

Zebraw

Zebraw is a lightweight and fast package for displaying code blocks with line numbers in Typst, supporting code line highlighting. The term **zebraw** is a combination of *zebra* and *raw*, as the highlighted lines display in the code block with a zebra-striped pattern.

Quick Start

Import the zebraw package with `#import '@preview/zebraw:0.5.3';` then add `#show: zebraw` to start using zebraw in the simplest way.

<pre>#import '@preview/zebraw:0.5.3'; #show: zebraw '' typ #grid(columns: (1fr, 1fr), [Hello], [world!], ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

To manually render specific code blocks with zebraw, use the `#zebraw()` function:

<pre>#zebraw('' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
---	---

Features

The zebraw function provides a variety of parameters to customize the appearance and behavior of code blocks. The following sections describe these parameters in detail:

- Core Features**
 - Customizable line numbering and range slicing
 - Line highlighting and explanatory comments for code
 - Headers and footers
 - Language identifier tabs
 - The indentation guide line and hanging indentation (and fast preview mode for better performance)
- Customization Options**
 - Custom colors for background, highlights, and comments
 - Custom fonts for different elements
 - Customizable insets
 - Custom themes
- Export Options**
 - Experimental HTML export

Line Numbering

Line numbers appear on the left side of the code block. Change the starting line number by passing an integer to the numbering-offset parameter. The default value is 0.

<pre>#zebraw(// The first line number will be 2. numbering-offset: 1, '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>2 #grid(3 columns: (1fr, 1fr), 4 [Hello], [world!], 5)</pre>
--	---

To disable line numbering, pass false to the numbering parameter:

<pre>#zebraw(numbering: false, '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>#grid(columns: (1fr, 1fr), [Hello], [world!],)</pre>
---	---

For more advanced numbering control, pass an array of arrays to the numbering parameter. Each inner array represents a column of markers that will be displayed instead of standard line numbers. This allows displaying multiple line numbers, markers or custom identifiers for each line.

<pre>#zebraw(numbering: ((1..), (1..), (1..), (1..),), '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>+ #grid(+ columns: (1fr, 1fr), + # [Hello], [world!], +)</pre>
---	---

Numbering Separator

You can add a separator line between line numbers and code content by setting the numbering-separator parameter to true:

<pre>#zebraw(numbering-separator: true, '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

Line Slicing

Slice code blocks by passing the line-range parameter to the zebraw function. The line-range parameter can be either:

- An array of 2 integers representing the range $[a, b)$ (b can be none as this feature is based on Typst array slicing)
- A dictionary with range and keep-offset keys

When keep-offset is set to true, line numbers maintain their original values. Otherwise, they reset to start from 1. By default, keep-offset is set to true.

<pre>#let code = '' typ #grid(columns: (1fr, 1fr), [Hello], [world!], ...) #zebraw(code) #zebraw(line-range: (2, 4), code) #zebraw(line-range: (range: (2, 4), keep-offset: false), code) #zebraw(numbering-offset: 30, line-range: (range: (2, 4), keep-offset: false), code) #zebraw(numbering-offset: 30, line-range: (range: (2, 4), keep-offset: true), code)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], 4 [world!], 5) 2 columns: (1fr, 1fr), 3 [Hello], 31 columns: (1fr, 1fr), 32 [Hello], 32 columns: (1fr, 1fr), 33 [Hello],</pre>
---	--

Line Highlighting

Highlight specific lines in the code block by passing the highlight-lines parameter to the zebraw function. The highlight-lines parameter accepts either a single line number or an array of line numbers.

<pre>#zebraw(// Single line number: highlight-lines: 2, '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...) #zebraw(// Array of line numbers: highlight-lines: (6, 7) + range(9, 15), '' typ = Fibonacci sequence The Fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$. It can also be expressed in <i>closed form</i>_: $SF_n = round(1 / \sqrt{5}) \phi^{n+1} n$, quad $\phi^{n+1} = (1 + \sqrt{5}) / 2$ #let count = 8 #let nums = range(1, count + 1) #let fib(n) = (if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) }) The first #count numbers of the sequence are: #align(center, table(columns: count, ..nums.map(n => SF_#n\$), ..nums.map(n => str(fib(n))),)) ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4) 1 = Fibonacci sequence 2 The Fibonacci sequence is defined through the 3 recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$. 4 It can also be expressed in <i>closed form</i>_: 5 $SF_n = round(1 / \sqrt{5}) \phi^{n+1} n$, quad 6 $\phi^{n+1} = (1 + \sqrt{5}) / 2$ 7 8 9 #let count = 8 10 #let nums = range(1, count + 1) 11 #let fib(n) = (12 if n <= 2 { 1 } 13 else { fib(n - 1) + fib(n - 2) } 14) 15 16 The first #count numbers of the sequence are: 17 18 #align(center, table(19 columns: count, 20 ..nums.map(n => SF_#n\$), 21 ..nums.map(n => str(fib(n))), 22))</pre>
--	---

Comments

Add explanatory comments to highlighted lines by passing an array of line numbers and comments to the highlight-lines parameter.

