scholatex

Print-ready teaching worksheets, the easy way

A small tag-based language for handouts and exercises,
built on LuaLaTeX

Gérard Dubard

Version 1.1

Contents

1 Introduction

scholatex is a small markup language for producing print-ready teaching materials — worksheets, exercise sheets, handouts, short assessments — without writing LaTeX. It is not a general-purpose replacement for LaTeX; it is a focused, consistent little language for the documents a teacher actually makes: a few framed exercises, a table of results, a couple of simple formulas, an image or two, laid out cleanly on a page meant to be printed and handed out.

It compiles to LuaLaTeX, so the output has full LaTeX typesetting quality, but the syntax is meant to be read and edited in minutes by someone who does not know LaTeX: no `\begin{tabular}{|c|c|}`, no counting ampersands, no hunting for the package that draws a coloured box. A single tag syntax covers text, tables, images, simple maths, framed boxes, lists and full-page named-area layouts — the building blocks of a worksheet. The full power of LaTeX stays one escape hatch away (any raw command still works), but for everyday teaching documents you rarely need it.

It is built for the people LaTeX usually loses: teachers preparing classroom materials, and anyone who wants a clean printable sheet without climbing the LaTeX learning curve.

You write a `.tex` file with a tiny, readable syntax between `\begin{document}` and `\end{document}`. At compile time the `scholatex` class reads that body, transpiles it to LaTeX, and typesets it. Nothing to install beyond a LuaLaTeX distribution.

```
% !TeX program = lualatex
\documentclass[margins=20, size=12, imgdir=IMG]{scholatex}
\begin{document}

<navy b 18pt c>My first scholatex document

This is a normal paragraph. <b>{Bold}, <i>{italic}, <red>{red}.

<box line:Navy fill:AliceBlue radius:3 title:{A framed note}>{
Boxes, tables, images and maths all use the same tag syntax.
}

\end{document}
```

2 The one rule

Everything is a **tag**: `<attributes>` followed either by `{content}` (inline) or by a line that ends in `{` (a block). Attribute words follow a single case convention:

Form	Meaning	Examples
`lowercase`	a short keyword or base colour	`b`, `i`, `c`, `red`, `2tab`
`CamelCase`	an extended CSS colour (151 of them)	`SteelBlue`, `Crimson`
`UPPERCASE`	a font name	`DEJAVU SANS`

The order of words inside a tag never matters — emission is always normalised (page break, then vertical skips, then alignment, then tabs, then styles).

3 Class options

Set in `\documentclass[...]{scholatex}`:

Option	Default	Meaning
`margins`	`20`	`N` (all sides) or `{top,right,bottom,left}` in mm
`font`	`Latin Modern Roman`	main text font
`mathfont`	`Latin Modern Math`	math font
`size`	`11`	base font size in pt
`imgdir`	`img`	folder(s) searched for bare image names; accepts a comma-separated list, e.g. `{IMG, IMAGES/PNG}`
`tabwidth`	`8`	width of one tab, in mm
`lineheight`	`8`	height of one skipped line, in mm
`scriptscale`	`100`	scale (%) of `up`/`down` scripts
`padding`	`2`	inner padding (mm) between a box/grid-area frame and its content
`lang`	`fr`	decimal separator: `fr` (comma) or `en` (point); affects both literal decimals in maths and interpolated values
`untrusted`	`false`	when `true`, runs the document's Lua in a restricted sandbox (see §15)

Headings carry no extra vertical space of their own, and no built-in colour: to style a heading (colour, weight, size, or a line skip before it), fold a heading keyword into an alias and use it (see §5).

4 Text attributes

Inline styles — `b` bold, `i` italic, `u` underline, `emph`, `tt` typewriter, `sf` sans-serif, `sc` small caps.

Colours — short keywords (`red`, `blue`, `green`, `navy`, `orange`, `purple`, `teal`, `brown`, `gray`, `pink`, ...) or any of the 151 CSS / svgnames colours in CamelCase (`Tomato`, `SteelBlue`, `ForestGreen`, ...). These are exactly the colours xcolor provides under `svgnames`: the 147 CSS Color Module names plus the four X11 extras `LightGoldenrod`, `LightSlateBlue`, `NavyBlue` and `VioletRed`. Names are case-sensitive — write `Seashell`, not `SeaShell`. The `README` groups all 151 by family for browsing.

Fonts and sizes — a font name in CAPITALS (`<DEJAVU SANS>{...}`); sizes as `Npt` or `Npx` (`<14pt>{...}`).

Alignment — `l` left, `c` centre, `r` right, `j` justified (the same letters as table columns).

