blockst Version 0.3.0

Render Scratch-style blocks for educational documents.

blockst renders Scratch-style programming blocks directly in Typst documents — for worksheets, tutorials and teaching material. Scratch code is written as plain text and rendered by a bundled WASM plugin, in 26 languages including right-to-left scripts, with a turtle-graphics execution engine and helpers for importing real `.sb3` project files.

AUTOREN Loewe1000

VERWEISE GitHub

Inhalt

1 Introduction

1.1 About blockst

blockst renders Scratch-style programming blocks directly in Typst documents. It is designed for worksheets, tutorials, teaching material, and visual programming explanations — anything where Scratch-like block syntax needs to appear in print or online documentation.

The current renderer uses a text-to-WASM pipeline: Typst passes Scratch text to a bundled WASM plugin, the plugin parses and renders SVG, and Typst embeds the SVG output in the document.

CORE DESIGN

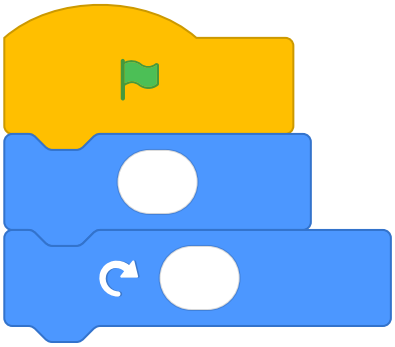
- Fully text-based: write Scratch blocks as plain text, get rendered blocks.
- *26 languages* supported via built-in WASM locale data, including right-to-left scripts.
- *Localized* block rendering follows official Scratch translations.
- *Turtle graphics* execution engine for demonstrating program flow visually.
- *SB3 import* helpers for reading real Scratch project files.

BREAKING CHANGE SINCE 0.2.0

The old pre-0.2.0 native-Typst renderer syntax is **removed**. From 0.2.0 onward, blockst uses only the text-to-WASM pipeline. Documents that still rely on the previous syntax must be migrated.

1.2 Quick start

```
#import "@preview/blockst:0.3.0": blockst, scratch, raw-scratch, sb3

#scratch("
when green flag clicked
move (10) steps
turn cw (15) degrees
")
```



1.3 Package information

- **Version:** 0.3.0
- **License:** MIT
- **Repository:** github.com/Loewe1000/blockst
- **Compiler requirement:** Typst 0.15.0+

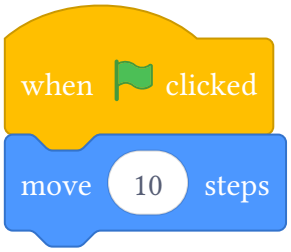
- **Font requirement:** Designed for Helvetica Neue (Scratch look). On Linux/Windows install a compatible font (e.g. Nimbus Sans) or override via set-blockst.

2 Core Rendering API

2.1 scratch() — Render Scratch Blocks

The primary function for rendering Scratch blocks. It parses Scratch-like text and produces visual blocks.

```
#scratch("when green flag clicked\nmove (10) steps")
```



2.1.1 Signature

```
#scratch(
  text,
  language: "en",
  theme: auto,
  scale: auto,
  line-numbers: auto,
  line-number-start: auto,
  line-number-gutter: auto,
  inset-scale: auto,
)
```

2.1.2 Parameters

Name	Default	Description
`text`	– (required)	Scratch block text. Supports blocks, reporters, booleans, inputs, dropdowns, and nested control structures.
`language`	`"en"`	Locale for block text. See Languages for available options.
`theme`	`auto`	Visual theme: `"normal"`, `"high-contrast"`, `"print"`, `"grayscale"`, or `auto` (global default).
`scale`	`auto`	Overall size multiplier (e.g. `80%`, `0.8`, `50%`).
`font`	`auto`	Font for block labels. Set it here rather than with `#set text(font: ...)`: the renderer measures each label and writes the font into the SVG, so a document-level `set text` changes what is drawn but not what was measured, and the labels overlap.
`line-numbers`	`auto`	Show line numbers (`true` or `false`).
`line-number-start`	`auto`	Starting line number (integer, default `1`).
`line-number-gutter`	`auto`	Width of line number gutter (pt, default `24`).
`inset-scale`	`auto`	Proportional block geometry scaling. `100%` = default, `60%` = thin blocks, `125%` = thick blocks. Does not affect text size.

2.1.3 Supported block syntax

The text parser supports the full Scratch 3 block vocabulary in all 26 locales. Key syntax patterns:

Pattern	Example	Description
Stack block	`move (10) steps`	Standard command block
Hat block	`when green flag clicked`	Event trigger
Reporter	`(x position)`	Value reporter (round ends)
Boolean	`<mouse down?>`	Predicate (pointy ends)
Input: number	`(10)`	Numeric input field
Input: string	`[hello]`	String input field
Input: dropdown	`[random position v]`	Dropdown selector
C-block	`repeat (4)\n...\nend`	Block with body
C-block + else	`if <...> then\n...\nelse\n...\nend`	Conditional with else
Line labels	`move (10) steps #step`	Named label (see Labels)
Category prefix	`@motion free text`	Force category color (see category)
Category prefix (render)	`#text("@category")` + text	Matches real block when text fits

2.2 set-blockst() — Global Defaults

Sets global rendering options for all subsequent `scratch()` and SB3 render calls.

```
#set-blockst(
  theme: none,
  scale: none,
  stroke-width: none,
  font: none,
  line-numbers: none,
  line-number-start: none,
  line-number-gutter: none,
  inset-scale: none,
)
```

All parameters are optional. Only provided values override the current defaults.

