Documentation

The default value of most parameters are `none` for it will use the default value in `zebraw-init`.

- zebraw()

zebraw

Block of code with highlighted lines and comments.

Parameters

```
zebraw(
  numbering: boolean,
  highlight-lines: array int,
  numbering-offset: int,
  header: string content,
  footer: string content,
  inset: dictionary,
  background-color: color array,
  highlight-color: color,
  comment-color: color,
  lang-color: color,
  comment-flag: string content,
  lang: boolean string content,
  comment-font-args: dictionary,
  lang-font-args: dictionary,
  numbering-font-args: dictionary,
  extend: boolean,
  numbering-separator: boolean,
  hanging-indent: boolean,
  indentation: int,
  line-range: array dictionary,
  block-width: length relative,
  wrap: boolean,
  body: content
) -> content
```

numbering `boolean`

Whether to show the line numbers.

~~~
#zebraw(
  numbering: false,
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
   )
  ~~~
)
```

typ

```
#grid(
  columns: (1fr, 1fr),
  [Hello,], [world!],
)
```

Default: `none`

highlight-lines `array` or `int`

Lines to highlight or comments to show.

```
#zebraw(
  highlight-lines: range(3, 7),
  header: [*Fibonacci sequence*],
  ~~~typst
  #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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ```,
  footer: [The fibonacci sequence is
defined through the recurrence relation
$F_n = F_(n-1) + F_(n-2)$],
)
~~~

typst

```
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 => $F_#n$),
11   ..nums.map(n => str(fib(n))),
12 ))
The fibonacci sequence is defined
through the recurrence relation
$F_n = F_{n-1} + F_{n-2}$
```

Default: `()`

numbering-offset `int`

The offset of line numbers. The first line number will be `numbering-offset + 1`. Defaults to `0`.

Default: `0`

header `string` or `content`

The header of the code block.

Default: `none`

footer `string` or `content`

The footer of the code block.

Default: `none`

inset `dictionary`

The inset of each line.

~~~
#zebraw(
  inset: (top: 6pt, bottom: 6pt),
  ```typst
  #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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ```,
)
~~~

typst

```
 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 => $F_#n$),

11   ..nums.map(n => str(fib(n))),

12 ))
```

Default: `none`

background-color `color` or `array`

The background color of the block and normal lines.

~~~
#zebraw(
  background-color: (luma(240), luma(245),
luma(250), luma(245)),
  ```typst
  #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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ```,
)
~~~

typst

```
 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 => $F_#n$),
11   ..nums.map(n => str(fib(n))),
12 ))
```

Default: `none`

highlight-color `color`

The background color of the highlighted lines.

Default: `none`

comment-color `color`

The background color of the comments. The color is set to `none` at default and it will be rendered in a lightened `highlight-color`.

~~~
#zebraw(
  highlight-color: yellow.lighten(80%),
  comment-color: yellow.lighten(90%),
  highlight-lines: (
    (1, [The Fibonacci sequence is defined
through the recurrence relation $F_n =
F_(n-1) + F_(n-2)$]),
    ..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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ~~~
)
```

typ

```
 1 = Fibonacci sequence
   > The Fibonacci sequence is
   defined through the recurrence
   relation $F_n = F_{n-1} + F_{n-2}$
 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 #align(center, table(
10   columns: count,
11   ..nums.map(n => $F_#n$),
12   ..nums.map(n => str(fib(n))),
13 ))
   > The first #count numbers of
   the sequence.
```

Default: `none`

lang-color `color`

The background color of the language tab. The color is set to `none` at default and it will be rendered in comments' color.

```
#zebraw(
  lang: true,
  lang-color: eastern,
  lang-font-args: (
    font: "libertinus serif",
    weight: "bold",
    fill: white
  ),
  ```typst
  #grid(
    columns: (1fr, 1fr),
    [Hello], [world!],
  )
  ```
)
```

typst

```
1 #grid(
2   columns: (1fr, 1fr),
3   [Hello], [world!],
4 )
```

Default: `none`

comment-flag `string` or `content`

The flag at the beginning of comments. The indentation of codes will be rendered before the flag. When the flag is set to `""`, the indentation before the flag will be disabled as well.

```
#zebraw(
  comment-flag: "",
  highlight-lines: (
    (1, [The Fibonacci sequence is defined
through the recurrence relation $F_n =
F_(n-1) + F_(n-2)$]),
    ..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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ```
)
```

typ

```
 1 = Fibonacci sequence
   The Fibonacci sequence is
   defined through the recurrence
   relation $F_n = F_{n-1} + F_{n-2}$
 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 #align(center, table(
10   columns: count,
11   ..nums.map(n => $F_#n$),
12   ..nums.map(n => str(fib(n))),
13 ))
   The first #count numbers of the
   sequence.
```

Default: `none`

lang `boolean` or `string` or `content`

Whether to show the language tab, or a string or content of custom language name to display.

```
#zebraw(
  lang: true,
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
  )
  ```
)
```

typ

```
1 #grid(
2   columns: (1fr, 1fr),
3   [Hello,], [world!],
4 )
```

```
#zebraw(
  lang: strong[Typst],
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
  )
  ```
)
```

Typst

```
1 #grid(
2   columns: (1fr, 1fr),
3   [Hello,], [world!],
4 )
```

Default: `none`

comment-font-args `dictionary`

The arguments passed to comments' font.

