

chronos

Clea F. Rees*

v0.9.4 (SVN 11797)

Abstract

`chronos` is a \LaTeX 2 ϵ package, based on `PGF/TikZ`, for typesetting timelines or chronologies. Externalisation is supported out-of-the-box with `memoize`. The package developed from two sources: first, the creation of a timeline for use in teaching¹ and, second, questions on tex.stackexchange.com concerning obstacles encountered in using existing packages. This package might be considered an attempt to use the former to partially remedy the latter. It also means both the code and the user-interface contain strange and tangled regions where the wild errors may grow.

*Bug tracker: codeberg.org/cfr/chronos/issues | Code: codeberg.org/cfr/chronos | Mirror: github.com/cfr42/chronos
¹See [this answer on TeX StackExchange](#) or [view the PDF](#).

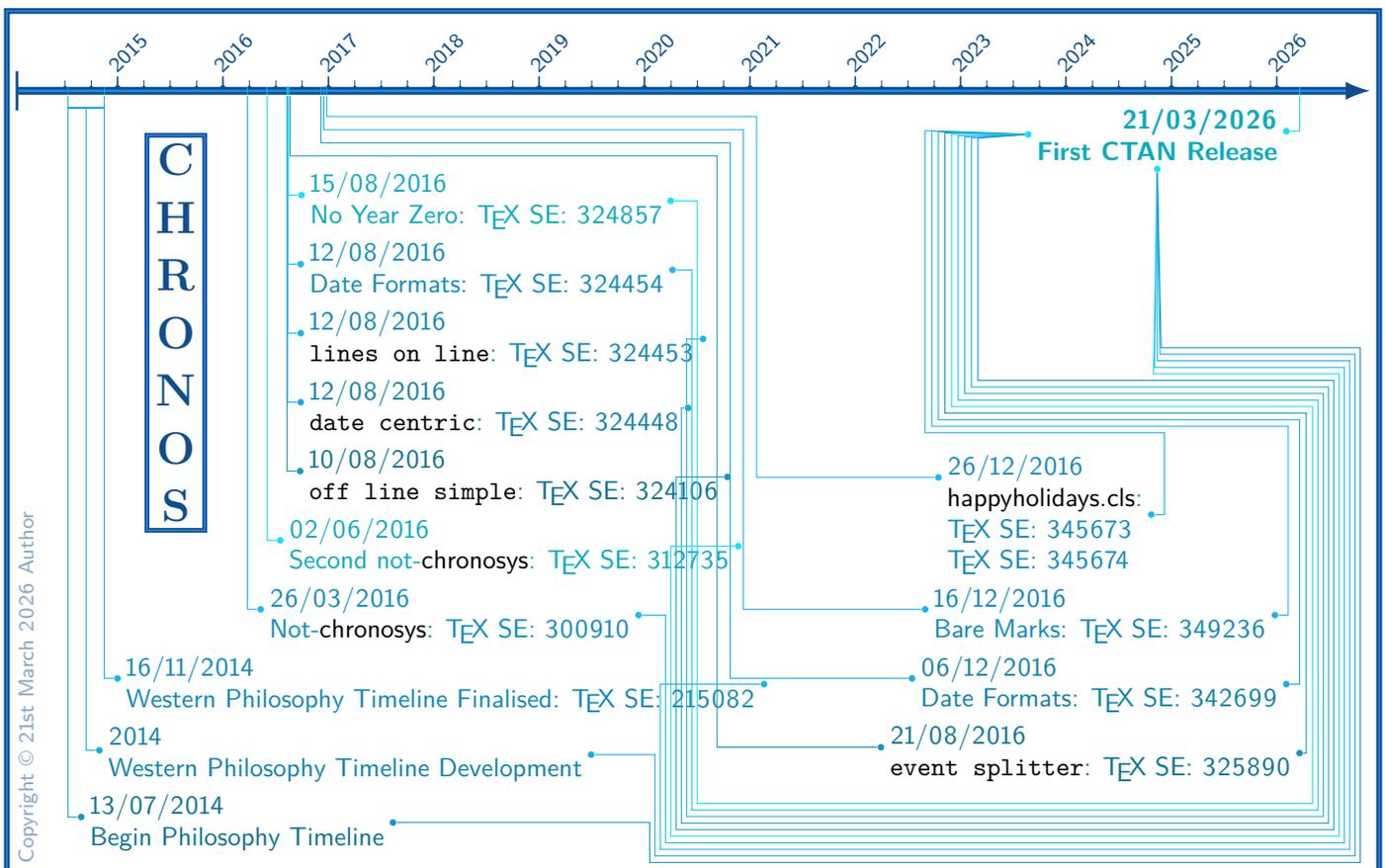


Figure 1: Chronos development: a chronos timeline (sections 6 and 8.4) with chronos style blues below (section 7.1.2) and custom styles tag left, tag post and tag right (section 13.3).

Contents

1	Raison d'être	3
2	Caveats, Assumptions & Limitations	3
3	Typesetting a Timeline	5
4	Loading the Package	10
5	Invocation	11
6	Chronos Anatomy	12
6.1	Chronos Timeline	12
6.2	Chronos Additional Element Types	12
6.2.1	Primary Types	14
6.2.2	Secondary (Sub-)Elements	15
6.3	Chronos Coordinate and Node Names	15
6.4	Chronos Layers	15
7	Chronos Schemes and Styles	17
7.1	Chronos Styles	17
7.1.1	'On Line' Styles	17
7.1.2	'Off Line' Styles	19
7.1.3	'No Year' Styles	21
7.2	Chronos Colour Schemes	24
8	Configuration	29
8.1	Documentation Notes	30
8.1.1	Font Conventions	30
8.1.2	Keys and Values	31
8.1.3	Key Specifications	31
8.1.4	Syntax Notes	32
8.1.5	Dimension Notes	33
8.1.6	Date Specification Notes	34
8.1.7	Colour Notes	34
8.2	Dates	34
8.2.1	Input	34
8.2.2	Output	35
8.2.3	The Problem of the Non-Existent Year	38
8.3	Basic Colours	40
8.4	Timeline	41
8.4.1	Timeline Dates	41
8.4.2	Timeline Dimensions	42
8.4.3	Timeline Marks and Years	45
8.4.4	Timeline Fonts	50
8.4.5	Timeline Colours	51
8.4.6	Timeline Style	52
8.5	Frame	53
8.6	Placing Things: Levels & Coordinates	54
8.6.1	Levels	54
8.6.2	Chronos Coordinates	55
8.6.3	Miscellaneous	55
8.7	Headings	55
8.7.1	Example	56
8.7.2	Headings Configuration	57
8.8	Colours	57

8.8.1	Colour Rotation	58
8.8.2	Using Colours	58
8.8.3	Colour Lists	59
8.8.4	Simple Colour Names	60
9	Adding Elements to the Timeline	61
9.1	Adding Connectable Elements	61
9.1.1	Timeline-Connectable Elements	61
9.1.2	Adding Other Connectable Elements	64
9.2	Adding Non-Connectable Elements	65
9.3	Additional Elements: Local Configuration	67
9.4	Additional Elements: Local/Global Configuration	72
9.4.1	Specialist Fonts for Text Tags	76
9.5	Additional Elements: Global Configuration	77
9.6	Adding Connections, Using Colours and Accessing Styles	82
10	Drawing on Chronos Layers	83
11	Externalising Chronos Timelines with Memoize	84
12	Deferring Code	85
12.1	Additional TikZ	85
13	Custom Schemes and Styles	86
13.1	Defining Chronos Colour Schemes	86
13.1.1	How Colour Schemes are Processed	88
13.2	Defining Chronos Styles	88
13.2.1	How (Not) to Customise Colours	90
13.2.2	How to Rotate Years	92
13.2.3	Hashes	93
13.2.4	Timeline Arrow	94
13.2.5	Styles and Automemoization	96
13.3	Defining Styles for Additional Elements	96
14	Debugging	98
15	Compatibility	102
15.1	Compatibility with Code from T _E X SE Answers	103
Index		106

1 Raison d'être

Chronos aims to make it easy

- to specify timelines covering from days to centuries;
- to customise a timeline's appearance using the standard key-value syntax familiar to users of TikZ;
- to define new timeline styles in a straightforward manner;
- to utilise a range of timeline styles provided out-of-the-box, including some based on those offered by other packages and/or featured on tex.stackexchange.com.

2 Caveats, Assumptions & Limitations

First, the caveats . . .

Chronos is *experimental*. Future releases will not make significant backwards-incompatible changes to documented features of the user interface without good reason. If such changes are made, a compatibility option will be offered, unless there is extremely good reason not to do so. *This applies only to documented features. It applies to neither undocumented features nor the implementation details of those documented.*

Chronos makes some use of undocumented internal PGF/TikZ commands.

Chronos uses `etoolbox` to patch certain internal PGF/TikZ commands. While some of these changes, such as modifications to `rectangle`² are applied only locally, others, including changes to the `tikzpicture` initialisation code³, are made globally.

Chronos has known incompatibilities with certain standard PGF/TikZ libraries (section 15).

Chronos has unknown incompatibilities with other standard and non-standard PGF/TikZ libraries and packages. These will be documented when discovered.

Chronos differs substantially from code previously published as `chronos` on [T_EX StackExchange](#). In particular, the user interface has changed: `chronos` now uses a key-value interface rather than multiple arguments when adding things to the timeline and the timeline itself is now created by the environment `chronos`⁴. See section 15.1 for guidance on converting existing timelines.

Caveat emptor ...

Second, (some of) the assumptions ...

Within the `chronos` environment, `chronos` assumes control over PGF/TikZ layers, disregarding any configuration setup by the user or other packages (section 6.4). This means you cannot use additional, custom layers in `chronos` environments unless you integrate them appropriately with `chronos`'s changes. These changes are made locally and do not affect the use of whatever layers you please in a non-`chronos` environment, such as a regular `tikzpicture`.

Caveat emptor ...

Third, (some of) the limitations ...

The most serious limitation, given `chronos`'s aims (section 1), is that you cannot define styles involving `chronos` keys using the standard PGF/TikZ interface, if you want to use them to configure individual additional elements (sections 6 and 9). Moreover, the alternative mechanism provided has serious shortcomings (section 13.3).

Chronos cannot produce timelines covering hundreds of thousands of years or which need to distinguish temporal units less than a day. It does days, months, years and centuries; it does not do (many) millennia, hours, minutes or seconds.

In particular, `chronos` is not designed to deal with dates outside the current Julian period. In theory, this means any date from 24th November, 4714 BCE should be permissible, but in fact, 24th November, 4713 BCE is the first date for which the package's behaviour should be relatively well-defined⁵. Matters are a little different when it comes to dates in the *next* Julian period. The cut off date for these is sometime in 3268 CE, according to Wikipedia, but `pgfcalendar` appears to be unaware of this. This means you may be able to get away with later dates, even though they are officially beyond the scope of this package⁶.

Chronos draws horizontal timelines. It does not support alternative orientations. In particular, vertical timelines are not currently supported.

Caveat emptor ...

²I am grateful to Symbol 1 for providing the code implementing this at [T_EX StackExchange: 385953](#).

³I am grateful to Martin Scharrer for for this at [T_EX StackExchange: 56405](#).

⁴Early versions on T_EX SE actually used an environment, so this difference applies only to some `chronos`-based answers there.

⁵`pgfcalendar` says it uses the Wikipedia method, but appears to return dates 1 year later than some Wikipedia specifies e.g. day 0 gives a date in 4713 exactly a year after Wikipedia's one in 4714. But Wikipedia itself seems inconsistent, sometimes suggesting a date in 4713 and sometimes the previous year. For current purposes, the right answer doesn't matter: what matters is that `pgfcalendar`'s answer is consistent. This means quibbles about the start date are unimportant (unless you're drawing a timeline starting with Winter Solstice 4714 BCE, of course. If you are, you might want to look into the matter.)

⁶That is, it may work, but it isn't a bug if it doesn't.

Finally, the code lacks both the virtues of sophistication and simplicity, while the user interface is characterised by confusion and complexity, the documentation is spotted with lacunae and unclarities, and the index is a conglomeration of misdirection and bull shit⁷.

Caveat emptor ...

3 Typesetting a Timeline

Further details concerning loading and invocation are explained in sections 4 and 5. The overall structure `chronos` provides is outlined in sections 6 and 6.4. Section 7 covers simple customisation using colour schemes and `chronos` styles. Detailed configuration of the timeline is explained in section 8. Section 9 covers the addition of elements such as lives, events, periods, theories, info boxes and titles to timelines. In this section, we begin by looking at a simple example.

After loading `chronos` in the document preamble:

```
% in document's preamble
\usepackage{chronos}
```

the `chronos` environment is available for typesetting timelines.

```
\begin{chronos}
[]
\end{chronos}
```

This takes an optional argument used to configure the timeline. This determines the size, appearance and duration of the timeline, as well as the use of headings, subheadings and frame. The body of the environment should consist of material to be added to the timeline itself, typically using `chronos`'s commands for adding lives, events, periods, theories, theory circles, info boxes and/or main titles. It is also possible to include arbitrary `TikZ` code in the body of the environment, but commands need to be added to the appropriate `chronos` layer if they are to have their intended effects.