<pre>#zebraw(highlight-lines: ((1, [The Fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$]),), '' typ = Fibonacci sequence The Fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$. It can also be expressed in <i>closed form</i>_: $SF_n = round(1 / \sqrt{5}) \phi^{n+1} n$, quad $\phi^{n+1} = (1 + \sqrt{5}) / 2$ // Passing a range of line numbers in the array should begin with "...". ..range(9, 14), (13, [The first #count numbers of the sequence.]),), '' typ = Fibonacci sequence #let count = 8 #let nums = range(1, count + 1) #let fib(n) = (if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) }) #align(center, table(columns: count, ..nums.map(n => SF_#n\$), ..nums.map(n => str(fib(n))),)) ...)</pre>	<pre>1 = Fibonacci sequence 2 > The Fibonacci sequence is defined through the 3 recurrence relation $F_n = F_{n-1} + F_{n-2}$ 4 It can also be expressed in <i>closed form</i>_: 5 $F_n = \left\lfloor \frac{1}{\sqrt{5}} \phi^n \right\rfloor$, $\phi = \frac{1 + \sqrt{5}}{2}$ 6 7 8 #let count = 8 9 #let nums = range(1, count + 1) 10 #let fib(n) = (11 if n <= 2 { 1 } 12 else { fib(n - 1) + fib(n - 2) } 13) 14 15 16 The first #count numbers of the sequence. 17 18 #align(center, table(19 columns: count, 20 ..nums.map(n => SF_#n\$), 21 ..nums.map(n => str(fib(n))), 22)) 23 24 > The first #count numbers of the sequence.</pre>
---	---

Comments begin with a flag character, which is "-" by default. Change this flag by setting the comment-flag parameter:

<pre>#zebraw(highlight-lines: (// Comments can only be passed when highlight-lines is an array, so a comma is needed at the end of a single- element array (6, [The Fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$]),), comment-flag: "-->", '' typ = Fibonacci sequence #let count = 8 #let nums = range(1, count + 1) #let fib(n) = (if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) }) #align(center, table(columns: count, ..nums.map(n => SF_#n\$), ..nums.map(n => str(fib(n))),)) ...)</pre>	<pre>1 = Fibonacci sequence 2 #let count = 8 3 #let nums = range(1, count + 1) 4 #let fib(n) = (5 if n <= 2 { 1 } 6 else { fib(n - 1) + fib(n - 2) } 7) 8 9 --> The Fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$ 10 11 #align(center, table(12 columns: count, 13 ..nums.map(n => SF_#n\$), 14 ..nums.map(n => str(fib(n))), 15)) 16 17 > The first #count numbers of the sequence.</pre>
--	---

To disable the flag feature entirely, pass an empty string "" to the comment-flag parameter (this also disables comment indentation):

<pre>#zebraw(highlight-lines: ((6, [The Fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$]),), comment-flag: "", '' typ = Fibonacci sequence #let count = 8 #let nums = range(1, count + 1) #let fib(n) = (if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) }) #align(center, table(columns: count, ..nums.map(n => SF_#n\$), ..nums.map(n => str(fib(n))),)) ...)</pre>	<pre>1 = Fibonacci sequence 2 #let count = 8 3 #let nums = range(1, count + 1) 4 #let fib(n) = (5 if n <= 2 { 1 } 6 else { fib(n - 1) + fib(n - 2) } 7) 8 9 The Fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$ 10 11 #align(center, table(12 columns: count, 13 ..nums.map(n => SF_#n\$), 14 ..nums.map(n => str(fib(n))), 15)) 16 17 The fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$</pre>
--	--

Headers and Footers

You can add headers and footers to code blocks. One approach is to use special keys in the highlight-lines parameter:

<pre>#zebraw(highlight-lines: ((header: ["Fibonacci sequence"]), ..range(8, 13),), // Numbers can be passed as strings in the dictionary, // though this approach is less elegant ("12", [The first #count numbers of the sequence.]), (footer: [The fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$]),), '' typ #let count = 8 #let nums = range(1, count + 1) #let fib(n) = (if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) }) #align(center, table(columns: count, ..nums.map(n => SF_#n\$), ..nums.map(n => str(fib(n))),)) ...)</pre>	<pre>Fibonacci sequence 1 #let count = 8 2 #let nums = range(1, count + 1) 3 #let fib(n) = (4 if n <= 2 { 1 } 5 else { fib(n - 1) + fib(n - 2) } 6) 7 8 #align(center, table(9 columns: count, 10 ..nums.map(n => SF_#n\$), 11 ..nums.map(n => str(fib(n))), 12)) 13 14 > The first #count numbers of the sequence. The fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$</pre>
---	---