2.2.1 Themes

Theme	Description
`"normal"`	Default — full color Scratch style.
`"high-contrast"`	Lighter, high-contrast variant for accessibility.
`"print"`	Every block white with a black outline. The lightest on ink, but all categories look alike.
`"grayscale"`	One distinct grey per category, so a control block still reads differently from an operator on a monochrome page.

```
#set-blockst(theme: "print", scale: 70%)
#scratch("when green flag clicked\nmove (10) steps")
```

2.2.2 Fonts

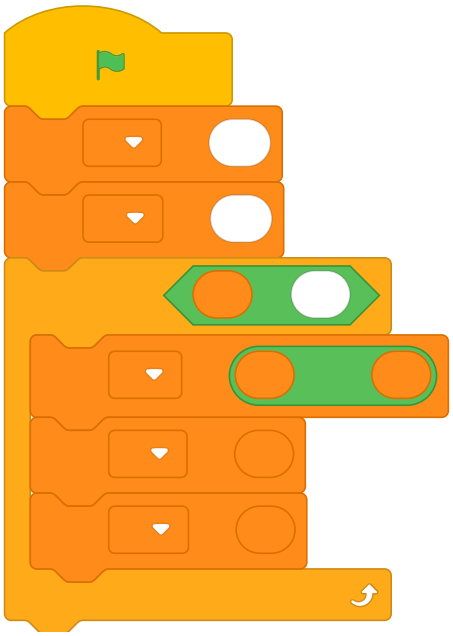
```
#set-blockst(font: "Nimbus Sans")
```

The default font is Helvetica Neue. On systems without it, set an alternative.

2.2.3 Line numbers

```
#set-blockst(line-numbers: true, line-number-start: 1, line-number-gutter: 24)
```

Euclidean Algorithm (gcd label walkthrough)



Explanation of labeled lines

- **Line 4**: Loop condition `b != 0` controls termination.
- **Line 5**: Core rule: `r = a mod b`.
- **Line 7**: State update that advances to the next pair.

2.2.4 Inset scale

Controls the visual thickness/slimness of blocks without changing text size.

```
#set-blockst(inset-scale: 60%)   // compact blocks
#set-blockst(inset-scale: 125%)  // generous blocks
```

2.3 blockst() — Group Override Container

Optional wrapper for grouped blocks that differ from global settings.

```
#blockst(
  theme: auto,
  scale: auto,
  line-numbers: auto,
  line-number-start: auto,
  line-number-gutter: auto,
  inset-scale: auto,
  spacing: 1.5em,
  body,
)
```

All parameters match `scratch()` but apply only within the body. Use when a specific group of blocks needs different scaling or theme.

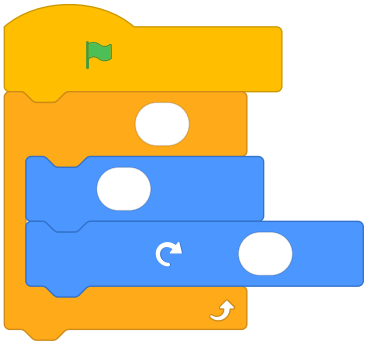
```
#blockst(theme: "high-contrast")[
  #scratch("when green flag clicked\nmove (10) steps")
]
```

3 Languages

blockst supports **26 languages** via built-in WASM locale data. Set via `scratch(..., language: "de")` or `set-blockst(...)`.

Code	Language	Code	Language
`"en"`	English	`"de"`	German
`"fr"`	French	`"es"`	Spanish
`"it"`	Italian	`"nl"`	Dutch
`"pt"`	Portuguese	`"pl"`	Polish
`"ru"`	Russian	`"ja"`	Japanese
`"ca"`	Catalan	`"cs"`	Czech
`"cy"`	Welsh	`"el"`	Greek
`"fa"`	Persian	`"gd"`	Scottish Gaelic
`"he"`	Hebrew	`"hi"`	Hindi
`"hr"`	Croatian	`"hu"`	Hungarian
`"id"`	Indonesian	`"nb"`	Norwegian (Bokmål)
`"ro"`	Romanian	`"sl"`	Slovenian
`"tr"`	Turkish	`"ar"`	Arabic

```
#set-blockst(scale: 67.5%)
#scratch("
Wenn die grüne Flagge angeklickt
wiederhole (4) mal
  gehe (30) er Schritt
  drehe dich nach rechts um (90) Grad
end
", language: "de")
```



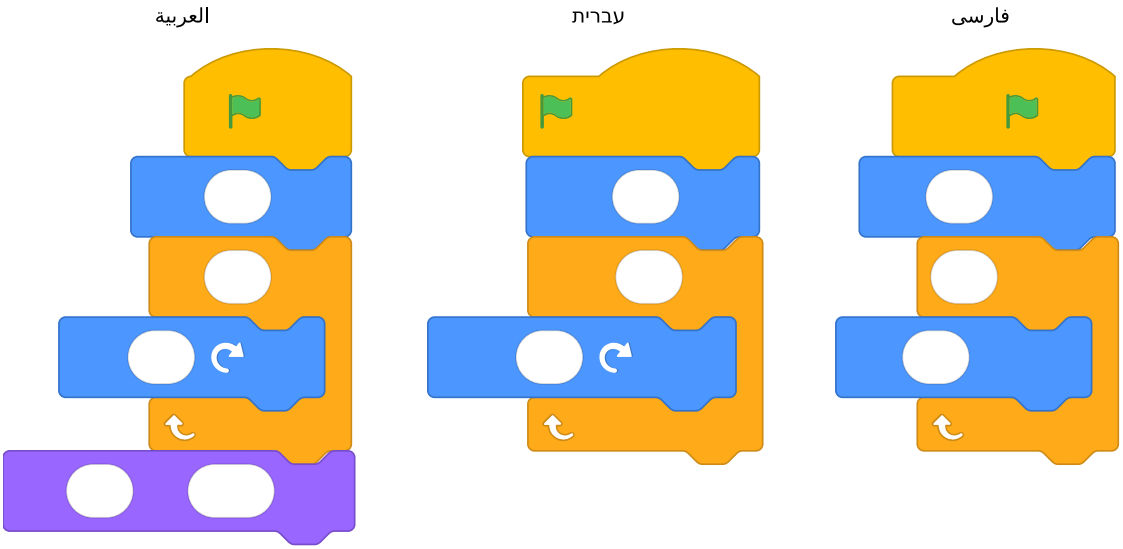
Block text must match the chosen locale's vocabulary — the parser matches against official Scratch translations.