Suppose that we wish to typeset a timeline illustrating developments in the history of writing and printing. Having done exhaustive research utilising a single Wikipedia page, we decide our timeline should begin around 3,100BCE and end in the present. We're going to use the `chronos` style `cronoleg`, which puts year markers on the timeline itself. We decide we'd like large markers every 500 years and a smaller marker halfway between each pair of larger ones. We might, therefore, try

```
\begin{chronos}
[
  cronoleg,% load chronos style
  timeline={% configure the timeline 'line' itself
    start date={-3100},
    end date=2100,
    step minor year=250,
    step major year=500,
  },
  levels=10:10,
]
```

This will result in 'major' markers (marks and years) at 3,000BCE, 2,500 BCE etc. and 'minor' at 2,750BCE, 2,250BCE and so on. Note that `chronos` starts the timeline at 3,100BCE, but assumes we'd like the first marker at 3,000BCE. `levels=10:10` will create a series of invisible nodes above and below the timeline named `level 1, ..., level 10` and `level -1, ..., level -10` respectively. The nodes are constructed so they take the same space as a 'standard' text tag of 'tag' type life created with `\chronoslife`. We can refer to these nodes when placing items to facilitate stacking, spacing and packing.

⁷In what sense 'bull shit'? Take your pick from any of several technical philosophical senses.

Based on our exhaustive seconds-long research, we now want to add some items of interest onto our timeline. We decide we'd like to note the lives of significant figures in the development of contemporary typography, most notably Donald Knuth, as well as a few luminaries from the modern era⁸. We'd also like to note certain specific events, such as key publication dates, and processes of longer duration.

```
\chronosevent{%  
  name=\emph{jikji},  
  date=1377  
}
```

This will create an event in the default style in the default location, just off the timeline. Note that the text displayed in the event's node is '*Jikji*'. The coordinate `jikji` is placed at the point the element is added on the border of the timeline. The circular connector created at this point is the node `chronos connector jikji`. The circular connector on the event's text tag is the node `main connector jikji`. The text tag itself is the node `tag jikji`. As it stands, we may not be able to actually see all these elements if the event's text tag is placed right on the border of the timeline. If `text tag yshift` is non-zero, `chronos` will shift the node but, in general, it is necessary to tell `chronos` where to place the text tag. This doesn't affect the placement of the event on the timeline itself.

```
\chronosevent{%  
  name=\emph{jikji},  
  date=1377,  
  yshift=20pt,  
}
```

This will place the text tag node due north of the circular connector on the timeline with a straight line connecting the circular connector nodes `main connector jikji` and `chronos connector jikji`. However, we might also want to shift the text tag node horizontally and have the connection drawn to the west or east of the text tag.

```
\chronosevent{%  
  name=\emph{jikji},  
  date=1377,  
  yshift=20pt,  
  xshift=-5pt,  
  anchor=east,  
}
```

will shift the text tag 5pt to the left and draw the connection up and left from the timeline to `main connector jikji` which is now drawn `east` rather than the default `south`.

We decide to place a second event, for which we have a precise date. This time, we use `as is` to tell `chronos` not to attempt to capitalise the text. This is necessary because we have an `\emph{<word> <word>}` and `chronos`'s capitalisation command can't cope with this. This also means we need to add appropriate capitalisation ourselves.

```
\chronosevent{%  
  date={868-05-11},  
  name={Publication of \emph{Diamond Sutra}},  
  yshift=-40pt,  
  xshift=20pt,  
  anchor=west,  
  as is,  
  connectors={east,south},  
}
```

Note that this event is placed below the timeline.

We decide to add some notable figures next. For this, we create elements of tag type `life`, beginning with the inventor of movable type, Bi Sheng.

⁸In my discipline, 'modern' means roughly the sixteenth to nineteenth centuries.

```
\chronoslife{%
  name=bi sheng,
  birth=972,
  death=1051,
  at=tag jikji.north -| bi sheng,
  connectors={east,north},
}
```

Note the use of `at` to place the text tag detailing the name and dates. Since this node is placed above the timeline, its anchor is `south` by default. `at=tag jikji.north -| bi sheng` aligns this anchor directly above the relevant point on the timeline (`bi sheng`) and just on top of `tag jikji`. If you want to fit many items onto your timeline, fitting them closely together is useful but you could, of course, lift the box higher if you want a bit more space.

Leaping ahead, we now want to add Donald Knuth.

```
\chronoslife{%
  name=donald knuth,
  birth={1938-01-10},
  text tag yshift=40pt,
  connectors={west,north},
}
```

Note the omission of `death` for a living person. Chronos assigns today's date internally for placement purposes, but will not typeset it when constructing the text tag⁹. This works reasonably, but the connection from the timeline crosses the text node for the publication of the *Diamond Sutra* because `chronos` has placed this item below the timeline, even though there is plenty of space above. This is because, by default, `chronos` alternates between placement above and below the line. In this case, we decide to override the default choice.

```
\chronoslife{%
  name=donald knuth,
  birth={1938-01-10},
  text tag yshift=40pt,
  connectors={west,north},
  place above,
}
```

Note that the `cronoleg` rotates the colours used for elements belonging to tag types `life`, `event` and `period`, but not `theory`, but colour lists are rather subdued for `events` and `periods`. For each type of elements, one set of colours is used below and another above the timeline. These colours can be accessed later as `colour <name>`¹⁰.

Colour rotation can be switched on or off for particular kinds of elements, overridden for individual elements and configured by altering the colour lists `chronos` cycles through. These colours are tracked by copying them to new names for each element created and may be accessed using these names later. This means you can draw something in the colour assigned to Donald Knuth, say, without knowing which colour that is. If you add an element to the timeline or change the colour lists later, the drawing will use the appropriate colour. For example,

```
\node (pi) [colour donald knuth, font=\Huge, right=5pt of tag donald knuth.base east, anchor=base west] {${\pi}$};
```

will add a large π in the colour (automatically or otherwise) assigned to Knuth.

```
\draw [colour donald knuth] (tag donald knuth.north) ++(0pt,20pt) circle (10pt);
```

would draw a circle above Donald Knuth's text tag in the colour automatically assigned to Donald Knuth.

⁹`chronos` is not the most optimistic of packages.

¹⁰In most cases, you can also access items using American spelling. So `color` would work here. So would `lliw`.

We next decide to indicate the period when woodblock printing was used to produce books. This is a *circa* date, so we can't use `chronos`'s automatic production of the date information, though we still need to specify dates for placement on the timeline. We'd still like `chronos` to format the name of the text tag, though, so we use `dates content` to override the automatic production of date labels.

```
\chronosperiod{%
  name=woodblock printing,
  start=600,
  end=1450,
  yshift=-20pt,
  xshift=10pt,
  anchor=west,
  dates content={c600--1450\ceyearlabel},
  place below,
}
```

If we wanted to override the formatting of the name rather than the dates, we could use

```
name=woodblock printing,
name content={WoOdBlOcK pRiNtInG},
```

If we wanted something completely different in place of the name and date information, we could instead use

```
text content={something entirely different\--- not even about woodblocks!},
```

BCE dates require special consideration. In general, a minus indicates BCE, but `chronos` needs to be able to distinguish this from the hyphen between years and months or months and days in standard date specifications (section 8.2). This means either providing a full date of the form `-YYYY-MM-DD`, for example, or ensuring `chronos` expects only a partial date such as a year.

```
\chronosperiod{%
  name=proto-Elamite use of cylinder seals,
  start={{-3100}-01-01},
  end={{-2700}-12-31},
  dates content={c3000\,\bceyearlabel},
  yshift=20pt,
  connectors=north,
  connectors=east,
}
```

Here, we protect the BCE year with curly brackets, specify a default month and day. If we specified only a year, `chronos` would assign a month and day; if we assigned only a year and month, `chronos` would assign a day. (The outer set of curly brackets is standard and cannot be omitted for full date specifications, regardless of era.)

We've now added examples of each of the three basic types `chronos` supports connecting to our timeline. However, the package also offers some complementary elements. These are not connected to the timeline, though theories are designed to be connected to the types which are.

```
\chronostheory {%
  name=TeX,
  text content=\TeX,
  at=donald knuth-text.north west,
  xshift=-10pt,
  anchor=south east,
  connectors={east},
}
```

We also want to indicate Knuth's connection with $\text{T}_{\text{E}}\text{X}$, so we join the connector we made when creating the text tag for Knuth to the connector we've just created for $\text{T}_{\text{E}}\text{X}$. `Chronos` supports the addition of such connectors on most text tags created with its commands and the drawing of connections between connectors.

```
\draw [chronos connect=life:donald knuth] (connector donald knuth) -- ++(-5pt,0pt) |- (connector TeX);
```

This makes it possible to connect multiple people to the same theory, for example, as well as connecting a single person to multiple theories. In a more complete chronology, several different font designers or book publishers, for example, might be connected with a particular approach to typography. Elements which support connectors out-of-the-box are those belonging to tags of types life, event, period and theory.

When `cronoleg` is used, connectors are small circular nodes on the timeline's border and the borders of text tags i.e. the nodes containing information about the `chronos` elements presented in the chronology illustrated.

In contrast, theory circles, info (information boxes), copyleft or copyright notices and main titles are freestanding objects without ready-made connectors.

Headings and subheadings are designed to label stretches of time and are placed in relation to the timeline, though no connecting lines are drawn.

When we've finished adding material to the timeline, of course, we need to complete it.

```
\end{chronos}
```

4 Loading the Package

Chronos requires a $\text{\LaTeX} 2_{\epsilon}$ format no older than 2021–11–15. To load the package simply add the following to your document’s preamble.

```
\usepackage{chronos}
```

Chronos will load the following packages and libraries automatically:

Packages:

- calc
- chronos-lib-colschemes (part of chronos)
- chronos-lib-styles (part of chronos)
- etoolbox
- expl3 (if required)
- fp
- pgfcalendar
- svn-prov
- tikz
- xcolor
- xparse (for $\text{\LaTeX} 2_{\epsilon}$ formats prior to 2020–10–01)

PGF/TikZ libraries:

- arrows.meta
- backgrounds
- calc
- decorations.text
- fit
- fixedpointarithmetic
- positioning
- shadows

```
simple colour names = true|false
no simple colour names
simple color names
no simple color names
boolean key
```

The only two options currently supported are `simple colour names` or `simple color names` and its complement `no simple colour names` or `no simple color names`. The following are equivalent:

```
\usepackage{chronos}
\usepackage[simple colour names]{chronos}
\usepackage[simple colour names=true]{chronos}
\usepackage[no simple colour names=false]{chronos}
```

In these cases, `chronos` will create an additional colour for each additional element of `tag`-type `life`, `event`, `period`, `theory` or `info` named $\langle name \rangle$, where $\langle name \rangle$ is the value given to `name` when creating the element.

Since `chronos` creates these colours globally, this is potentially problematic. To disable it use any of the following

```
\usepackage[no simple colour names]{chronos}  
\usepackage[no simple colour names=true]{chronos}  
\usepackage[simple colour names=false]{chronos}
```

If you want to disable such names later, perhaps for specific timelines, see section 8.8.

5 Invocation

chronos [*⟨chronos preamble⟩*]
environment

The *⟨chronos preamble⟩* is a *⟨key-value list⟩* setting any non-default options which should be applied to the timeline and any other macro-level elements of the picture to be constructed. At a minimum, most users will want to specify start and end dates, but the majority will likely want to customise the timeline further. (If you do not much care about customisation, there are simpler packages to typeset timelines!)

Some options can be given only *in or before* the *⟨timeline specification⟩* in the optional *⟨chronos preamble⟩*. Others will have no effect or unwanted effects at this point and must be specified later.

The environment chronos is a wrapper for a tikzpicture. It can neither include, nor be included in, another tikzpicture. Additional drawing commands must, therefore, be included in chronos itself.

6 Chronos Anatomy

Figure 2 provides an overview of the configuration and anatomy of a `chronos` timeline.

As explained in section 5, the `timeline` itself is constructed by the `chronos` environment, as determined by the `<chronos preamble>`, any prior use of `\chronosset` and fallback defaults.

In addition to configuring the `timeline` itself, the `<chronos preamble>` and any prior use of `\chronosset` determine the use and configuration of any `frame`, `headings` and `subheadings`, as well as the default configuration of any additional elements.

The body of the `chronos` environment is the `<timeline additions specification>`. The `<timeline additions specification>` specifies what should be added to the `tikzpicture` besides the `timeline` itself and any `frame`, `headings` or `subheadings`. It will typically consist of a series of `chronos` commands specifying the items to be connected to the `timeline` and any non-connected elements (section 9). However, it may include any code valid in a `tikzpicture` environment or be entirely empty.

Section 6.1 provides a breakdown of the various elements of which the `timeline` is composed. Section 6.2 provides an overview of the additional elements which may be added in the `<timeline additions specification>`.

If your `timeline` uses non-`chronos` commands, you will need to read sections 6.4 and 10, which explains the layers `chronos` uses. If your commands are not having their usual effects, you should first check whether they are simply hidden by another layer.

6.1 Chronos Timeline

The `timeline` itself is a horizontal line consisting of some or all of the following elements

- `Timeline line` refers to the main line, which is drawn or filled by default depending on height and configuration. The `height`, `width` and `timeline border height` are responsible for the total size of the `timeline`.
- `Borders` are (potentially) filled with a gradient above and below the main line. By default, borders are added when marks are placed on the `timeline` itself, which necessitates a taller `timeline`.
- `Era labels` are (potentially) placed at each end of the line, depending on the time period covered.
- `Timeline years`, `minor years`, `marks`, `minor marks` and `bare marks` may be placed above, below or on the main `timeline` line.