Alternatively, use the dedicated header and footer parameters for cleaner code:

<pre>#zebraw(highlight-lines: ((12, [The first #count numbers of the sequence.]),), header: ["Fibonacci sequence"], '' typ #let count = 8 #let nums = range(1, count + 1) #let fib(n) = (if n <= 2 { 1 } else { fib(n - 1) + fib(n - 2) }) #align(center, table(columns: count, ..nums.map(n => SF_#n\$), ..nums.map(n => str(fib(n))),)) ... footer: [The fibonacci sequence is defined through the recurrence relation $SF_n = F_{(n-1)} + F_{(n-2)}$],)</pre>	<pre>Fibonacci sequence 1 #let count = 8 2 #let nums = range(1, count + 1) 3 #let fib(n) = (4 if n <= 2 { 1 } 5 else { fib(n - 1) + fib(n - 2) } 6) 7 8 #align(center, table(9 columns: count, 10 ..nums.map(n => SF_#n\$), 11 ..nums.map(n => str(fib(n))), 12)) 13 14 > The first #count numbers of the sequence. The fibonacci sequence is defined through the recurrence relation $F_n = F_{n-1} + F_{n-2}$</pre>
---	---

Language Tab

Display a floating language identifier tab in the top-right corner of the code block by setting lang to true:

<pre>#zebraw(lang: true, '' typst #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>typst 1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
---	---

Customize the language display by passing a string or content to the lang parameter:

<pre>#zebraw(lang: strong(Typst), '' typst #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>Typst 1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

Indentation Lines, Hanging Indentation and Fast Preview

Display indentation guides by passing a positive integer to the indention parameter, representing the number of spaces per indention level:

<pre>#zebraw(indention: 2, '' typ #let forecast(day) = block{ #box(square(width: 2cm, inset: 8pt, fill: if day.weather == "sunny" { yellow } else { aqua }, align(bottom + right, strong(day.weather),),)) #h(6pt) #set text(22pt, baseline: -8pt) #day.temperature *day.unit } ...)</pre>	<pre>1 #let forecast(day) = block{ 2 #box(square(3 width: 2cm, 4 inset: 8pt, 5 fill: if day.weather == "sunny" { 6 yellow 7 } else { 8 aqua 9 }, 10 align(11 bottom + right, 12 strong(day.weather), 13), 14)) 15 #h(6pt) 16 #set text(22pt, baseline: -8pt) 17 #day.temperature *day.unit 18 }</pre>
--	---

Enable hanging indentation by setting hanging-indent to true:

<pre>#zebraw(hanging-indent: true, '' typ #let forecast(day) = block{ #box(square(width: 2cm, inset: 8pt, fill: if day.weather == "sunny" { yellow } else { aqua }, align(bottom + right, strong(day.weather),),)) #h(6pt) #set text(22pt, baseline: -8pt) #day.temperature *day.unit } ...)</pre>	<pre>1 #let forecast(day) = block{ 2 #box(square(3 width: 2cm, 4 inset: 8pt, 5 fill: if day.weather == "sunny" { 6 yellow 7 } else { 8 aqua 9 }, 10 align(11 bottom + right, 12 strong(day.weather), 13), 14)) 15 #h(6pt) 16 #set text(22pt, baseline: -8pt) 17 #day.temperature *day.unit 18 }</pre>
--	---

Indentation lines can slow down preview performance. For faster previews, enable fast preview mode by passing true to the fast-preview parameter in zebraw-init or by using zebraw-fast-preview in the CLI. This renders indentation lines as simple | characters:

<pre>#zebraw(hanging-indent: true, '' typ #let forecast(day) = block{ #box(square(width: 2cm, inset: 8pt, fill: if day.weather == "sunny" { yellow } else { aqua }, align(bottom + right, strong(day.weather),),)) #h(6pt) #set text(22pt, baseline: -8pt) #day.temperature *day.unit } ...)</pre>	<pre>1 #let forecast(day) = block{ 2 #box(square(3 width: 2cm, 4 inset: 8pt, 5 fill: if day.weather == "sunny" { 6 yellow 7 } else { 8 aqua 9 }, 10 align(11 bottom + right, 12 strong(day.weather), 13), 14)) 15 #h(6pt) 16 #set text(22pt, baseline: -8pt) 17 #day.temperature *day.unit 18 }</pre>
--	---

Themes

Zebraw includes built-in themes. PRs for additional themes are welcome!