Default: `none`

lang-font-args `dictionary`

The arguments passed to the language tab's font.

```
#zebraw(
  lang: true,
  comment-font-args: (font: "IBM Plex
Serif", style: "italic"),
  lang-font-args: (font: "IBM Plex Sans",
weight: "bold"),
  highlight-lines: (
    (1, [The Fibonacci sequence is defined
through the recurrence relation $F_n =
F_(n-1) + F_(n-2)$]),
    ..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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ```
)
```

typ

```
 1 = Fibonacci sequence
   > The Fibonacci sequence is defined through
   the recurrence relation $F_n = F_{n-1} + F_{n-2}$
 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 #align(center, table(
10   columns: count,
11   ..nums.map(n => $F_#n$),
12   ..nums.map(n => str(fib(n))),
13 ))
   > The first #count numbers of the sequence.
```

Default: `none`

numbering-font-args `dictionary`

The arguments passed to the line numbers' font.

```
#zebraw(
  numbering-font-args: (fill: blue),
  ```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 => $F_#n$),
    ..nums.map(n => str(fib(n))),
  ))
  ```
)
```

typ

```
 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 #align(center, table(
10   columns: count,
11   ..nums.map(n => $F_#n$),
12   ..nums.map(n => str(fib(n))),
13 ))
```

Default: `none`

extend `boolean`

Whether to extend the vertical spacing.

```
#zebraw(
  extend: false,
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
  )
  ```
)
```

typ

```
1 #grid(
2   columns: (1fr, 1fr),
3   [Hello,], [world!],
4 )
```

Default: `none`

numbering-separator `boolean`

Whether to show the numbering separator line.

```
#zebraw(
  numbering-separator: true,
  ```typ
  #grid(
     columns: (1fr, 1fr),
     [Hello,], [world!],
  )
  ```
)
```

typ

```
1 | #grid(
2 |    columns: (1fr, 1fr),
3 |    [Hello,], [world!],
4 | )
```

Default: `none`

hanging-indent `boolean`

Whether to show the hanging indent.

```
#zebraw(
  hanging-indent: true,
  ```typ
  This is a short line.
    Do a deer, a female deer. Ray, a drop
of golden sun. Me, a name I call myself.
Far, a long, long way to run. Sew, a
needle pulling thread. La, a note to
follow sew. Tea, a drink with jam and
bread. That will bring us back to do, oh,
oh, oh.
  ```
)
```

typ

```
1 This is a short line.
2   Do a deer, a female deer. Ray,
    a drop of golden sun. Me, a
    name I call myself. Far, a
    long, long way to run. Sew, a
    needle pulling thread. La, a
    note to follow sew. Tea, a
    drink with jam and bread. That
    will bring us back to do, oh,
    oh, oh.
```

Default: `none`

indentation `int`

The amount of indentation, used to draw indentation lines.

```
#zebraw(
  indentation: 2,
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
  )
  ```
)
```

typ

```
1 #grid(
2 | columns: (1fr, 1fr),
3 | [Hello,], [world!],
4 )
```

Default: `none`

line-range `array` or `dictionary`

Line range to show. Accepts an array of 2 integers [a, b) or a dictionary with keys named `range` and `keep-offset`. Defaults to [1, none). (none means the last line). Noticed that the line numbers are 1-based.

```
#zebraw(
  line-range: (2, 4),
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
  )
  ```
)
```

typ

```
2   columns: (1fr, 1fr),
3   [Hello,], [world!],
```

```
#zebraw(
  line-range: (range: (2, 4), keep-offset:
false),
  ```typ
  #grid(
    columns: (1fr, 1fr),
    [Hello,], [world!],
  )
  ```
)
```

typ

```
1   columns: (1fr, 1fr),
2   [Hello,], [world!],
```

Default: `(1, none)`

block-width `length` or `relative`

(Only for HTML) The width of the code block.

Default: `42em`

wrap `boolean`

(Only for HTML) Whether to wrap the code lines.

Default: `true`

body `content`

The body.
```