Some elements must be specified in the `<chronos preamble>`, but are constructed only at the end of the `chronos` environment. These include optional `headings` and `subheadings` to be placed at the top of the `chronos` environment and an optional `frame`.

`Headings` and `subheadings` are constructed after and above most other elements on `chronos foreground layer`. As explained in section 8.7, `headings` and `subheadings` may be used to roughly indicate named stretches of time such as ‘Tudors’ or ‘Bronze Age’.

- `Headings` are placed in a single row at the top.
- `Subheadings` are placed just below the `headings` in two rows:
 - The upper `subheadings` are placed in a single row just beneath the `headings`.
 - The lower `subheadings` are placed in a single row just beneath the upper `subheadings`.

The `frame` is constructed even later, but drawn behind most other elements on `chronos background layer`.

6.2 Chronos Additional Element Types

Aside from the `timeline` itself, its `headings` and `subheadings` and `frame`, `chronos` provides six primary types of element which may be added to the `timeline`: `life`, `event`, `period`, `theory`, `info` and `theory circle`. In this documentation, these are referred to as ‘`tags`’ or ‘`tag types`’. Three further `tags` encompass one-off elements:

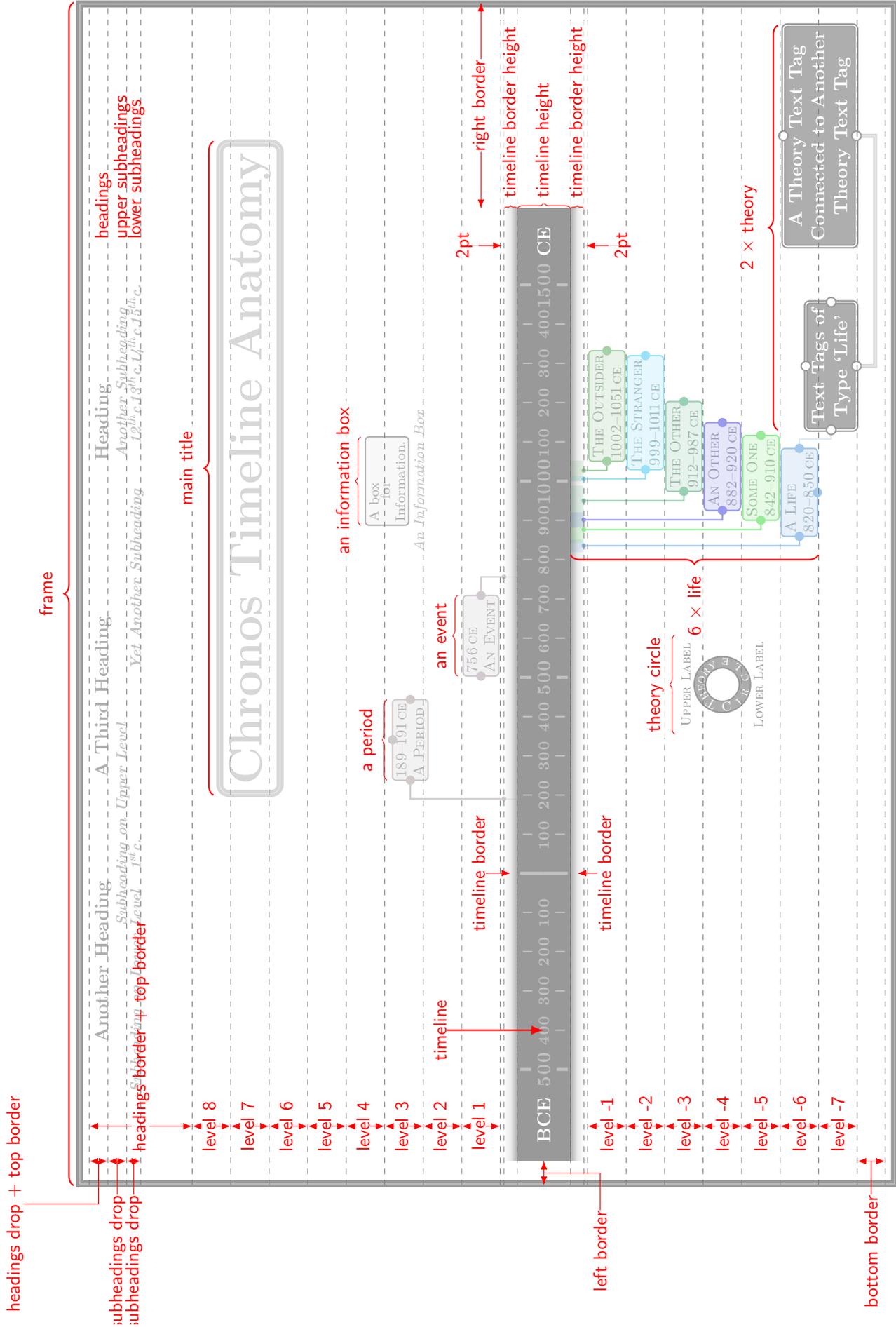


Figure 2: chronos anatomy

main covers the main title and frame, while `copyleft` and `copyright` account for any declaration of `copyleft` or `copyright`.

For example, all elements created using `\chronoslife` are said to belong to tag type `life`.

6.2.1 Primary Types

6.2.1.1 Timeline-Connectable Elements Elements belonging to the first three tags (`life`, `event`, `period`) are (potentially) connected to the `timeline` and are placed according to date of occurrence.

- These elements are assigned colours and colour names are created so they may easily be reused. These colours may (and, by default, are) used to create `connections`, `connectors`, `lines` and `text tags`.
- These elements are connected to the `timeline` by default using `connections` which join `chronos connectors` to `text tag connectors` on the elements' `text tags`.
- Dates/periods are (potentially) drawn or filled on, above or below the `timeline` using `lines`.
- `Text tags` are created for the elements¹¹. By default, these typically include a name and date or date-range, though arbitrary content is permissible. The location of `text tags` is configurable, though it usually makes sense to place them in relation to their `chronos connectors`.
- `Life` and `period` use two dates for placement. A line is (potentially) drawn and/or filled on, above or below the `timeline`, by default in the element's associated colour.
- `Event` uses a single date for placement. A line is (potentially) drawn on the `timeline`, by default in the element's associated colour.

Timeline-connectable elements are also connectable (note 6.2.1.2).

6.2.1.2 Connectable Elements Elements belonging to the first four tags (`life`, `event`, `period`, `theory`) are (potentially) connectable to each other.

- These elements (potentially) feature `connectors` which may be used to connect elements together. When the first three are connected to the `timeline`, one such connector is created by default¹².
- Elements belonging to the `theory` tag are connectable, but not timeline-connectable. Unlike timeline-connectable elements (note 6.2.1.1), they cannot be connected to the `timeline` and may be freely placed; unlike non-connectable elements (note 6.2.1.3), they may be connected to each other and/or timeline-connectable elements.

6.2.1.3 Non-Connectable Elements

Elements belonging to the remaining tags (`info`, `theory circle`, `main`, `copyleft` and `copyright`) are non-connectable and, with the exception of `frame` may be located according to user preference.

- Like connectable-but-not-timeline-connectable elements, non-connectable elements are not connected to the `timeline` and may involve no date information at all, but unlike theories they do not feature `connectors` so may not easily be connected to other elements.
- `Info` and `theory circle` elements are standalone items for providing content. The former (potentially) have `captions` below; the latter (potentially) have `labels` above and/or below. The first are basically just text nodes with arbitrary content; the second can display two small chunks of text arranged in semicircles with a hole in the middle for a letter or symbol.
- `Theory circles` are *slow* and their use should be limited to avoid excessive compilation times. They are also arguably the most difficult to read and should be used only for items of minor or secondary importance.

¹¹I am grateful to Symbol 1 for enabling `connectors` to be centred correctly on the borders of `text tags` at [TeX StackExchange: 385953](https://tex.stackexchange.com/questions/385953).

¹²Connectors may be customised to 'disappear', but even invisible connectors can be used in connections.

- The standalone elements are best created last and are most useful for filling in ‘holes’ in a timeline which would otherwise look unbalanced. If chiropody didn’t develop much in the twelfth century or not much is known about the finer points of tortoise-raising in the second, these elements may be used to plug the unsightly gaps left by inconvenient histories.

6.2.2 Secondary (Sub-)Elements

Orthogonal to the primary elements explained above, `chronos` uses the following (sub-)elements:

- **Connectors** are small elements drawn on the boundaries of `text tags` and the `timeline` which can be used as connection points. By default, they are small and circular, but they may be rendered invisibly or otherwise according to preference.
- **Connections** are drawn between **connectors**. The package draws a connection between the `timeline` and date-placed elements by default, but occasionally you may prefer to specify this connection manually. Other connections can be added to link elements.
- `Text tags` hold information associated with all elements except **theory circles**.
- Lines are marked on the `timeline` to indicate the date and/or duration of dated elements.

6.3 Chronos Coordinate and Node Names

Figure 3 shows key coordinate and node names. Those available by default can be shown on any `timeline` using the option `debug`. Examples of different `tags` have been added with labels to illustrate how `chronos` names their coordinates and nodes. Detailed documentation is provided in sections 8 and 9.

6.4 Chronos Layers

In addition to loading the `backgrounds` library, which defines the layer `background`, and the default layer `main`, `chronos` defines another four layers, for a total of six: `chronos background` and `chronos middle ground`, which are layered between `background` and `main`, and `chronos foreground` and `chronos overlay`, which are layered above `main`. From top to bottom:

```
chronos overlay
chronos foreground
main
chronos middle ground
chronos background
background
```

Section 10 explains how to draw directly on different layers. You may wish to do this if you are using non-`chronos` code in the (*timeline additions specification*) or the facilities explained in section 12 for deferring code.

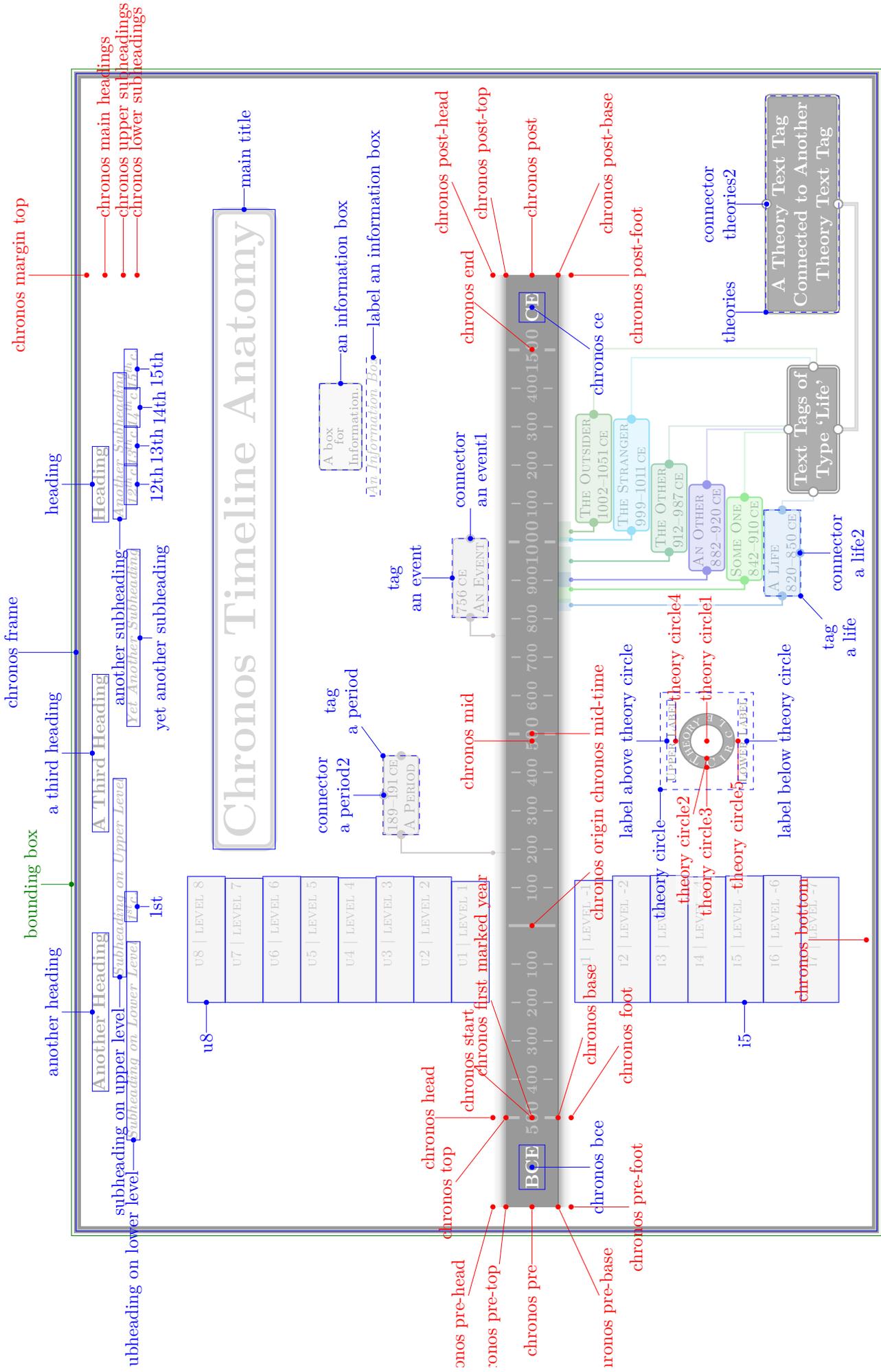


Figure 3: chronos anatomy: key coordinate and node names

7 Chronos Schemes and Styles

Two simple methods for applying, defining and reusing chronos styles are provided: chronos styles and colour schemes. If using both, load the chronos style first, since it may already load a colour schemes.