3.1 Right-to-left languages

Arabic (`"ar"`), Hebrew (`"he"`) and Persian (`"fa"`) render right-to-left. Nothing has to be switched on: each locale declares its own direction, and the renderer mirrors the layout — the notch, the hat dome, the

C-block mouth, the loop arrow, the pen badge, the `define` hat and the order of the labels all move to the reading edge.

```
#scratch("
عند نقر @greenFlag
تحرك (10) خطوة
كرِّر (4) مرة
استدر @turnRight (90) درجة
نهاية
", language: "ar")
```



> **WHAT IS MIRRORED, AND WHAT IS NOT**
> Layout is mirrored; meaning is not. The loop arrow points back to the top of the loop, which is a claim about the drawing, so it flips with the drawing. `@turnRight` and `@turnLeft` are never mirrored — their direction is what the sprite is being told to do, and a mirrored `@turnRight` would tell the reader to turn the other way.

Arabic short vowels are optional and their order is not canonical, so blocks match whether or not you type the harakat: `كرِّر` and `كرر` both find the repeat block.

4 Labels and Line Numbers

blockst provides a label system for creating line-aware worksheets. Lines ending with `#label-name` are tagged and can be referenced later.

4.1 scratch-labels() — Extract Labels

Parses Scratch text and returns a dictionary mapping label names to line numbers.

```
#let labels = scratch-labels("
repeat (4) #loop
  move (20) steps #step
  turn cw (90) degrees
end
")
// labels = ("loop": 1, "step": 2)
```

4.2 blockst-labels() — Query Labels

Queries globally collected labels from all rendered `scratch()` calls.

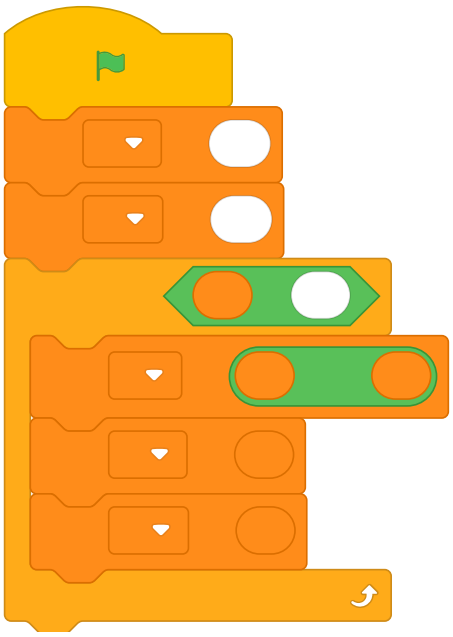
```
#blockst-labels("loop")   // → line number or "NaN"
#blockst-labels()         // → full label → line dictionary
```

4.3 blockst-register-labels() — Pre-register Labels

Register labels globally without rendering blocks. Useful when labels are needed before the first block output.

```
#blockst-register-labels("
repeat (4) #loop
  move (20) steps #step
end
")
```

Euclidean Algorithm (gcd label walkthrough)



Explanation of labeled lines

- **Line 4**: Loop condition `b != 0` controls termination.
- **Line 5**: Core rule: `r = a mod b`.
- **Line 7**: State update that advances to the next pair.

5 `@category` — Quick Color Defaults

Use `@category` prefix to force a block's category color, even without matching full localized syntax.

```
@motion         // → default motion block
@motion free text  // → unrecognized block in motion color
```

Available categories:

Token	Normalized	Default Block
`@motion`	motion	`move (10) steps`
`@looks`	looks	`say [Hello!]`
`@sound`	sound	`play sound [pop v]`
`@events`	events	`when green flag clicked`
`@control`	control	`repeat (10)`
`@sensing`	sensing	`ask [What's your name?] and wait`
`@operators`	operators	`(() + ())`
`@variable`	variables	`set [var v] to (0)`
`@list`	lists	`add (thing) to [list v]`
`@pen`	pen	`clear`
`@event`	events	(alias for `events`)
`@operator`	operators	(alias for `operators`)

When `@category` is followed by text that matches a known block in that category, the actual block is rendered:

```
@list add (12) to [my list v]    // → DATA_ADDTOLIST
@variable change [score v] by (1) // → DATA_CHANGEVARIABLEBY
```

If the text does not match any known block, the fallback is an `unrecognized` block in the forced category color.

6 Parsing API

6.1 scratch-parse() — Parse to AST

Parses Scratch text to an abstract syntax tree for programmatic use.

```
#scratch-parse(
  text,              // Scratch block text
  language: "en",    // locale for parsing
)
```

Returns a nested structure representing blocks, inputs, and bodies.

