Search (S)

Overview
Tutorial
Reference
LANGUAGE
Syntax
Styling
Scripting
Context
LIBRARY
Foundations
Model
Text
Math
Symbols
Layout
Visualize
Introspection
Data Loading
Guides
Changelog
Roadmap
Community

Reference > Styling

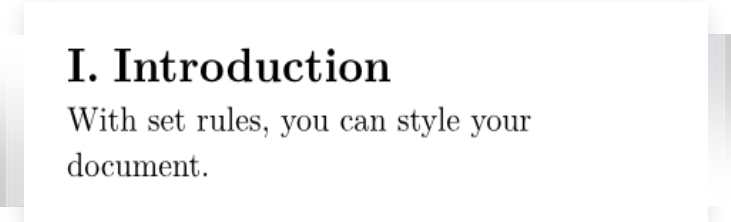
Styling

Typst includes a flexible styling system that automatically applies styling of your choice to your document. With *set rules,* you can configure basic properties of elements. This way, you create most common styles. However, there might not be a built-in property for everything you wish to do. For this reason, Typst further supports *show rules* that can completely redefine the appearance of elements.

Set rules

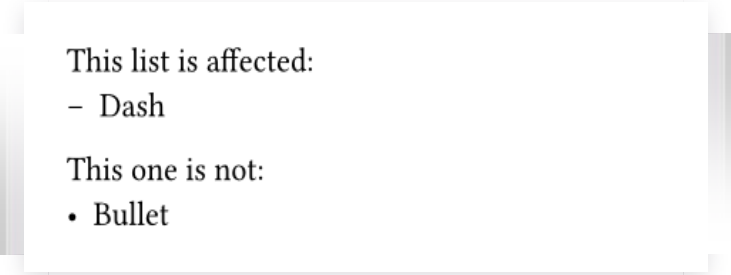
With set rules, you can customize the appearance of elements. They are written as a function call to an element function preceded by the `set` keyword (or `#set` in markup). Only optional parameters of that function can be provided to the set rule. Refer to each function's documentation to see which parameters are optional. In the example below, we use two set rules to change the font family and heading numbering.

```
#set heading(numbering: "I.")
#set text(
  font: "New Computer Modern"
)

= Introduction
With set rules, you can style
your document.
```



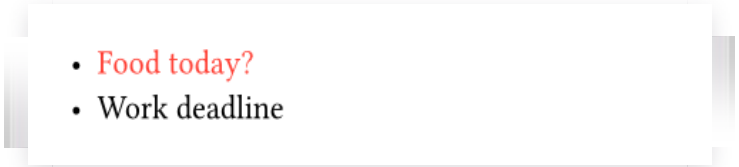
A top level set rule stays in effect until the end of the file. When nested inside of a block, it is only in effect until the end of that block. With a block, you can thus restrict the effect of a rule to a particular segment of your document. Below, we use a content block to scope the list styling to one particular list.

```
This list is affected: #[
  #set list(marker: [--])
  - Dash
]

This one is not:
- Bullet
```



Sometimes, you'll want to apply a set rule conditionally. For this, you can use a *set-if* rule.

```
#let task(body, critical: false) = {
  set text(red) if critical
  [- #body]
}

#task(critical: true)[Food today?]
#task(critical: false)[Work deadline]
```



Show rules

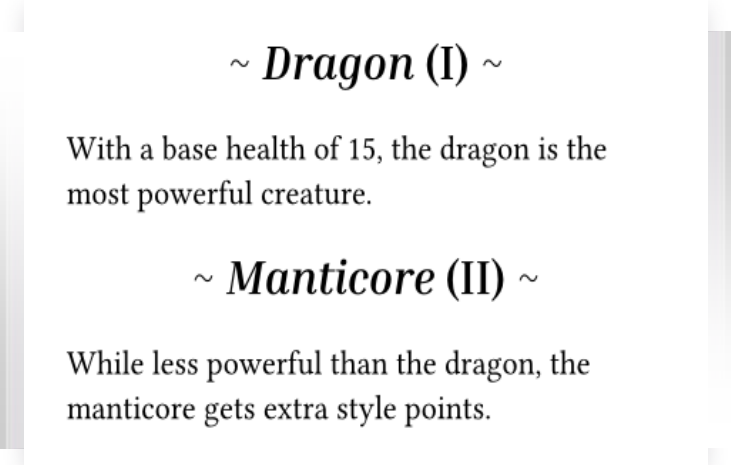
With show rules, you can deeply customize the look of a type of element. The most basic form of show rule is a *show-set rule*. Such a rule is written as the `show` keyword followed by a selector, a colon and then a set rule. The most basic form of selector is an element function. This lets the set rule only apply to the selected element. In the example below, headings become dark blue while all other text stays black.

```
#show heading: set text(navy)

= This is navy-blue
But this stays black.
```

With show-set rules you can mix and match properties from different functions to achieve many different effects. But they still limit you to what is predefined in Typst. For maximum flexibility, you can instead write a show rule that defines how to format an element from scratch. To write such a show rule, replace the set rule after the colon with an arbitrary function. This function receives the element in

question and can return arbitrary content. The available fields on the element passed to the function again match the parameters of the respective element function. Below, we define a show rule that formats headings for a fantasy encyclopedia.

```
#set heading(numbering: "(I)")
#show heading: it => [
  #set align(center)
  #set text(font: "Inria Serif")
  \~ #emph(it.body)
     #counter(heading).display(
       it.numbering
     ) \~
]

= Dragon
With a base health of 15, the
dragon is the most powerful
creature.

= Manticore
While less powerful than the
dragon, the manticore gets
extra style points.
```



Like set rules, show rules are in effect until the end of the current block or file.

Instead of a function, the right-hand side of a show rule can also take a literal string or content block that should be directly substituted for the element. And apart from a function, the left-hand side of a show rule can also take a number of other *selectors*

that define what to apply the transformation to:

- **Everything:** `show: rest => ..`
  Transform everything after the show rule. This is useful to apply a more complex layout to your whole document without wrapping everything in a giant function call.
- **Text:** `show "Text": ..`
  Style, transform or replace text.
- **Regex:** `show regex("\w+"): ..`
  Select and transform text with a regular expression for even more flexibility. See the documentation of the regex type for details.
- **Function with fields:**
  `show heading.where(level: 1): ..`
  Transform only elements that have the specified fields. For example, you might want to only change the style of level-1 headings.
- **Label:** `show <intro>: ..`
  Select and transform elements that have the specified label. See the documentation of the label type for more details.

```
#show "Project": smallcaps
#show "badly": "great"

We started Project in 2019
and are still working on it.
Project is progressing badly.
```

Syntax
Previous page

Scripting
Next page