7.1 Chronos Styles

By far the easiest way to customise a timeline is simply to load a chronos style in the `<chronos preamble>`. This section illustrates a basic timeline typeset with each of chronos’s standard styles.

Note that you will typically need to set `start date` and `end date` and perhaps adjust how often years and marks appear on your timeline. Chronos styles such as `key[chronosstyle]event splitter` set highly idiosyncratic dates by default, simply by way of example. chronos will not warn you if you don’t override options set by a chronos style.

In selecting a chronos style, bear in mind that some things are easy to change, while others are harder. At a minimum, you should pick an ‘on line’ chronos style if you want `timeline years on line` and an ‘off line’ one if you want them above or below. `event years on line` requires an ‘on line’ chronos style; `event dates split` is designed for an ‘off line’ one.

You should also think about how much information you need to display. `date centric` won’t work for a densely packed timeline, so if you have a lot of things to pack in, don’t choose this unless you’re drawing an extremely long timeline. Likewise, `cronoleg` will look rather silly if you only want to represent the lives of Socrates and Plato.

7.1.1 ‘On Line’ Styles

All ‘on line’ styles are designed to support adding elements both above and below the timeline. This includes the default settings. See table 1 and fig. 4.

`cronoleg`
chronos style The most developed and best tested, if somewhat idiosyncratic, chronos style, based on the code used to construct my Western Philosophy Timeline. It constructs a 235mm timeline and uses a colour scheme highlighting elements of type life, but the colours may be adjusted or the same colour scheme applied to event and period as well. By default, it is designed to produce a picture occupying an entire A4 page and has a wide right-hand margin for additional elements, in addition to ten levels above and below the timeline. See table 1 and fig. 5. By default, this chronos style does *not* use the bounding box for the frame.

`date centric`
chronos style A chronos style with a monochrome appearance and sans-serif fonts of 150mm¹³. Intended for timelines highlighting relatively few dates. See table 1 and fig. 6. This style demonstrates the use of `event years on line` and `special date`.

`lavender menace`
chronos style A variant of `modern` with a muted colour scheme and sans-serif fonts. By default, it produces a timeline covering the modern era (1500–1900 CE). See table 1 and fig. 7a.
`modern`
chronos style A chronos style with a monochrome appearance and sans-serif fonts. By default, it produces a timeline covering the modern era (1500–1900 CE). See table 1 and fig. 7b.

`rainbow serif`
chronos style A colourful variant of `serif on line` utilising xcolor colour series and serif fonts. See table 1 and fig. 8a.

`serif on line`
chronos style A chronos style with a monochrome appearance and serif fonts. See table 1 and fig. 8b.

`sober judge`
chronos style A somewhat subdued chronos style with a monochrome appearance, sans-serif fonts and boxed text tags. See table 1 and fig. 9.

¹³Based on my answer at [TeX StackExchange: 324448](https://tex.stackexchange.com/questions/324448).

Table 1: Summary of chronos styles.

Name	Timeline Year Style	Defaults				
		Levels	Dates	Colour Scheme	Rotation	Arrow
-	on line	0:0	1800–2050 CE	default	✓	–
cronoleg	on line	10:10	500 BCE– 2050 CE	cronoleg	✓	–
date centric	[on line]	–	1935–2010 CE	default	–	–
lavender menace	on line	3:3	1500–1900 CE	lavender+chronosSilver	✓	–
modern	on line	3:3	1500–1900 CE	modern	–	–
rainbow serif	on line	3:3	1500–2100 CE	xcolseries	✓	–
serif on line	on line	3:3	1800–1900 CE	default	–	–
sober judge	on line	3:3	1/10/1001– 14/6/1003 CE	default	–	–
blues below	off line, below	0:3	1550–2050 CE	blues	✓	✓
flipping blues	off line, above	3:0	1550–2050 CE	blues	✓	✓
contemporary 90	off line, above	0:3	2002-2016 CE	contninety	–	✓
off line colour	off line, below	–	3000– 2000 BCE	offlinebasic	✓	✓
off line colour alt	off line, below	–	3000– 2000 BCE	offlinealt	✓	✓
off line simple	off line, below	–	3000– 2000 BCE	offlinebasic	–	✓
rotated 45	off line, above	–	25 BCE–20 CE	default	–	–
simple arrow	off line, above	–	1–2000 CE	default	–	✓
somewhat plain	off line, above	0:3	500 BCE– 2050 CE	default	–	–
event splitter	[above]	–	01/13– 02/22/2014 CE	default	–	–
lines on line	none	–	1–2016 CE	default	✓	✓
plain arrow	none	–	1–2016 CE	default	✓	✓

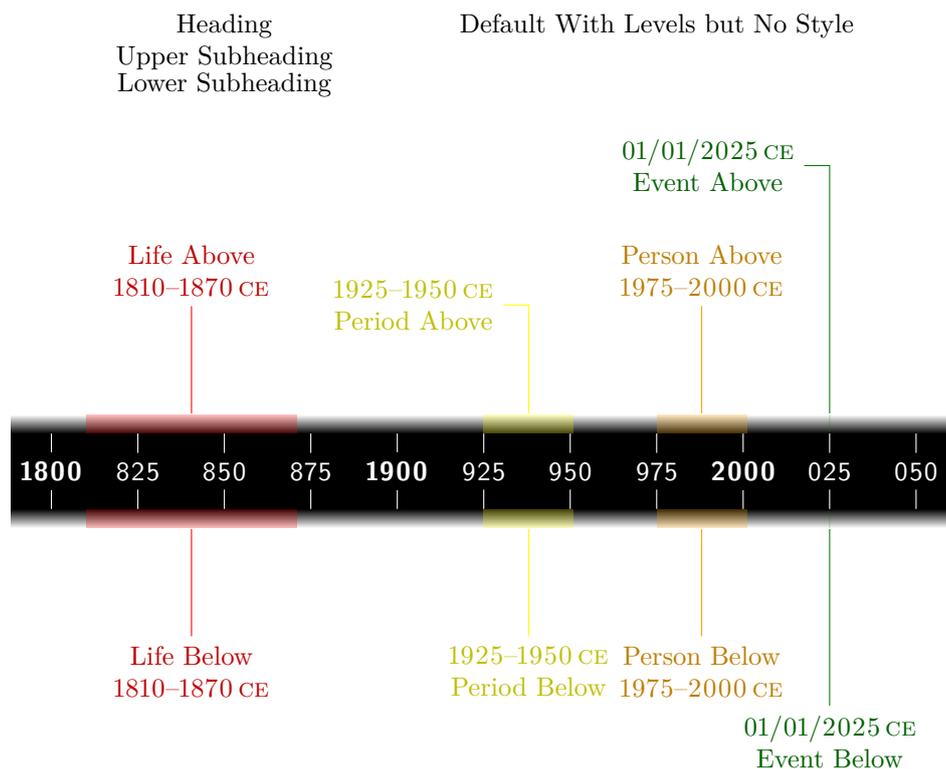


Figure 4: Chronos style: none.

7.1.2 ‘Off Line’ Styles

blues below A chronos style featuring the **blues** colour scheme, off-set lines and year labels rotated through 45° .
chronos style Intended for timelines which add elements below. See table 1 and fig. 10a. This style demonstrates how to rotate year labels.

contemporary 90 A chronos style with a monochrome appearance, sans-serif fonts and rotated year labels, which produces a relatively short timeline of 90mm by default. Intended for timelines which add elements below. See table 1 and fig. 11.

flipping blues A variation of **blues below** featuring year labels rotated through -45° . Intended for timelines which add elements above. See table 1 and fig. 10b. This style demonstrates how to utilise an existing chronos style to produce a variant.

off line colour = $\langle length \rangle$
chronos style

A straightforward style utilising scientific dates in which the line tapers to form an arrow. Intended for timelines which add elements above and/or below. The optional $\langle length \rangle$ specifies the length of the tapering.

Default: 20mm

See table 1 and fig. 12a. This style demonstrates the use of **chronos middle ground layer** to reduce visual clutter where **connections** cross **timeline marks**. Although the **connections** are drawn after the **timeline**, they are placed on a lower layer, with a partially transparent rectangle in between.

off line colour alt = $\langle length \rangle$
chronos style

A variant of **off line colour** which uses a different colour scheme.

Default: 20mm

Heading
Upper Subheading
Lower Subheading

Cronoleg

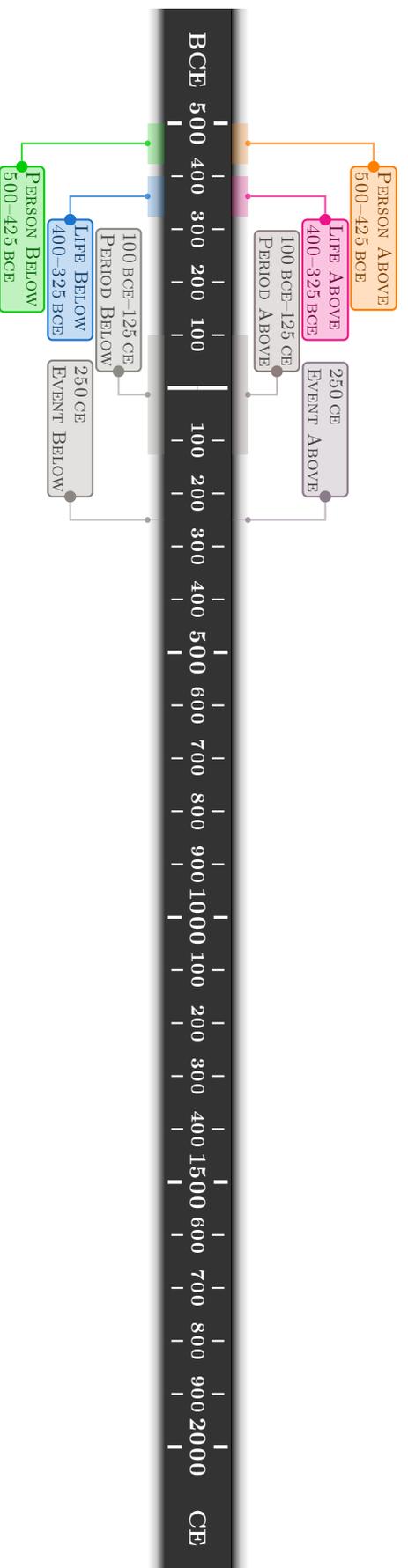


Figure 5: Chronos style: cronoleg.

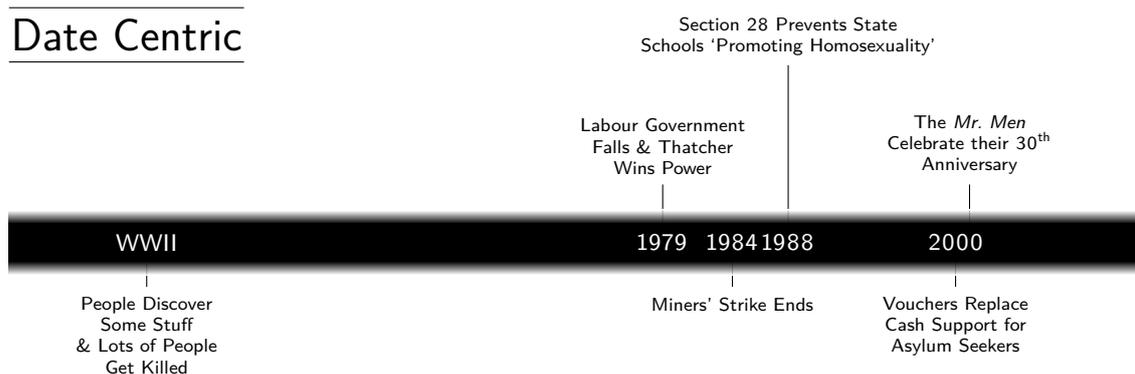


Figure 6: Chronos style: date centric.

See table 1 and fig. 12b.

off line simple = $\langle length \rangle$
chronos style

A less colourful variant of `off line colour` utilising only two colours¹⁴.

Default: 20mm

See table 1 and fig. 12c.

rotated 45 A chronos style featuring the off-set lines and text tags rotated through 45°. Intended for timelines which add elements below. See table 1 and fig. 13. This style demonstrates how to rotate text tags.
chronos style

simple arrow = $\langle length \rangle$
chronos style

A monochrome appearance with a plain 200mm arrow timeline and years and marks above¹⁵. $\langle length \rangle$ determines the length of the taper comprising the arrow.

Default: 10mm

Intended for timelines which add elements below. See table 1 and fig. 14.

somewhat plain A chronos style with a monochrome appearance and sans-serif fonts which produces a relatively short timeline of 100mm by default. Intended for timelines which add elements below. See table 1 and fig. 15. This style demonstrates how to create a style to draw lines above and below the main title node, without drawing the left and right sides of the node.
chronos style

7.1.3 ‘No Year’ Styles

event splitter A 150mm timeline with no year labels which demonstrates the use of `event dates split`¹⁶. Intended for timelines with connected elements solely of tag type event. See table 1 and fig. 16.
chronos style

lines on line = $\langle dimension \rangle$
chronos style

A 120mm timeline arrow, $\langle dimension \rangle$ high, with no year labels and life, event and period lines drawn on the timeline itself¹⁷. Date information is confined to text tags. Out-of-the-box, this chronos style adds elements of tag type event above and those of type life and period below.