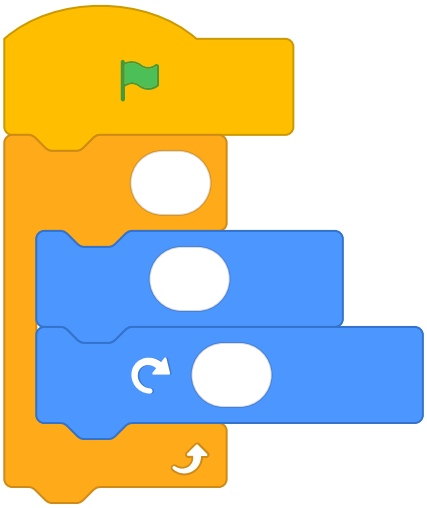
7 Markdown Code Blocks with raw-scratch

The `raw-scratch()` show rule converts scratch code fences into rendered blocks automatically.

```
#show: raw-scratch(language: "en")   // locale for block text
```

Then use scratch code fences in your document:

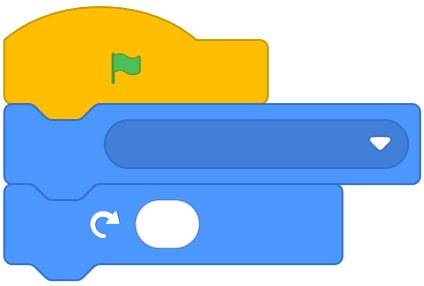
```
when green flag clicked
repeat (4)
  move (30) steps
  turn cw (90) degrees
end
```

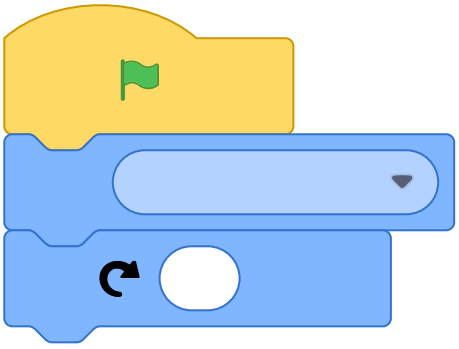


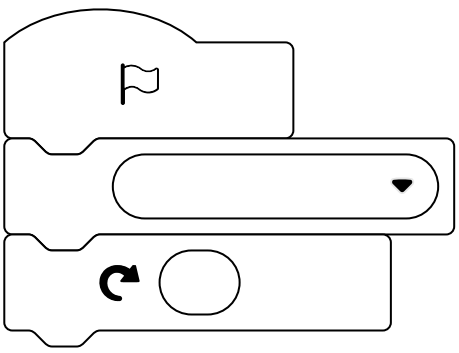
Optionally pass language: raw-scratch(language: „de“) in the show rule.

8 Theme and Styling Examples

```
#let script = "when green flag clicked
go to (random position v)
turn cw (30) degrees"

#blockst(inset-scale: 50%)[#scratch(script)]

#v(5mm)

#blockst(theme: "high-contrast")[#scratch(script)]

#v(5mm)

#blockst(theme: "print")[#scratch(script)]
```







9 Turtle Graphics / Executable Scratch

blockst includes an execution engine that runs Scratch programs visually — ideal for demonstrating program flow and pen drawing.

9.1 scratch-run Module

Import via the `scratch-run` module:

```
#import "@preview/blockst:0.3.0": scratch-run, set-scratch-run

#scratch-run.stage("...", scale: 2)
#scratch-run.grid("...", grid: true)
```

9.1.1 scratch-run.stage() — Stage Canvas

Renders pen drawing on a stage canvas, like the Scratch stage.

```
#scratch-run.stage(
  program,
  language: "en",      // locale for block text
  size: auto,          // (width, height) in pixels (default 480×360)
  scale: auto,         // drawing scale multiplier
  start: auto,         // (x: 0, y: 0, angle: 90)
  pen: auto,           // (down: false, color: ..., size: ...)
  background: auto,    // background color (e.g. white, black)
  cursor: auto,        // show/hide turtle cursor (default: true)
  border: auto,        // show/hide stage border (default: true)
)
```

9.1.2 scratch-run.grid() — Coordinate Grid

Renders the drawing on a Cartesian coordinate grid with optional axes and grid lines.

```
#scratch-run.grid(
  program,
  language: "en",      // locale for block text
  x: auto,             // view bounds (tuple, e.g. (-10, 10))
  y: auto,             // view bounds (tuple)
  step: auto,          // grid step size
  scale: auto,         // drawing scale multiplier
  start: auto,         // (x: 0, y: 0, angle: 90)
  pen: auto,           // (down: false, color: ..., size: ...)
  background: auto,    // background color
  axes: auto,          // show axis lines (default: false)
  grid: auto,          // show grid lines (default: false)
  grid-style: auto,    // grid line stroke (default: 0.5pt + gray)
  cursor: auto,        // show/hide turtle cursor (default: true)
  fit: auto,           // auto-fit view to drawing bounds
)
```

9.1.3 set-scratch-run() — Run Global Defaults

```
#set-scratch-run(
  scale: none,         // default scale for all runs
  start: none,         // (x: 0, y: 0, angle: 90)
  pen: none,           // (down: false, color: ..., size: ...)
  background: none,    // default background color
```

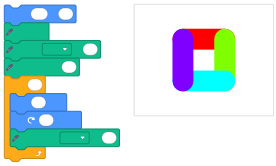
```
  cursor: none,        // show/hide turtle cursor (default: true)
  stage: none,         // (size: (480, 360), border: true)
  grid: none,          // (visible: false, axes: false, step: auto, style: auto)
)
```