Tabs and skips (the number is always a prefix) — `Ntab` indents the first line by N tabs; vertical skips follow singular/plural agreement: `line` or `1line` skips one line, `2lines`, `3lines`, ... skip several. Bare `tab` = `1tab`.

Scripts — `upN` raises text by N mm (superscript-like), `downN` lowers it (subscript-like): `x<up4>{2}`, `H<down2>{2}O`.

Page break — `nextpage`.

Section headings — `section`, `subsection`, `subsubsection` give the three heading levels, numbered automatically:

```
<section>First topic
<subsection>A detail
<subsubsection>A finer point
```

renders as '1 First topic', '1.1 A detail', '1.1.1 A finer point'. A heading can also be written as a **block** carrying a `title:` option, with its body in braces:

```
<subsection title:{A heading}>{
<tab>This body paragraph is indented on its first line.
}
```

A **table of contents** is printed with `<tableofcontents>`; give it a title in braces:

```
<tableofcontents>{Table of contents}
```

Combine anything in one tag:

```
<nextpage 2lines 2tab j navy b>{A new page, a two-line skip, a two-tab
indent, justified, navy, bold --- all before the first letter of the text.}
```

5 Aliases and macros — the factoring tool

This is what makes scholatex scale. Define a style **once**, at the top of the document, then name it everywhere instead of repeating its attributes. One edit at the definition restyles the whole document.

```
let title = <navy b 18pt c>          % style alias
let h1    = <navy b section>         % a heading style, reusable
let p     = <tab>                    % a standard indented paragraph
<title>My heading
<h1>First topic
<p>{ A paragraph, indented and justified, named not described. }

let n = 7                            % value, usable in #{...}
Seven squared is #{n*n}.

let greet{name} = Hello #name!       % text macro with parameters
```

Change `let h1 = <navy b section>` to `<ForestGreen b section>` once and **every** first-level heading follows — the single point of control that keeps a long document consistent.

6 Tables

Columns are declared in brackets, **one two-letter placement code per column**. The first letter is vertical (`t` top, `m` middle, `b` bottom), the second horizontal (`l`, `c`, `r`). So `mc` is middle-centre, `br` bottom-right. Columns share the page equally; `N:` before the code fixes a width in mm (`[40:tl, 30:mc]`). Cells are separated by `|`, rows by line breaks, and `\\` breaks a line inside a cell.

```
<table [mc, tl, br] borders header>{
Figure | Formula | Value
<img 25>{cat.png} | $1/2 + 1/3$ | $5/6$
}
```

`borders` draws rules; `header` bolds the first row. `gap:N` sets the horizontal spacing between columns, in mm.

A cell may span several columns with `colspan:N`, or several rows with `rowspan:N` ($N \geq 2$). For a `rowspan`, each cell it covers on the lines below is marked with a lone `.`.

```
<table [mc, ml, mc, mc] borders header gap:3>{
<colspan:4 mc>{Term report}
Day | Subject | Mark | Coef.
<rowspan:2 mc>{Monday} | Maths | 15 | 4
. | French | 12 | 3
}
```

Colour options act **inside** the table, on its cells and rules — `fill:` colours every body cell, `line:` the rules, `text:` the text colour, and `headerfill:` / `headertext:` the header row. These are not box options: a table has no `radius:`, `title:` or outer frame. To frame a table, wrap it in a `<box>`.

7 Images

```
<img>{chat.png}        % fills the available width (the cell or line)
<img 25>{chat.png}     % 25 mm wide
<img 25x15>{chat.png}  % fits a 25x15 mm box, ratio preserved
```

A bare name is searched in each folder of `imgdir` in turn, then at the project root. An explicit path always works, with or without `./`.

8 Maths

Wrap maths in `$...$`. A small mini-language keeps it light:

You write	You get
`*`	$\times$
`+-`	$\pm$
`<= >= !=`	$\leq \quad \geq \quad \neq$
`a/b`	fraction (chained `a/b/c` reads as `(a/b)/c`)
`x^2  x_i`	power / index
`sqrt(2)`	$\sqrt{2}$
`sum(i=1, n) i`	$\sum$ with bounds, display style
`prod(k=1, n) k`	$\prod$ with bounds, display style
`int(x) f(x)`	$\int f(x)\,\mathrm{d}x$ (differential added)
`int(x=a, b) f(x)`	$\int_a^b f(x)\,\mathrm{d}x$
`contourint(C) f(z)`	$\oint$ contour integral
`pvint(x=a, b) f(x)`	p.v.$\int$ principal value
`meanint(x=a, b) f(x)`	$⨍$ average integral
`dy/dx`, `df/dx`	derivative, upright d (ISO)
`sin(x) cos(x) ln(x)`	upright function names
`abs(x)`	$\lvert x\rvert$
`norm(v)`	$\lVert v\rVert$
`vec(AB)`	$\overrightarrow{AB}$
`lim(x->0) f(x)`	limit, target under the word
`partial`, `nabla`	∂, ∇
`pi`, `alpha`, …	Greek letters
`inf`	∞