Default: 5mm

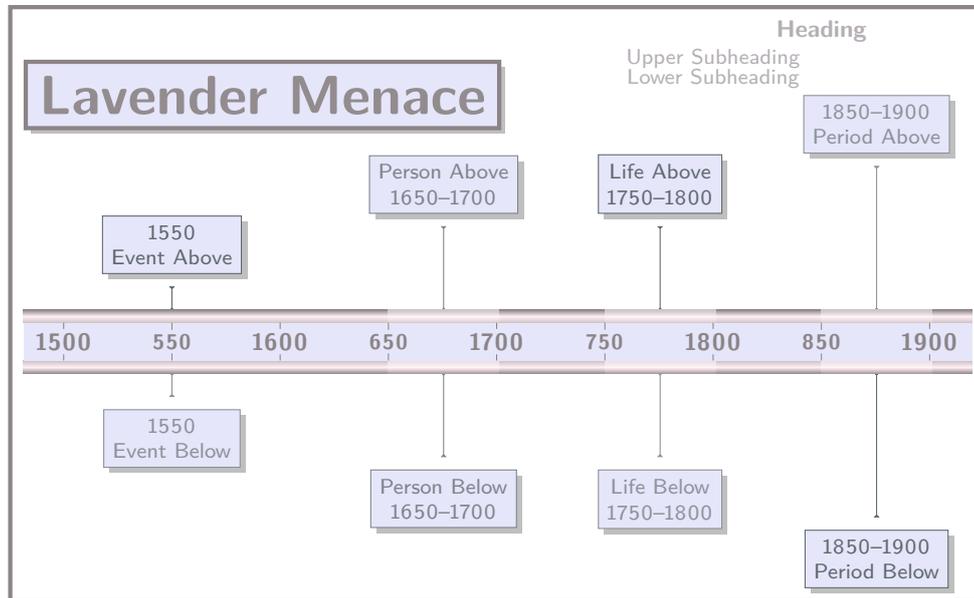
See table 1 and fig. 17.

¹⁴In fact, this version is closest to the original. See my answer at [TeX StackExchange: 324106](#).

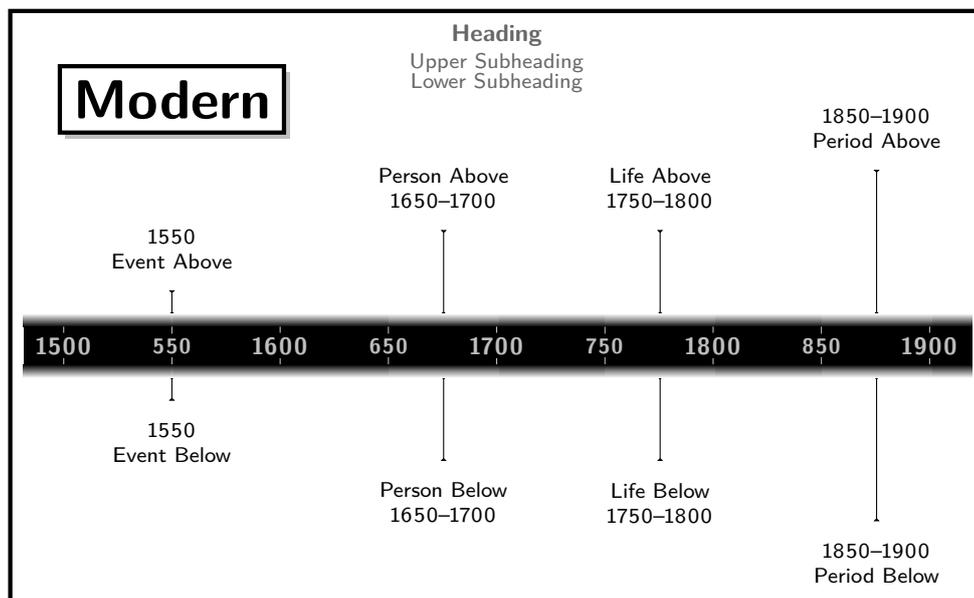
¹⁵Based on my answer at [TeX StackExchange: 342699](#).

¹⁶Based on my answer at [TeX StackExchange: 325890](#).

¹⁷Based on my answer at [TeX StackExchange: 324453](#).

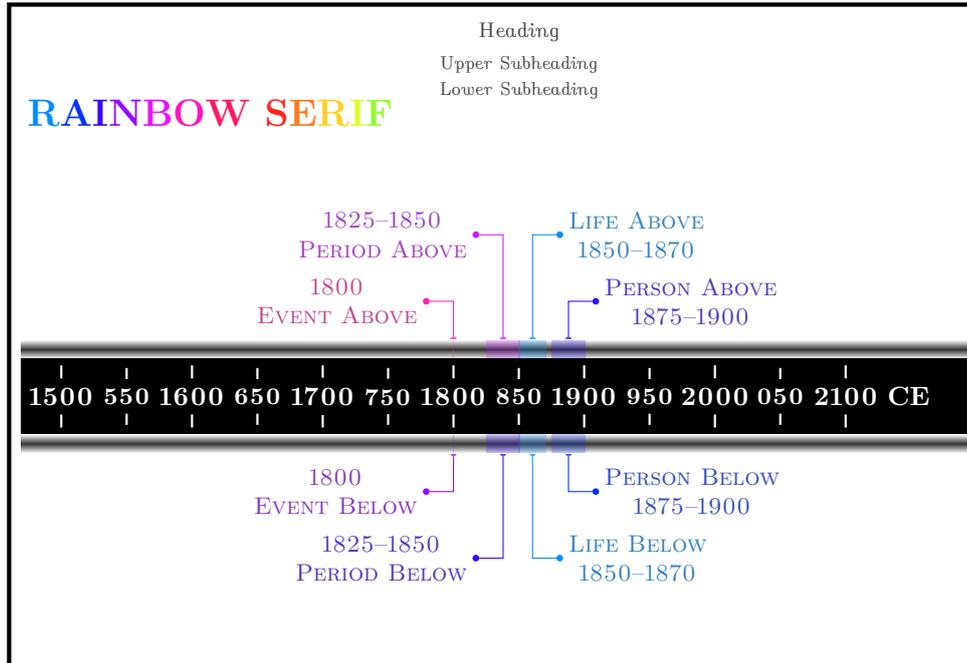


(a) Chronos style: lavender menace

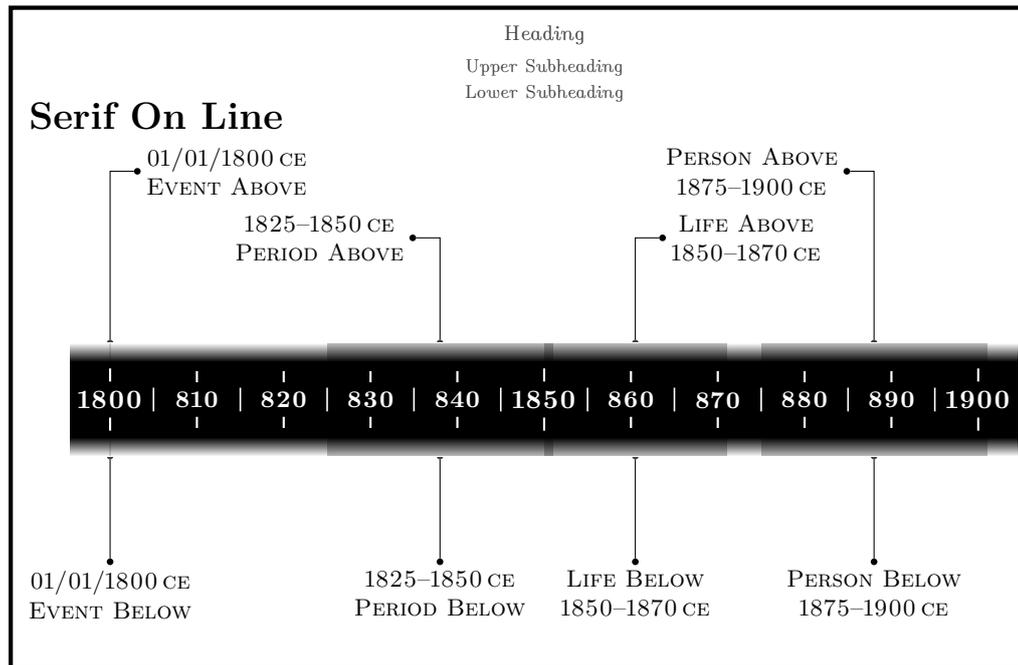


(b) Chronos style: modern

Figure 7: Figure 7a is a variant of fig. 7b.



(a) Chronos style: rainbow serif.



(b) Chronos style: serif on line.

Figure 8: Figure 8a is a variant of fig. 8b.

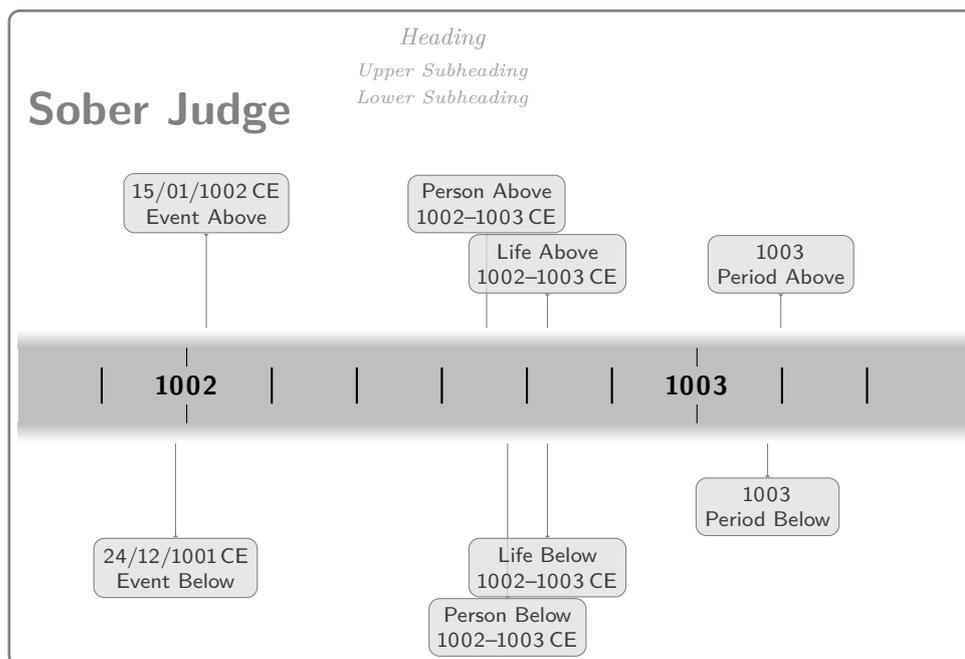


Figure 9: Chronos style: sober judge.

`plain arrow` = \langle *dimension* \rangle
chronos style

A variant of `lines on line` (fig. 17) which draws a 120mm timeline arrow with no year labels and life, event and period lines drawn on the timeline itself¹⁸. Date information is confined to text tags.

Default: 5mm

Intended for timelines which add elements of `tag` type event above and those of type life and period below. See table 1 and fig. 17b.

7.2 Chronos Colour Schemes

As explained in section 8.8, `chronos` utilises a somewhat complex system for colour customisation. In many cases, however, you will not need to delve into the mechanisms used. Instead, you can simply load an existing colour scheme. If none of the provided schemes meet your needs, see section 13.1.

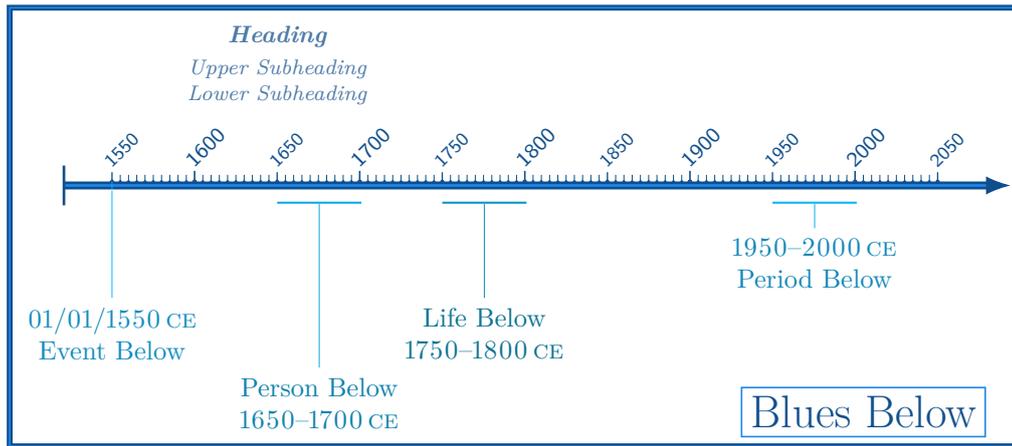
To load a colour schemes, you just write

```
\begin{chronos}
[
  modern,
  colour scheme=blues,
]
\end{chronos}
```

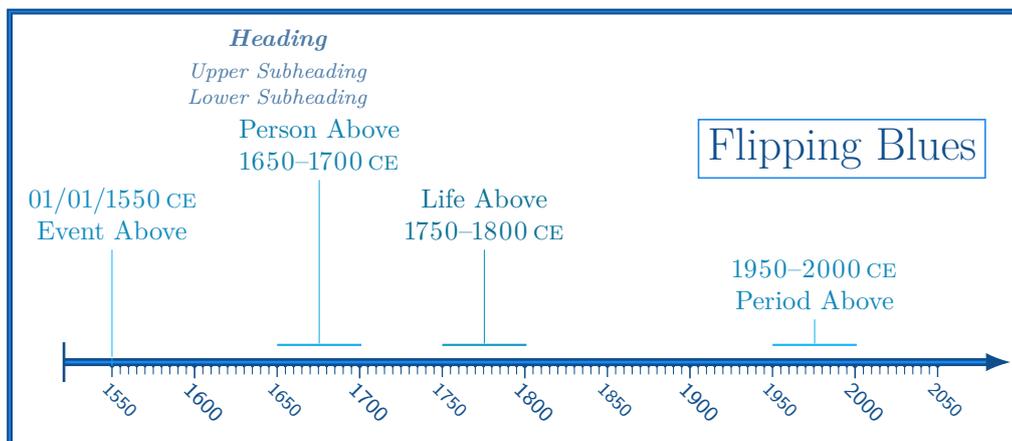
which would load the `chronos` style `modern` followed by the colour schemes `blues`. Since `chronos` styles may legitimately load colour schemes, but colour schemes may not load `chronos` styles, always load any `chronos` style *before* any colour scheme. Then make any further modifications you wish.

```
\begin{chronos}
[
```

¹⁸Based on my answer at [TeX StackExchange: 324453](https://tex.stackexchange.com/questions/324453).