9.1.4 Supported Execution Commands

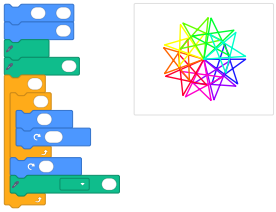
Category	Commands
Motion	`move`, `turn-right`, `turn-left`, `set-direction`, `go-to`, `set-x`, `set-y`, `change-x`, `change-y`
Pen	`pen-down`, `pen-up`, `set-pen-color`, `set-pen-size`, `change-pen-size`, `erase-all`, `stamp`, `set-pen-param`, `change-pen-param`
Variables	`set-variable`, `change-variable`, `variable`
Operators	`plus`, `minus`, `multiply`, `divide`, `modulo`, `random`, `round`, `greater`, `less`, `equals`, `op-and`, `op-or`, `op-not`
Control	`repeat`, `repeat-until`, `if-then`, `if-else`, `wait`
Looks	`say`, `think`
Shapes	`square`, `triangle`, `circle`, `star`, `spiral`
German	`gehe`, `drehe-rechts`, `drehe-links`, `setze-richtung`, `gehe-zu`, `stift-ein`, `stift-aus`, `setze-stiftfarbe-auf`, `setze-stiftdicke`, `setze-variable`, `aendere-variable`, `groesser`, `kleiner`, `gleich`, `und`, `oder`, `nicht`, `mal`, `geteilt`, `zufallszahl`

9.1.5 Complete Example

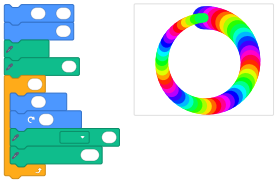
1) Hue Square



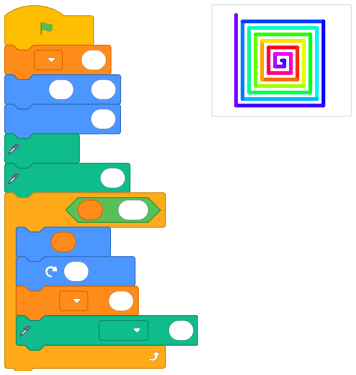
2) Star Rosette (Nested repeat)



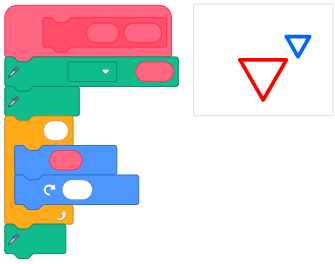
3) Colored Spiral (Hue + pen size)

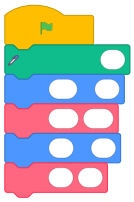


4) Conditional Loop

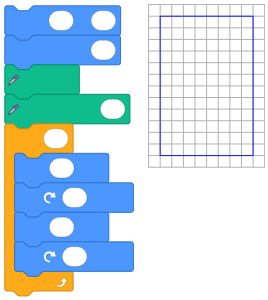


5) Custom Block: Triangle Pattern





6) Grid Preview (Axes + fixed bounds)



```
#let square-program = "
go to x: (-45) y: (45)
pen down
set pen [color v] to (0)
set pen size to (45)
repeat (4)
  move (90) steps
  turn cw (90) degrees
  change pen [color v] by (25)
end"

#set-scratch-run(
  stage: (size: (300, 240)),
  start: (x: 0, y: 0, angle: 90),
)

#grid(
  columns: (auto, auto),
  gutter: 6mm,
  [#scratch(square-program)],
  [#scratch-run.stage(square-program, scale: 2)],
)
```

10 SB3 Import API

blockst can read real Scratch 3 project files (`.sb3`) and extract scripts, variables, lists, images, and screen previews.

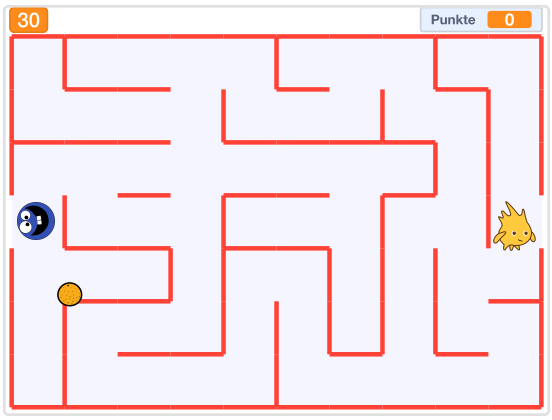
10.1 Basic Workflow

```
#let project = read("my-project.sb3", encoding: none)

#sb3.render-sb3-scripts(project, language: "en", target: "Sprite1")
```

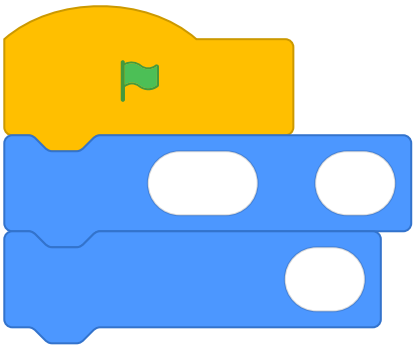
The `.sb3` file must be read as raw bytes (`encoding: none`).

10.2 render-sb3-scripts()

```
#sb3.render-sb3-scripts(
  sb3-bytes,                  // raw .sb3 file bytes (read with encoding: none)
  script-number: auto,        // global script index (1-based)
  target-script-number: auto, // script index within selected target
  target: auto,               // filter by target name ("Stage", sprite name, or auto for
all)
  sb3-plugin: auto,           // WASM plugin path (auto = bundled plugin)
  language: "en",             // locale for block text
  show-headers: auto,         // show target name header
  header-gap: 1.5mm,          // spacing between target name and scripts
  script-gap: 3mm,            // spacing between individual scripts
)
```



2. Sprite: Pacman - Script 2



10.3 render-sb3-variables()

```
#sb3.render-sb3-variables(
  sb3-bytes,                    // raw .sb3 file bytes
  target: auto,                 // filter by target name
  target-variable-name: auto,   // filter by variable name
  target-variable-number: auto,// variable index within target (1-based)
  sb3-plugin: auto,             // WASM plugin path
  language: "en",               // locale for display text
  show-target-headers: auto,    // show target name header
  target-gap: 2mm,              // spacing between targets
  item-gap: 0.8mm,              // spacing between variable items
)
```

