`. A single `slide:` prefix targets the whole group, so the next Div is styled under revealjs without naming each slide format. Under HTML and Typst no slide format is active, so the `slide:` value is dropped.

```
::: {slide:style="color: chocolate;"}
Styled under any HTML slide format; plain in HTML and Typst.
:::
```

Styled under any HTML slide format; plain in HTML and Typst.

An exact format prefix wins over the `slide` alias for the same attribute name. Under revealjs the `revealjs:` value applies; under any other slide format the `slide:` value would apply instead.

```
::: {slide:style="color: teal;" revealjs:style="color: crimson;"}
Crimson under revealjs, teal under the other slide formats.
:::
```

Crimson under revealjs, teal under the other slide formats.

Default fallback

A `default:name` prefix provides a fallback value, applied only when no format-specific variant of the same name matched the active format. Under HTML the `html:` value wins; under revealjs and Typst no specific variant matches, so the `default:` value is used.

```
::: {default:style="color: gray;" html:style="color: rebeccapurple;"}
Purple under HTML, gray under revealjs and Typst.
:::
```

Purple under HTML, gray under revealjs and Typst.