(a) Chronos style: blues below.



(b) Chronos style: flipping blues.

Figure 10: Figure 10b is a variant of fig. 10a.

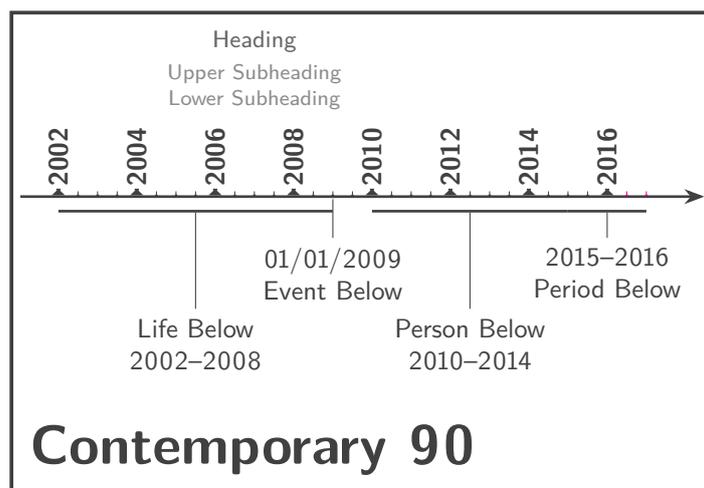
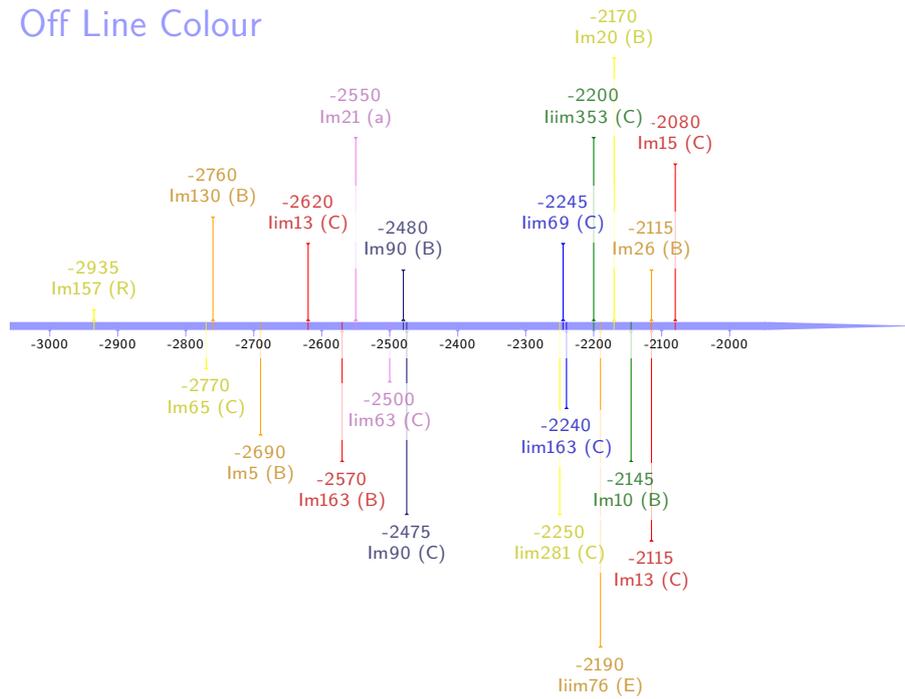


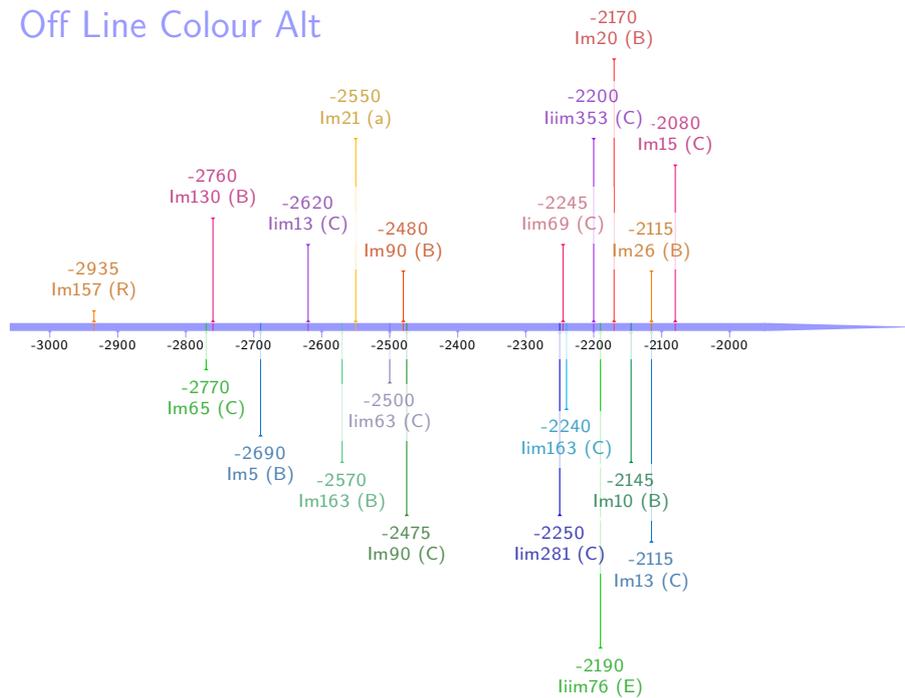
Figure 11: Chronos style: contemporary 90.

Off Line Colour



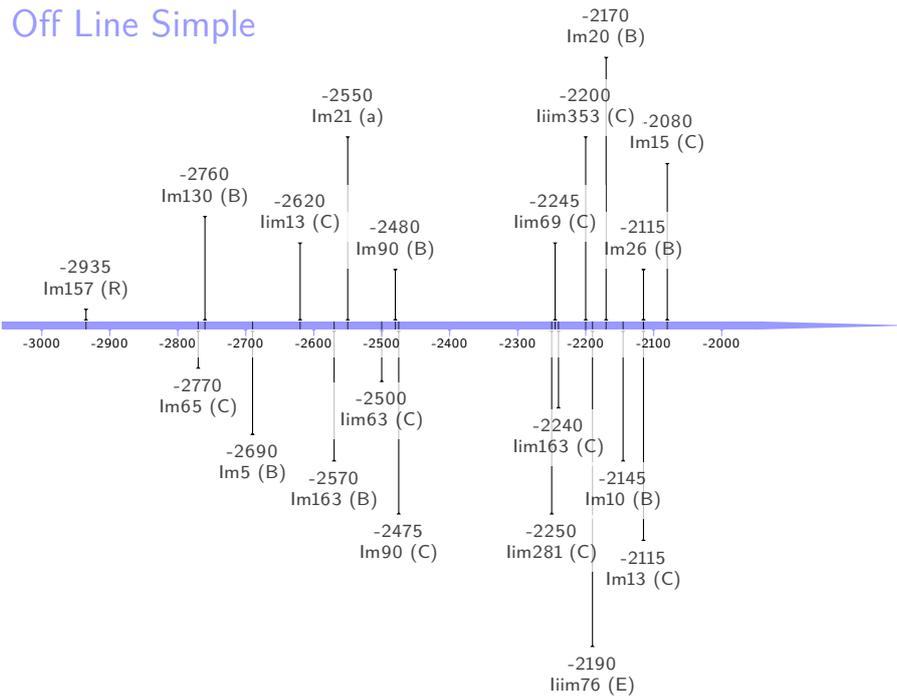
(a) Chronos style: off line colour.

Off Line Colour Alt



(b) Chronos style: off line colour alt.

Figure 12: Figures 12b and 12c are variants of fig. 12a.



(c) Chronos style: off line simple.

Continued Figure 12: Figures 12a and 12c are variants of fig. 12b.

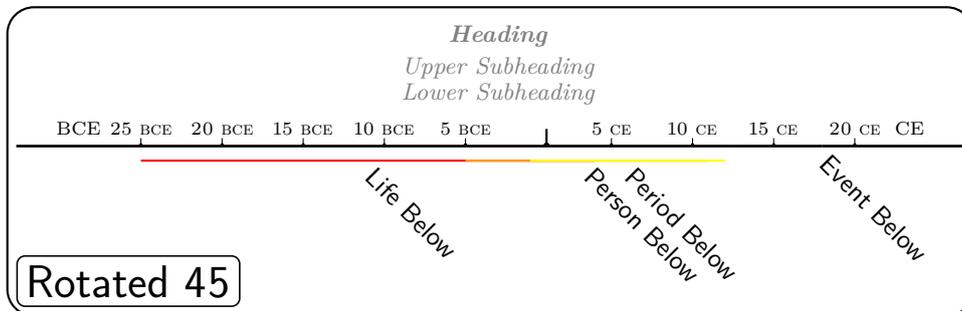


Figure 13: Chronos style: rotated 45.

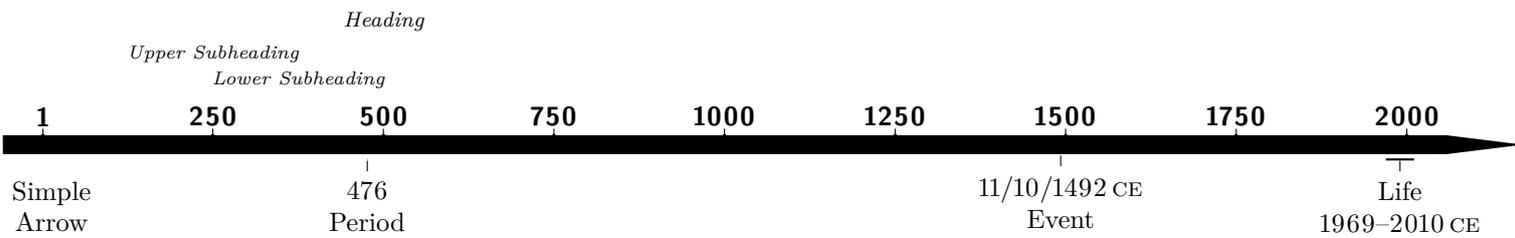


Figure 14: Chronos style: simple arrow.

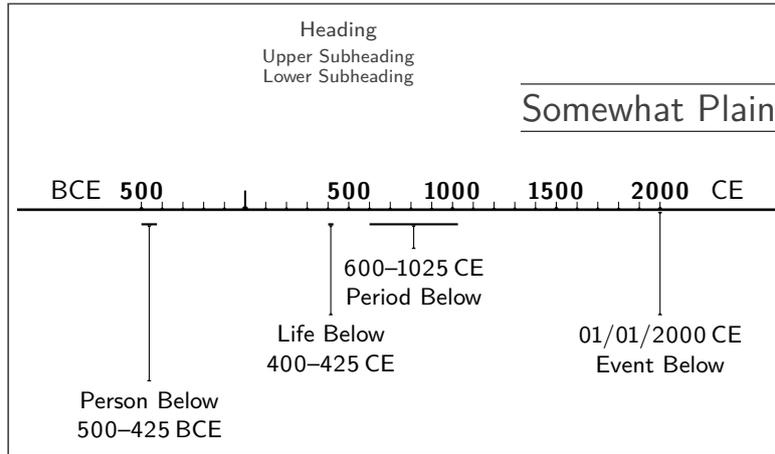
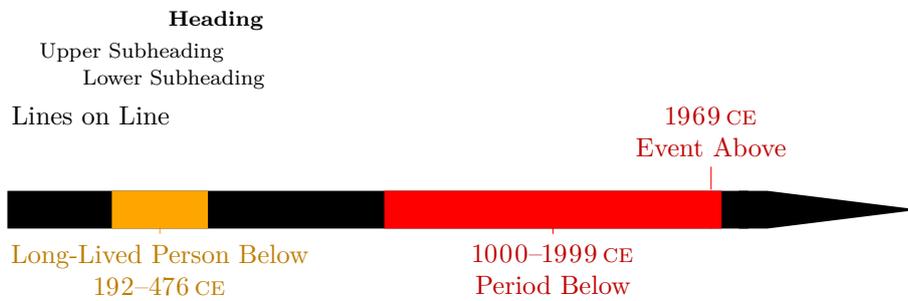


Figure 15: Chronos style: somewhat plain.