10.4 render-sb3-lists()

```
#sb3.render-sb3-lists(
  sb3-bytes,                    // raw .sb3 file bytes
  target: auto,                 // filter by target name
  target-list-name: auto,       // filter by list name
  target-list-number: auto,     // list index within target (1-based)
  sb3-plugin: auto,             // WASM plugin path
  language: "en",               // locale for display text
  show-target-headers: auto,    // show target name header
  target-gap: 2mm,              // spacing between targets
  item-gap: 0.8mm,              // spacing between list items
)
```

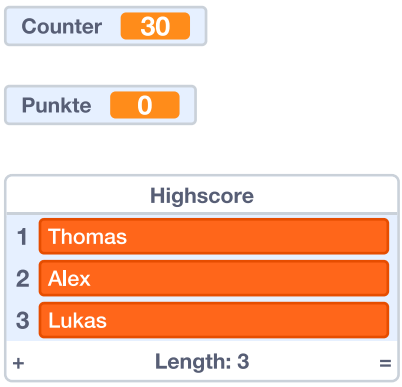
10.5 Standalone Monitors

10.5.1 list-monitor()

```
#list-monitor(
  name: "List",       // display name shown in header
  items: (),          // array of values to display
  width: 5.2cm,       // monitor width
  height: auto,       // auto-grows to fit content
  length-label: auto,// show length indicator (e.g. "length: 3")
)
```

10.5.2 variable-monitor()

```
#variable-monitor(
  name: "Variable",  // display name shown in header
  value: 0,          // current value to display
)
```



10.6 screen-preview()

```
#sb3.sb3-screen-preview(
  sb3-bytes,           // raw .sb3 file bytes
  width: 480,          // stage rendering width in pixels
  height: 360,         // stage rendering height in pixels
  unit: 1,             // size multiplier (2 = double size)
  background: none,    // override background color
  show-border: true,   // show stage border
  show-backdrop: true, // show costume/backdrop image
  monitor-scale: 1.5,  // scale factor for variable/list overlays
  language: auto,      // locale for monitor text
)
```

Renders a static Scratch stage preview with sprites, backdrop, and monitors.

10.7 Image Helpers

```
// List all image assets in the project
#sb3.sb3-image-assets-catalog(sb3-bytes, target: auto)

// Render a single image asset
#sb3.sb3-image(
  sb3-bytes,          // raw .sb3 file bytes
```

```
  target: auto,       // filter by sprite/stage name
  image-number: auto, // global image index (1-based)
  target-image-number: auto, // image index within target (1-based)
  image-name: auto,   // filter by image name (e.g. "costume1")
  width: auto,        // output width (auto = original size)
  height: auto,       // output height
)
```

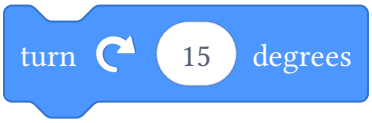
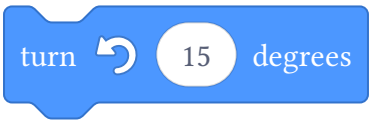
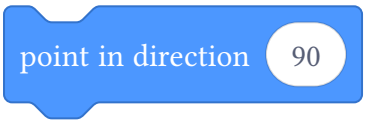
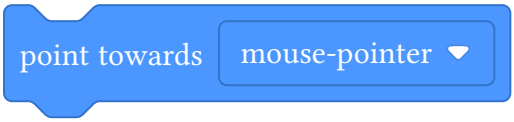
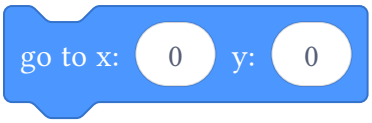
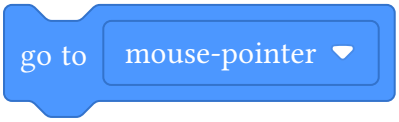
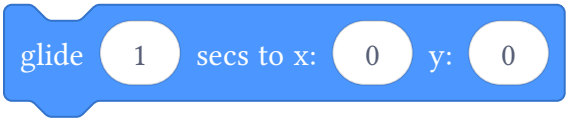
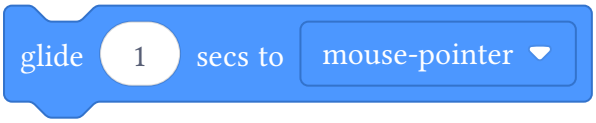
10.8 Catalog Helpers

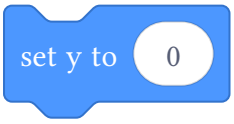
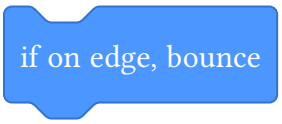
```
// Grouped script metadata (targets, scripts, blocks count)
#sb3.sb3-scripts-catalog(sb3-bytes)

// Target states (variables, lists, and sprite properties)
#sb3.sb3-state-catalog(sb3-bytes)

// Convert a specific SB3 script to Scratch text
#sb3.sb3-bytes-to-scratch-text(
  sb3-bytes,          // raw .sb3 file bytes
  script-number: auto,// global script index (1-based)
  language: "en",     // locale for output text
)
```

11 Catalog

The catalog shows every block in each Scratch 3 category, rendered live.