The helpers nest, so the secondary-school staples come for free: `norm(vec(AB))` is the norm of a vector, `vec(AB) + vec(BC) = vec(AC)` is Chasles' relation.

8.1 Operators with an index

`sum`, `prod` and `lim` carry their index in `(...)` and set their whole expression in display style, so a fraction in the body keeps full size and a limit's target sits *under* the word, as on a blackboard. A `sum`/`prod`/`lim` body runs to the end of the formula or to the first `=`, so the right-hand side of an identity stays outside the operator:

```
$sum(i=1, n) i = n(n+1)/2$
$lim(x->0) sin(x)/x = 1$
$lim(x->+inf) 1/x$
```

8.2 Functions and trigonometry

Function names are set upright automatically — no backslashes: `sin cos tan cot sec csc`, the inverses `arcsin arccos arctan`, the hyperbolics `sinh cosh tanh coth`, and `ln log exp det dim gcd deg ker arg max min sup`. A name glued to `(...)` takes its argument as one atom, so fractions and powers behave:

```
$sin(x)^2 + cos(x)^2 = 1$
$tan(x) = sin(x)/cos(x)$
```

8.3 Integrals

The integral family names its variable and bounds in the head `(...)` and captures the integrand as its body; the differential is appended for you. No bounds gives a primitive, a comma-separated pair a definite integral:

```
$int(x) f(x)$              % primitive
$int(x=a, b) f(x)$         % definite
```

Separate several domains with `;` for multiple integrals: the count of domains chooses $\int$, $\iint$ or $\iiint$, and the differentials come out in reverse order (the Fubini convention). A single named domain is a region integral:

```
$int(x=a, b ; y=c, d) f(x,y)$
$int(D) f$
```

Three named integrals complete the family: `contourint(C) f(z)` a contour integral $\oint$, `pvint(x=a, b) f(x)` a Cauchy principal value, and `meanint(x=a, b) f(x)` the average (normalised) integral.

8.4 Derivatives

A Leibniz derivative is written as the fraction it is, and the differential `d` is set upright (ISO 80000-2), matching the `d` of the integrals. The romanisation triggers *only* when both sides of the fraction carry the differential, so a variable named `d` is never disturbed: `d/2` stays the fraction d over 2.

```
$dy/dx$                 % upright d
$(d^2 y)/(dx^2)$        % second derivative
$dy/dx + y = 0$         % differential equation
```

Partial derivatives use `partial` (∂); parenthesise each side so the fraction groups correctly:

```
$(partial f)/(partial x)$
$(partial u)/(partial t) = (partial^2 u)/(partial x^2)$
```

Inject a computed value with `#{expr}` (or `#name`), including inside maths: `$#k^2$`. Decimal numbers follow the `lang` option.

8.5 Maths blocks

Matrices and systems are blocks: **one line is one row**, and inside a matrix `;` separates the entries.

```
<matrix>{
1 ; 2 ; 3
4 ; 5 ; 6
}
```

`<matrix>` draws parentheses, `<det>` the vertical bars of a determinant, `<bmatrix>` square brackets. A single `|` inside a row draws the separation bar of an **augmented matrix**, at the column where you type it; it is allowed on `matrix` and `bmatrix`, never on `det`:

```
<bmatrix>{
2 ; 1 | 7
1 ; -1 | 1
}
```

A `<system>` stacks equations under a brace and aligns them on the first relational operator:

```
<system>{
2x + 3y = 7
x - y = 1
}
```

9 Lists

`<list:STYLE>{ ...}` makes a list; the style is required and comes right after the name. One item per non-empty line — no item tag. A `<list:...>` written under an item becomes its sub-list, nested as deep as you like.

```
<list:disc>{
Fruits
<list:circle>{
apples
pears
}
Vegetables
}
```

Styles — bullets: `none` `disc` `circle` `square`; numbered: `decimal` `alpha` `ALPHA` `roman` `ROMAN` (the case of the keyword sets the case of the letters); checkboxes: `check`.