Figure 16: Chronos style: event splitter.



(a) Chronos style: lines on line.



(b) Chronos style: plain arrow.

Figure 17: Figure 17b is a variant of fig. 17a.

Table 2: Chronos Colour schemes.

Colour scheme	Variant Of	Default For	Examples
- (default)	-	rotated 45, serif on line and somewhat plain	figs. 4, 8b, 13 and 15
blues	-	blues below and flipping blues	figs. 1, 10a and 10b
contninety	-	contemporary 90	fig. 11
cronoleg	-	cronoleg	fig. 5
lavender	-	lavender menace	fig. 7a
modern	-	modern	fig. 7b
offlinebasic	-	off line colour and off line simple	figs. 12a and 12c
offlinealt	cronoleg	off line colour alt	fig. 12b
sobriety	-	sober judge	fig. 9
xcolseries	-	rainbow serif	fig. 8a

```

modern,
colour scheme=blues,
timeline={%
  dates=1066:1946,
},
event/default colour=ForestGreen,
every text tags+={draw=##1},
]
\end{chronos}

```

`colour scheme` = $\langle name \rangle$

`color scheme`

key

$\langle name \rangle$ should be the name of a colour scheme. A small number of colour schemes are provided by `chronos` (section 7.2); others may be defined using the method explained in section 13.1.

Default: the default set of colours.

Example: `colour scheme=cronoleg`

`chronos` styles may load colour schemes and typically should if they wish to make significant changes.

In addition to the default colours, `chronos` currently provides `blues`, `contninety`, `cronoleg`, `lavender`, `modern`, `offlinebasic`, `offlinealt`, `sobriety` and `xcolseries` (table 2). New colour schemes may be created using the interface explained in section 13.1.

8 Configuration

`Chronos` was designed to be highly configurable. However, by far the *easiest* way to customise a timeline is to load a `chronos` style. See section 7.1.

Most configuration uses the standard key/value interface provided by `TikZ`. In addition, a `\chronosset` is provided for configuring defaults.

Most `chronos` options have local scope. That is, changes do not survive the current group.

However, a small number of options are set *globally*. In these cases, `chronos` keeps track of a list of defaults, as well as the current options, and restores the defaults at the beginning and end of each `chronos` environment. By default, `\chronosset` changes the default values of globalised options, whereas the $\langle chronos preamble \rangle$ does not.

Globalised options saved as default are stored in `expl3` variables named with a package-specific prefix. A similar prefix is used for globalised colours.

`\chronosset` $\{ \langle key-value list \rangle \}$
macro

`\chronosset*` $\{(key\text{-}value\ list)\}$
macro

This should be used to configure `chronos` *outside* the `chronos` environment. It should *not* be used within that environment. The starred version does *not* make any global changes. In general, there is no reason to use the starred version as altering these variables non-globally will have no effect and other variables are not set globally in any case. It is provided ‘just in case’, even though I can’t think of a use-case for it.

`Chronos` sets the following options globally. At the end of the preamble, the active values are saved. These are then restored at the end of each `chronos` environment. This means the results of typesetting a `timeline` should not depend on earlier `timelines` in the same document, a phenomenon which may otherwise result in changes of position and colour, for example. Options set globally:

- the list of `century subheadings` (but neither other subheadings nor headings are globalised);
- most colours and lists of colours;
- whether the last `text tag` of a particular kind (event or period) was placed above or below the `timeline`.

All other settings should behave as usual for PGF/TikZ as they are not handled specially and all other L^AT_EX 3 variables are declared locally.

This approach is intended to ensure that things behave as I expect you to expect, but it is obviously not unlikely you may expect something I don’t expect you to expect. For this reason, it is strongly recommended that document-wide settings be configured in the preamble of your document. `\chronosset` should be used in the document body *only* when you wish to change the document defaults partway through your document. If at all possible, I recommend the use of styles, configured in the preamble, instead, but there will be cases where such an approach may be sub-optimal. `\chronosset` may be used later in such cases.

In particular, you are urged to configure default colours and colour lists, in your preamble. See sections 8.3, 8.8 and 9.5. If you get unexpected colours, please remember that `chronos` defines most colours *globally*. They are *not* limited to the current `chronos` environment. That is, `chronos` lets you customise the colours in many different ways, including many you might wish it did not.

8.1 Documentation Notes

The following notes apply throughout this document.

8.1.1 Font Conventions

This document uses the following typographic conventions.

Bold/***Bold Italics*** are used to emphasise important points, especially ones which might be overlooked.

Italics are used with `<` and `>` for $\{(mandatory\ arguments)\}$, $[(optional\ arguments)]$ and $\langle parameterised\ values\rangle$. When used in the text without delimiters, they are used for emphasis in accordance with standard typographic conventions for English language texts.

Monowidth Typewriter is used for `\macros` (e.g. `\commands`), `environments`, `key names` and `code`.

Sans Serif is used for concepts, elements, package names and class names.

The distinction between a ‘concept’, an ‘element’ and a ‘key’ is not always obvious. Where discussion meanders through the borderlands of fuzzy concepts¹⁹, the font in which a word appears

¹⁹A ‘fuzzy concept’ is one whose extension cannot be precisely defined without arbitrariness. For example, there are clear cases where ‘bald’ applies and equally clear cases where it does not, but there is no non-arbitrary point at which non-baldness becomes baldness. ‘Bald’ is clear in the middle and clear well beyond its scope, but decidedly fuzzy at its edges.

is sometimes arbitrary and the choice should not be taken too seriously. Moreover, some words, such as ‘timeline’, are used for all three.

8.1.2 Keys and Values

Chronos provides a user interface for customisation based almost exclusively on `pgfkeys`.

8.1.2.1 Keys In case you have somehow come across this package shortly after landing in contemporary \TeX land, the basic idea is that the package provides a set of **keys** which you use selectively to customise the output. Some of these keys are simple keywords.

Example: `no connections`,

8.1.2.2 Values When keys permit or require arguments, the arguments are called **values**. A given key will generally require a $\langle value \rangle$ of some particular sort, as explained for each key below.

Some `chronos` keys permit an argument, but don’t require it.

Example: `frame`,

Example: `frame=true`,

Example: `frame=false`,

The above are all valid (with the first two being equivalent).

Other `chronos` keys require one or more arguments.

Example: `colour=Cerulean`,

Example: `heading={chronos year -150}{chronos year 250}{past}`,

`Chronos` frequently requires multiple arguments to be separated by colons, because this often seemed less error-prone than multiplying curly brackets in complex cases.

Example: `dates={{-100}-01-12}:{900-12-24}`,

In some instances, where a proliferation of colons seemed no less an invitation to error than one of curly brackets, the colon cases are convenience keys, which you can avoid through the use of two or more alternate keys to specify items separately.

8.1.2.3 Key-Value Lists $\langle key\text{-}value\ list \rangle$ s are comma-separated lists of items, each of which is either a simple $\langle key\text{-}name \rangle$ or a $\langle key\text{-}name \rangle = \{ \langle comma\text{-}separated\ list\ of\ values \rangle \}$. In general, the $\langle comma\text{-}separated\ list\ of\ values \rangle$ will be a `TikZ` $\langle key\text{-}value\ list \rangle$, though it may sometimes be appropriate to include further `chronos` keys.

Example: `event/line={draw=blue,draw opacity=.75}`

8.1.3 Key Specifications

Key specifications in this document look like this:

<code>key name</code> <i>key type</i>	<code>=</code>	$\langle argument\ specification \rangle$	$tag1, tag2, tag3, \dots$
		$\langle Description\ of\ key\ and\ explanation\ of\ usage. \rangle$	
		Default: $\langle key's\ default\ value \rangle$	
		Initially: $\langle key's\ initial\ value \rangle$	
		Example: $\langle example\ of\ usage \rangle$	
		$\langle Commentary. \rangle$	

Table 3: chronos key types.

Key type	Description	Example
<i>boolean key</i>	Controls a boolean or toggle i.e. a conditional.	
<i>choice key</i>	Selects from a list of possible options.	
<i>comma-separated list key</i>	Processes or stores a comma-separated list of things.	
<i>colour key</i>	Specifies a colour.	
<i>colour list key</i>	Special kind of comma-separated list key which stores a list of colours.	
<i>date key</i>	Specifies a date or dates.	
<i>date format key</i>	Specifies one or more date output formats.	
<i>dimension key</i>	Specifies a T _E X dimension.	
<i>key</i>	Some other kind of key.	
<i>style</i>	A PGF/TikZ style.	

Here, **key name** is the name of the key, *key type* is the type of key, *⟨argument specification⟩* specifies the number, kind and format of the value or values the key expects and *tag1, tag2, tag3, ...* indicates to elements of which **tag** or **tags** the key applies. See table 3 for an explanation of the types of key **chronos** uses. See sections 6 and 6.2 for information about **tags**.

If no initial value is specified, the default value is also the initial value. Where both an initial and a default value are specified, the default is the value used if the *⟨key name⟩* is given without an argument and the initial value is the value used if *⟨key name⟩* is not used at all. This terminology follows the usage in **pgfkeys** and is especially prevalent in the handling of boolean keys, where it is common for the initial value to be **false**, but the default value to be **true**.

Schematically,

```

\begin{chronos}% ^^A initial value used
[
  % ^^A other keys
]
\end{chronos}
\begin{chronos}% ^^A default value used
[
  % ^^A other keys
  key name,
]
\end{chronos}
\begin{chronos}% ^^A new value used
[
  % ^^A other keys
  key name=new value,
]
\end{chronos}

```

8.1.4 Syntax Notes

See section 8.1.5 for the *syntax* of dimension keys, where *plus* and *prime* have different meanings.

8.1.4.1 Slash (/) Where a forward slash (/) occurs in a key, it indicates a context-specific key. For those familiar with PGF keys, this corresponds to a path under **/chronos**.

Example: **life/connection**

indicates a key affecting **connection(s)** belonging to elements of type **life**.

8.1.4.2 Plus (+) A plus sign (+) at the end of a key indicates that the key *adds* to any pre-existing list. This form is generally available when the base key replaces, rather than adding

to, any pre-existing list.

```
timeline line={draw=black,fill=green},
         timeline line+={opacity=.8},
```

is equivalent to

```
timeline line={draw=black,fill=green,opacity=.8},
```

A plus at the end of a dimension key indicates that the dimension key *adds* the value given to the current value of the dimension.

8.1.4.3 Prime (') A prime (') at the end of a key indicates that the key *replaces* any pre-existing list. This form is generally available when the base key adds to, rather than replacing, any pre-existing list.

```
century subheadings={15,17,19}{th},
century subheadings'={13,14}{th},
century subheading={21}{st},
```

is equivalent to

```
century subheadings'={13,14}{th},
century subheading={21}{st},
```

and will result in subheadings being created for the 13th, 14th and 21st centuries (assuming the timeline covers these time periods and the relevant coordinates exist).

A prime at the end of a dimension key, or at the end except for a plus ('+), indicates that the dimension key expects a \TeX dimension, as opposed to an expression to be evaluated by `pgfmath`.

8.1.5 Dimension Notes

8.1.5.1 Dimensions Each key described as a dimension key is available in six forms²⁰:

$\langle \text{dimension key} \rangle$ = $\{ \langle \text{pgfmath-parsable dimension} \rangle \}$
dimension key

The dimension key parses the $\langle \text{specified value} \rangle$ using `pgfmath` and assigns the result in points as the dimension. This base form, which is typically the only form explicitly listed in this documentation, is slow but flexible. Unless otherwise noted, the existence of the base form implies the availability of all six variants.

$\langle \text{dimension key} \rangle'$ = $\{ \langle \text{dimension} \rangle \}$
dimension key

The dimension key expects a \TeX $\langle \text{dimension} \rangle$, complete with units, which it assigns directly. This is faster but less flexible.

$\langle \text{dimension key} \rangle+$ = $\{ \langle \text{pgfmath-parsable dimension} \rangle \}$
dimension key

The dimension key parses the expression $(\langle \text{specified value} \rangle + \langle \text{existing value} \rangle)$ with `pgfmath` and assigns the result in points. This is slower but more flexible.

$\langle \text{dimension key} \rangle'+$ = $\{ \langle \text{dimension} \rangle \}$
dimension key

The dimension key expects a \TeX $\langle \text{dimension} \rangle$, complete with units, which it adds to the $\langle \text{existing dimension value} \rangle$ directly. This is faster but less flexible.

$\langle \text{dimension key} \rangle-$ = $\{ \langle \text{pgfmath-parsable dimension} \rangle \}$
dimension key

²⁰Occasionally, a convenience key may only support the prime, prime-plus and prime-minus forms. Where this applies, the limitation is noted in the description.

The dimension key parses the expression ($\langle\textit{specified value}\rangle - \langle\textit{existing value}\rangle$) with `pgfmath` and assigns the result in points. This is slower but more flexible.

`\dimension key' - = {<dimension>}`
dimension key

The dimension key expects a $\text{T}_{\text{E}}\text{X}$ $\langle\textit{dimension}\rangle$, complete with units, which it subtracts from the $\langle\textit{existing dimension value}\rangle$ directly. This is faster but less flexible.