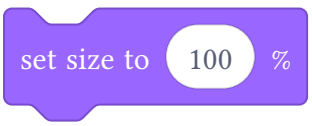
11.1 Motion

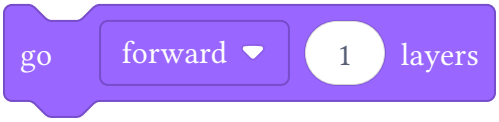
Block	Code
	`move (10) steps`
	`turn cw (15) degrees`
	`turn ccw (15) degrees`
	`point in direction (90)`
	`point towards [mouse-pointer v]`
	`go to x: (0) y: (0)`
	`go to [mouse-pointer v]`
	`glide (1) secs to x: (0) y: (0)`
	`glide (1) secs to [mouse-pointer v]`
	`change x by (10)`

Block	Code
	`set x to (0)`
	`change y by (10)`
	`set y to (0)`
	`if on edge, bounce`
	`set rotation style [left-right v]`
	`(x position)`
	`(y position)`
	`(direction)`

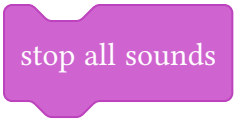
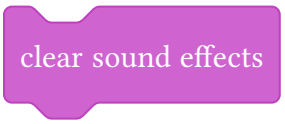
11.2 Looks

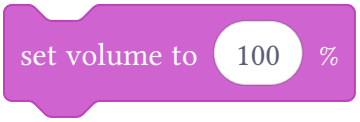
Block	Code
	`say [Hello!] for (2) seconds`
	`say [Hello!]`
	`think [Hmm...] for (2) seconds`
	`think [Hmm...]`

Block	Code
	`show`
	`hide`
	`switch costume to [costume1 v]`
	`next costume`
	`switch backdrop to [backdrop1 v]`
	`next backdrop`
	`change [color v] effect by (25)`
	`set [color v] effect to (0)`
	`clear graphic effects`
	`change size by (10)`
	`set size to (100) %`
	`go to [front v] layer`

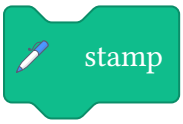
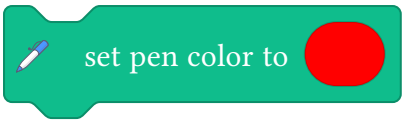
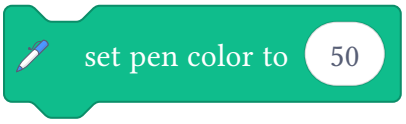
Block	Code
	`go [forward v] (1) layers`
	`(costume [number v])`
	`(backdrop [number v])`
	`(size)`

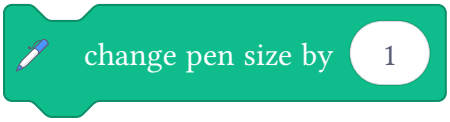
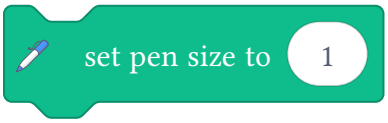
11.3 Sound

Block	Code
	`play sound [pop v] until done`
	`start sound [pop v]`
	`stop all sounds`
	`change [pitch v] effect by (10)`
	`set [pitch v] effect to (100)`
	`clear sound effects`
	`change volume by (-10)`

Block	Code
	`set volume to (100) %`
	`(volume)`

11.4 Pen

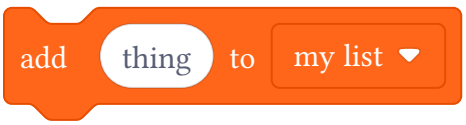
Block	Code
	`erase all`
	`stamp`
	`pen down`
	`pen up`
	`set pen color to [#ff0000]`
	`change pen color by (10)`
	`set pen color to (50)`
	`change pen shade by (10)`
	`set pen shade to (50)`

Block	Code
	`change pen size by (1)`
	`set pen size to (1)`

11.5 Variables

Block	Code
	`set [my variable v] to (0)`
	`change [my variable v] by (1)`
	`show variable [my variable v]`
	`hide variable [my variable v]`
	`(my variable)`

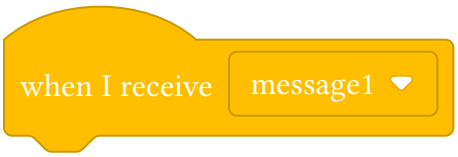
11.6 Lists

Block	Code
	`add [thing] to [my list v]`
	`delete (1) of [my list v]`
	`delete all of [my list v]`

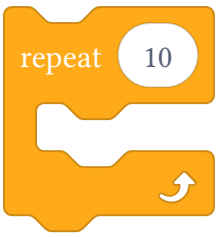
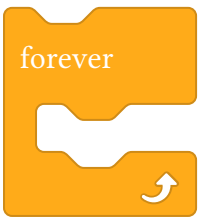
Block	Code
	`insert [thing] at (1) of [my list v]`
	`replace item (1) of [my list v] with [thing]`
	`(item (1) of [my list v])`
	`(item # of [thing] in [my list v])`
	`(length of [my list v])`
	`<[my list v] contains [thing]?>`
	`show list [my list v]`
	`hide list [my list v]`

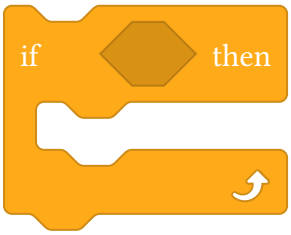
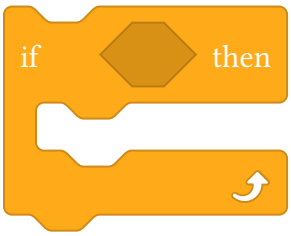
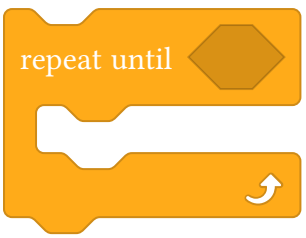
11.7 Events

Block	Code
	`when green flag clicked`
	`when [space v] key pressed`
	`when this sprite clicked`

Block	Code
	`when backdrop switches to [backdrop1 v]`
	`when [loudness v] > (10)`
	`when I receive [message1 v]`
	`broadcast [message1 v]`
	`broadcast [message1 v] and wait`