<pre>#show: zebraw.with(.zebraw-themes.zebra) '' rust pub fn fibonacci_recursive(n: i32) -> u64 { if n < 0 { panic!("{}", "is negative!", n); } match n { 0 => panic!("zero is not a right argument to fibonacci_recursive()"), 1 2 => 1, 3 => 2, _ => fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2), } ... }</pre>	<pre>1 pub fn fibonacci_recursive(n: i32) -> u64 { 2 if n < 0 { 3 panic!("{}", "is negative!", n); 4 } 5 match n { 6 0 => panic!("zero is not a right argument to fibonacci_recursive()"), 7 1 2 => 1, 8 3 => 2, 9 _ => fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2), 10 } 11 }</pre>
--	---

<pre>#show: zebraw.with(.zebraw-themes.zebra-reverse) '' rust pub fn fibonacci_recursive(n: i32) -> u64 { if n < 0 { panic!("{}", "is negative!", n); } match n { 0 => panic!("zero is not a right argument to fibonacci_recursive()"), 1 2 => 1, 3 => 2, _ => fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2), } ... }</pre>	<pre>1 pub fn fibonacci_recursive(n: i32) -> u64 { 2 if n < 0 { 3 panic!("{}", "is negative!", n); 4 } 5 match n { 6 0 => panic!("zero is not a right argument to fibonacci_recursive()"), 7 1 2 => 1, 8 3 => 2, 9 _ => fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2), 10 } 11 }</pre>
--	---

(Experimental) HTML Export

See example-htmltypst or GitHub Pages for more information.

Customization

There are three ways to customize code blocks using the zebraw() function with:

- Per-block customization:** Manually style specific blocks using the `#zebraw()` function with parameters.
- Local customization:** Apply styling to all subsequent raw blocks with `#show: zebraw.with(0)`. This affects all raw blocks after the `#show` rule, **except** those created manually with `#zebraw()`.
- Global customization:** Use `#show: zebraw-init.with(0)` to affect all raw blocks after the rule, **including** those created manually with `#zebraw()`. Reset to defaults by using zebraw-init without parameters.

Inset

Customize the padding around each code line(numberings are not affected) by passing a dictionary to the inset parameter:

<pre>#zebraw(inset: (top: 6pt, bottom: 6pt), '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
---	---

Colors

Customize the background color with a single color or an array of alternating colors:

<pre>#zebraw(background-color: luma(250), '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...) #zebraw(background-color: (luma(235), luma(245), luma(255), luma(245)), '' typ #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4) 1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

Set the highlight color for marked lines with the highlight-color parameter:

<pre>#zebraw(highlight-lines: 1, highlight-color: blue.lighten(90%), '' text I'm so blue! ... -- George III)</pre>	<pre>1 I'm so blue! 2 -- George III</pre>
--	---

Change the comment background color with the comment-color parameter:

<pre>#zebraw(highlight-lines: ((2, "auto indent!"),), comment-color: yellow.lighten(90%), '' text I'm so blue! ... -- George III I'm not. -- Hamilton)</pre>	<pre>1 I'm so blue! 2 -- George III 3 -- auto indent! 3 I'm not. 4 -- Hamilton</pre>
--	--

Set the language tab background color with the lang-color parameter:

<pre>#zebraw(lang: true, lang-color: teal, '' typst #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>typst 1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
---	---

Font

Customize font properties for comments, language tabs, and line numbers by passing a dictionary to the comment-font-args, lang-font-args, or numbering-font-args parameters respectively.

If no custom lang-font-args are provided, language tabs inherit the comment font styling:

<pre>#zebraw(highlight-lines: ((2, "columns..."),), lang: true, lang-color: eastern, lang-font-args: (font: "Buenard", weight: "bold", fill: white,), '' typst #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>typst 1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

Example with custom language tab styling:

<pre>#zebraw(highlight-lines: ((2, "columns..."),), lang: true, lang-color: eastern, lang-font-args: (font: "Buenard", weight: "bold", fill: white,), '' typst #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>typst 1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

Extend

Extend at vertical is enabled at default. When there's header or footer it will be automatically disabled.

<pre>#zebraw(extend: false, '' typst #grid(columns: (1fr, 1fr), [Hello], [world!],) ...)</pre>	<pre>1 #grid(2 columns: (1fr, 1fr), 3 [Hello], [world!], 4)</pre>
--	---

Example

Calculate Fibonacci number using recursive function

```
1 pub fn fibonacci_reccursive(n: i32) -> u64 {  
2     if n < 0 {  
3         panic!("{}", n);  
        > to avoid negative numbers  
4     }  
5     match n {  
6         0 => panic!("zero is not a right argument to fibonacci_reccursive()!"),  
7         1 | 2 => 1,  
8         3 => 2,  
9         _ => fibonacci_reccursive(n - 1) + fibonacci_reccursive(n - 2),  
        > 50 => 12586269025  
10    }  
11 }
```