10 Boxes

```
<box line:Crimson fill:MistyRose radius:4 title:{A note}>{
Content here.
}
```

Options: `line:` frame colour, `fill:` background, `text:` text colour, `radius:N` rounded corners (mm), `width:N` or `width:N%`, `boxrule:N`, `boxsep:N`, `break:yes` (allow page breaks), `title:{...}`, `titlefill:`, `titletext:`. A line containing only `---` splits a box into an upper and a lower region.

`<row gap:N>{ ...}` lays its child boxes side by side, with equal widths and equalised heights. A `row` accepts only `<box>` children.

A box also takes a **two-letter placement code** (`tl tc tr ml mc mr bl bc br`, default `tl`) that positions its content; the vertical part needs a `height:` to have room to act.

11 Grid (named-area layout)

For full-page layouts, `<grid>` borrows CSS Grid's named-area idea. A `template:[ ...]` of quoted rows draws the layout; each word names the cell at that position. A name repeated **horizontally** spans columns; repeated **vertically** it spans rows. A dot `.` is an empty cell. Each name must form a solid rectangle.

```
<grid template:[
  "title  title  logo"
  "intro  info   logo"
  "body   body   body"
] gap:4>{
  <area title>{ <red b 16pt>Maths assessment }
  <area logo >{ <img>{blason.png} }
  <area intro>{ Instructions: no calculator. }
  <area info >{ Name: \\ First name: }
  <area body line:Navy fill:AliceBlue radius:2 title:{Exercise 1}>{
    Solve the equation...
  }
}
```

An area can be framed like a box by giving it the same options (`line:`, `fill:`, `radius:`, `title:`). An area with no options stays invisible (a bare cell). The grid itself takes `width:` (a percentage of the text width or a millimetre value) and `height:` (a millimetre value fixing the total height).

12 Block aliases

Define a reusable component once; `#param` placeholders are filled at the call site, and the call-site body becomes the block content.

```
let card{title, frame} = <box title:{#title} line:#frame radius:2>

<card First, Crimson>{
Called with two arguments. No box options to repeat.
}
<card Second, Navy>{
Same component, different look.
}
```

13 Control flow

```
for n in 1..3 {            % numeric range
<c navy b>Sheet #n
}

for f in [chat.png, chien.png] {   % explicit list
<img 16>{#f}
}

if score >= 10 {
<green>Passed.
} else {
<red>Try again.
}
```

Loops and conditions work in the document body, inside boxes, and inside table bodies. The loop variable interpolates everywhere via `#`.

14 Escapes

Literal special characters: `\< \> \{ \} \#`. The characters `_ & % ~` are escaped automatically. A double backslash `\\` is a line break.

A line whose first non-space character is `%` is a **comment** and is dropped. A bare `#` not followed by a name or `{...}` is a literal `#`: only `#name` and `#{expr}` interpolate.

15 Security

scholatex evaluates `let name = expr`, `#{expr}` and the conditions of `for`/`if`/`while` as Lua at compile time, so by default a document can run arbitrary code — exactly like `\directlua`.

Setting `untrusted=true` runs that Lua in a restricted environment: only pure, side-effect-free names are visible (the maths the document needs plus `string`/`table` helpers), while `os`, `io`, `package`, `require`, `load`, `debug` and `setmetatable` are absent. A blocked access stops the compile with a clear message; a runaway loop is aborted by an instruction-count ceiling.

Scope. `untrusted` hardens the scholatex expression layer only. It does *not* sandbox LuaLaTeX as a whole: a hostile `.tex` can still call `\directlua`, `\write18` (with shell-escape), `\input`, and so on. To compile a whole `.tex` you do not trust, run `lualatex` **without** `--shell-escape`, ideally inside a container.

16 Examples

The `examples/` folder contains three self-contained, fully commented documents that together exercise every feature:

File	Covers
`01-text-style.tex`	case rule, styles, colours, fonts, sizes, alignment, aliases, TOC
`02-containers.tex`	tables, boxes and the named-area grid
`03-math.tex`	mini-language, operators, integrals, trig, derivatives, matrix/system blocks

Compile any of them with `lualatex <file>.tex` from the `examples/` folder. The image folders `IMG/` and `IMAGES/PNG/` ship alongside them; `02-containers.tex` uses `imgdir={IMG, IMAGES/PNG}` to show that bare image names are resolved across several directories.

17 Diagnostics

Errors point at the source line, e.g. `scholatex: line 12: unknown tag attribute: 'xyz'`. Defining an alias whose name is a built-in (`let section = ...`, `let tab = ...`) prints a warning: the built-in always wins, so the alias would be silently dead — pick a different name.