11.8 Control

Block	Code
	`wait (1) seconds`
	`repeat (10)`
	`end`
	`forever`

Block	Code
	`end`
	`if <> then`
	`end`
	`if <> then`
	`else`
	`end`
	`wait until <>`
	`repeat until <>`
	`end`
	`stop [all v]`

Block	Code
	`when I start as a clone`
	`create clone of [myself v]`
	`delete this clone`

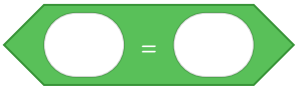
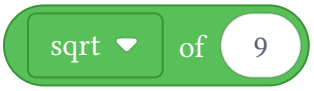
11.9 Sensing

Block	Code
	`<touching [mouse-pointer v]?>`
	`<touching color [#ff0000]?>`
	`<color [#ff0000] is touching [#00ff00]?>`
	`(distance to [mouse-pointer v])`
	`ask [What's your name?] and wait`
	`(answer)`
	`<key [space v] pressed?>`
	`<mouse down?>`
	`(mouse x)`
	`(mouse y)`

Block	Code
	`(loudness)`
	`(timer)`
	`reset timer`
	`([x position v] of [Sprite1 v])`
	`(current [year v])`
	`(days since 2000)`
	`(username)`

11.10 Operators

Block	Code
	`(() + ())`
	`(() - ())`
	`(() * ())`
	`(() / ())`
	`(pick random (1) to (10))`
	`<() > ()>`
	`<() < ()>`

Block	Code
	`<() = ()>`
	`<> and <>`
	`<> or <>`
	`not <>`
	`(join [apple] [banana])`
	`(letter (1) of [apple])`
	`(length of [apple])`
	`([apple] contains [a]?)`
	`(() mod ())`
	`(round ())`
	`([sqrt v] of (9))`

12 Contributing

Contributions are welcome: bug reports, missing blocks, parser improvements, rendering polish, docs, and new localizations.

- **Repository:** github.com/Loewe1000/blockst
- **License:** MIT

13 API Reference

The reference is generated from the `///` comments in the source. It covers the public rendering API in `libs/scratch/api.typ`, which `lib.typ` and `package.typ` re-export unchanged.

13.1 `blockst`

```
blockst(
  theme: auto,
  scale: auto,
  font: auto,
  line-numbering: auto,
  line-numbers: auto,
  line-number-start: auto,
  line-number-first-block: auto,
  line-number-gutter: auto,
  inset-scale: auto,
  language: auto,
  spacing: 1.5em,
  body,
)
```

Optional container for grouped blocks with theme/scale override. Only needed when a specific group should differ from global settings.

Name	Vorgabe
`theme`	`auto`
`scale`	`auto`
`font`	`auto`
`line-numbering`	`auto`
`line-numbers`	`auto`
`line-number-start`	`auto`
`line-number-first-block`	`auto`
`line-number-gutter`	`auto`
`inset-scale`	`auto`
`language`	`auto`
`spacing`	`1.5em`
`body`	–

13.2 `blockst-labels`

```
blockst-labels(name)
```

Read globally collected line labels from all previously rendered `scratch()` blocks. With a name: `#blockst-labels("start", default: "?")`. Without a name: returns the full label-to-line dictionary.

13.3 `blockst-register-labels`

```
blockst-register-labels(text, language: "en")
```

Register labels globally without rendering blocks. Useful when labels are needed before the first `#scratch()` output appears.

Name	Vorgabe
`text`	–
`language`	`"en"`

13.4 `raw-scratch`

```
raw-scratch(..args)
```

Enable scratch code blocks in raw text:

```
#show: raw-scratch()
```

With language: `#show: raw-scratch(language: "de")`.

13.5 `scratch`

```
scratch(
  text,
  language: auto,
  theme: auto,
  scale: auto,
  font: auto,
  line-numbering: auto,
  line-numbers: auto,
  line-number-start: auto,
  line-number-first-block: auto,
  line-number-gutter: auto,
  inset-scale: auto,
)
```

Render scratch blocks from text. Works standalone or inside `#blockst[...]`. Supports 26 languages: en, de, fr, es, it, pt, nl, pl, ru, ja, …

Name	Vorgabe
`text`	–
`language`	`auto`
`theme`	`auto`
`scale`	`auto`
`font`	`auto`
`line-numbering`	`auto`
`line-numbers`	`auto`
`line-number-start`	`auto`
`line-number-first-block`	`auto`
`line-number-gutter`	`auto`
`inset-scale`	`auto`

13.6 `scratch-execute`

```
scratch-execute(text, language: "en")
```

Execute scratch text, producing scratch-run commands.

Name	Vorgabe
`text`	–

Name	Vorgabe
`language`	`"en"`

13.7 `scratch-labels`

```
scratch-labels(text, language: "en")
```

Parse scratch text and return a dictionary mapping `#labels` to rendered line numbers.

Name	Vorgabe
`text`	–
`language`	`"en"`

13.8 `scratch-parse`

```
scratch-parse(text, language: "en")
```

Parse scratch text to AST (for programmatic use).

Name	Vorgabe
`text`	–
`language`	`"en"`

13.9 `set-blockst`

```
set-blockst(
  theme: none,
  scale: none,
  stroke-width: none,
  font: none,
  line-numbering: none,
  line-numbers: none,
  line-number-start: none,
  line-number-first-block: none,
  line-number-gutter: none,
  inset-scale: none,
  language: none,
)
```

Global settings applied to all scratch() and sb3 calls.

Name	Vorgabe
`theme`	`none`
`scale`	`none`
`stroke-width`	`none`
`font`	`none`
`line-numbering`	`none`
`line-numbers`	`none`
`line-number-start`	`none`
`line-number-first-block`	`none`
`line-number-gutter`	`none`
`inset-scale`	`none`
`language`	`none`