When dimension keys end in prime, prime-plus or prime-minus, $\langle\textit{dimension}\rangle$ s must be given as $\text{T}_{\text{E}}\text{X}$ dimensions complete with units and may not require calculation.

Example: `timeline height'=10mm`

Example: `timeline border height'+=20pt`

Example: `timeline width'-=2em`

When dimension keys do not include prime, any value which can be parsed by `pgfmath` is valid.

Example: `timeline height=.01\texttheight`

Example: `timeline border height+=1.5\headrulewidth`

Example: `timeline width-=0.05\linewidth+1.5pt`

8.1.6 Date Specification Notes

8.1.6.1 Date Format Specifications A $\langle\textit{date format specification}\rangle$ ($\langle\textit{date format spec.}\rangle$) is an expression using the syntax explained in section 8.2.2.

Example: `date format={!d !B !Y !E}`

8.1.6.2 Dates $\langle\textit{date}\rangle$ s must be specified using the syntax explained in section 8.2.1.

Example: `dates={{-200}-04-05}:{200-12-31}`

8.1.7 Colour Notes

8.1.7.1 Colours $\langle\textit{colour}\rangle$ s should be colour names or mixtures supported by `xcolor`.

Example: `colour=WildStrawberry`

Example: `foreground=WildStrawberry!50!black`

8.1.7.2 Colour Lists $\langle\textit{colour list}\rangle$ s are comma-separated lists of colour names or mixtures supported by `xcolor`.

Example: `life/colours above={blue,green,blue!50!green}`

8.1.7.3 Colour `colour` and `color` are synonyms in key names.

Example: `colours below={black,gray}`

Example: `colors below={black,gray}`

8.2 Dates

Chronos uses a fixed format for date input and offers a flexible format for date output.

8.2.1 Input

All date keys expect one or two arguments specifying a date or dates in the format $\{\{Y\}-M-D\}$. Y , M and D must be integers. If Y is negative, the date is interpreted as BCE; otherwise CE is assumed. The additional curly brackets around Y are *mandatory* for negative values.

Table 4: Date and year format specification codes.

code	meaning	example output	date format specifier?	year format specifier?
!a	short weekday name	Mon	✓	—
!A	full weekday name	Monday	✓	—
!b	short month name	Jan	✓	—
!B	full month name	January	✓	—
!c	semi-shortened year	900	✓	✓
!d	day of the month	23	✓	—
!E	era	BCE or CE label	✓	✓
!m	month number	01	✓	—
!q	minus if year is BCE	-	✓	✓
!Q	minus if year is BCE; plus for CE	+	✓	✓
!y	last two digits of year	66	✓	✓
!Y	year	1066	✓	✓

```
start date={{-3000}-05-23},
end date={1500-12-04},
```

It is also permissible to specify only a year, in which case `chronos` will specify values for the month and day. Hence,

```
dates={-245}:789,
```

is also valid. Where two dates are required, `dates` offers a more concise syntax, but dates may always be specified singly if this is preferred.

8.2.2 Output

All date format keys expect one or three arguments using the syntax specified in table 4.

Example: `date format={ B d, Y}`

This would result in a full month name followed by the day of the month, then a comma and finally the year.

Each character in the format is either translated into an element of the date format or passed through as is. This includes punctuation and spaces. (Note that macros etc. won't work here because the macro will be broken down and 'translated' token-by-token.)

The format codes, listed in table 4, are mostly a subset of the format codes provided by GNU's date command, with a few extras not relevant to GNU²¹.

A subset of the date-specification codes (as indicated in table 4) is available to customise the formatting of years on the timeline itself. In the case of the timeline, era labels may instead be added at each end to avoid the clutter of including BCE or CE with every year.

`date format` = $\{ \langle \text{date format specification} \rangle \}$
date format key

When used in the $\langle \text{chronos preamble} \rangle$ or in `\chronosset`, sets the default format for dates.

Default: `!d/!m/!Y\thinspace !E` (with eras)

Default: `!d/!m/!Y` (without eras)

`event/date format` = $\{ \langle \text{date format specification} \rangle \}$
date format key

event

²¹I am grateful to Joseph Wright for providing the code implementing this at [T_EX StackExchange: 327642](https://tex.stackexchange.com/questions/327642).

When used in the *<chronos preamble>* or in `\chronosset`, sets the default format for event dates. *This key overrides show eras, without eras, full dates and only years for elements of tag type event.*

Default: `!d!/m!/Y\thinspace !E` (with eras)

Default: `!d!/m!/Y` (without eras)

The following keys set `event/date format` conditionally. This may be used to switch between formats showing eras or only years and no eras or full dates while ensuring uniformity of all formats with or without eras, for example. For instance, it may make little sense to use full dates for events where only the year is known or which occurred when different calendars were used, but you might still want full dates for other cases. *These keys override show eras, without eras, full dates and only years for elements of tag type event.*

`event/show eras/full` = *{<date format specification>}* *event*
date format key

When used in the *<chronos preamble>* or in `\chronosset`, sets the default format to use for event when showing full dates with eras.

Default: `!d!/m!/Y\thinspace !E`

`event/show eras/only years` = *{<date format specification>}* *event*
date format key

When used in the *<chronos preamble>* or in `\chronosset`, sets the default format to use for event when showing only years with eras.

Default: `!Y\thinspace !E`

`event/without eras/full` = *{<date format specification>}* *event*
date format key

When used in the *<chronos preamble>* or in `\chronosset`, sets the default format to use for event when showing full dates without eras.

Default: `!d!/m!/Y`

`event/without eras/only years` = *{<date format specification>}* *event*
date format key

When used in the *<chronos preamble>* or in `\chronosset`, sets the default format to use for event when showing only years without eras.

Default: `!Y`

life and period are more complex as date ranges are involved, but the basic structure works in the same way.

`life/date formats` = *{<date format spec.>}:{<date format spec.>}:{<date format spec.>}* *life, period*
`period/date formats`
date format key

When used in the *<chronos preamble>* or in `\chronosset`, sets the default formats for life or period dates. In these cases, we have two dates — either a birth and death or a start and end. You might want different formats for the two and you might want different formats when the first date is BCE and the second CE. Hence, we need to specify three formats. The first argument specifies the format to use for the birth or start date when the death or end date occurs in the same era. The second specifies the format to use for the first date when the eras differ. The third specifies the format to use for the death or end date. *These keys override show eras, without eras, full dates and only years for elements of tag types life and period respectively.*

Default: `{!d!/m!/Y}:{!d!/m!/Y\thinspace !E}:{!d!/m!/Y\thinspace !E}` (with eras)

Default: `{!d!/m!/Y}:{!d!/m!/Y}:{!d!/m!/Y}` (without eras)

The following keys override date formats for elements of tag types life and period respectively. They work in the same way as those explained above for event.

`life/show eras/full` = *{<date format spec.>}:{<date format spec.>}:{<date format spec.>}* *life, period*
`period/show eras/full`
date format key

When used in the *<chronos preamble>* or in `\chronosset`, sets the default formats to use for life or period when showing full dates with eras.

Default: `{!d/!m/!Y}:{!d/!m/!Y\thinspace !E}:{!d/!m/!Y\thinspace !E}`

`life/show eras/only years` = `{(date format spec.):{(date format spec.):{(date format spec.)}}` *life, period*
`period/show eras/only years`
date format key When used in the `<chronos preamble>` or in `\chronosset`, sets the default formats to use for life or period when showing only years with eras.

Default: `{!Y}:{!Y\thinspace !E}:{!Y\thinspace !E}`

`life/without eras/full` = `{(date format spec.):{(date format spec.):{(date format spec.)}}` *life, period*
`period/without eras/full`
date format key When used in the `<chronos preamble>` or in `\chronosset`, sets the default formats to use for life or period when showing full dates without eras.

Default: `{!d/!m/!Y}:{!d/!m/!Y}:{!d/!m/!Y}`

`life/without eras/only years` = `{(date format spec.):{(date format spec.):{(date format spec.)}}` *life, period*
`period/without eras/only years`
date format key When used in the `<chronos preamble>` or in `\chronosset`, sets the default formats to use for life or period when showing only years without eras.

Default: `{!Y}:{!Y}:{!Y}`

`every date format` = `{(date format specification)}`
date format key

Sets *all* date formats for *all* tags and the default format to `<date format specification>`. This key does not affect the formatting of years, minor years or eras on the timeline itself.

Default: none

Initially: none

`bce year label` = `<text>`
key

The label to use if showing the BCE era in `text tags`. Note this is not the label used if marking eras on the timeline, unless including them as part of year labels.

Default: `\textsc{bce}`

```
\begin{chronos}
[
  bce year label=BCE,
]
\end{chronos}
```

The label is available as `\bceyearlabel` inside the environment `chronos`. In addition, it is made available at the end of the preamble if the command is not otherwise defined.

`ce year label` = `<text>`
key

The label to use if showing the CE era in `text tags`. Note this is not the label used if marking eras on the timeline, unless including them as part of year labels.

Default: `\textsc{ce}`

```
\begin{chronos}
[
  ce year label=\textsc{ad},
]
\end{chronos}
```

The label is available as `\ceyearlabel` inside the `chronos` environment. In addition, it is made available at the end of the preamble if the command is not otherwise defined.

The timeline itself features only years (but see `event years` on line for a limited exception).

`year format` = `{(year format specification)}`
date format key

When used in the `<chronos preamble>` or in `\chronosset`, sets the default format for years. This is the format used to format ‘major’ years on the timeline.

Default: `!Y\thinspace !E` (with eras)

Default: `!Y` (without eras)

`minor year format` = `{<year format specification>}`
date format key

When used in the `<chronos preamble>` or in `\chronosset`, sets the default format for ‘minor’ years.

Default: `!c`

The idea is that you might want, say, four-digit years every half century and three-digit years every hundred years in between.

`timeline/timeline mark eras` = `true|false`
boolean key

Should era labels be included at the end(s) of the timeline? Note that a label will only be shown if the dates the timeline covers include some in the relevant era. So if your timeline starts at 500 CE, the BCE will be omitted and if it ends at 200 BCE, the CE will be omitted.

Default: `true`

Initially: `false`

`timeline bce label` = `<text>`
key

The label to use if marking the BCE era on the timeline. Note this is not the label used if showing eras in text tags.

Default: `BCE`

```
\begin{chronos}
[
  timeline bce label=BC,
]
\end{chronos}
```

The label is available as `\celabel` inside the `chronos` environment. In addition, it is made available at the end of the document preamble for general use if the command is not otherwise defined.

`timeline ce label` = `<text>`
key

The label to use if marking the CE era on the timeline. Note this is not the label used if showing eras in text tags.

Default: `CE`

```
\begin{chronos}
[
  timeline ce label=AD,
]
\end{chronos}
```

The label is available as `\celabel` inside the `chronos` environment. In addition, it is made available for general use at the end of the document preamble if the command is not otherwise defined.

8.2.3 The Problem of the Non-Existent Year

Chronos uses `pgfcalendar` to calculate Julian day numbers from dates when constructing the timeline. Generally, this works well, but an issue occurs if your timeline spans the two eras (BCE

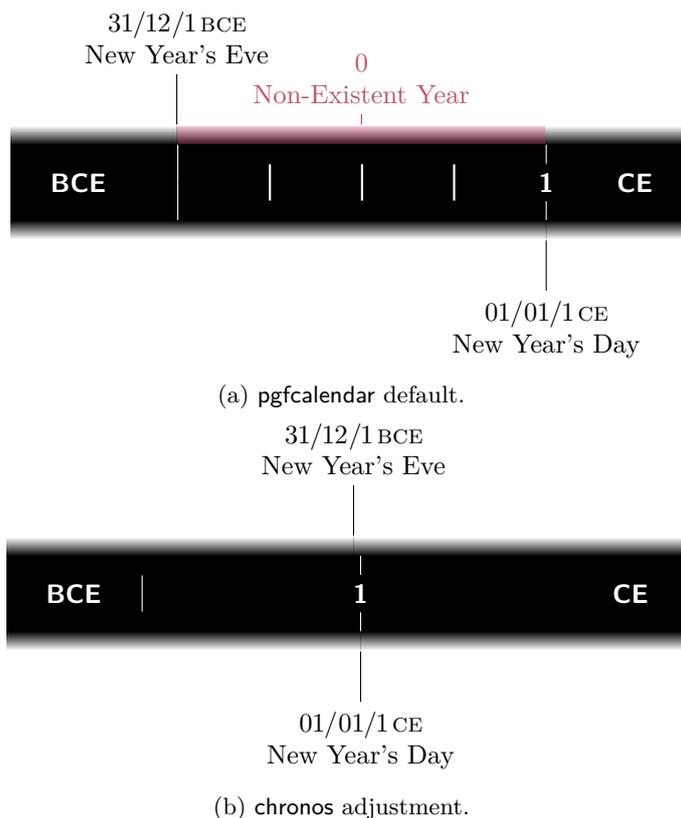


Figure 18: The problem of the non-existent year.

and CE). Pgfcalendar assumes there was a year zero (fig. 18a), which historians will assure you there was not.

By default, `chronos` corrects for this (fig. 18b), but the correction can be switched off if desired (fig. 18a).

```
timeline/year zero = true|false
    boolean key
```

Whether to tolerate the year zero.

Default: `true`

Initially: `false`

If there is no year zero, certain complications arise. First, what should be marked on the timeline at the ‘era switch’? Second, if you ask `chronos` to mark every hundredth year, say, you probably do not expect it to mark 200 BCE, 100 BCE, 1 CE, 101 CE and so on. Moreover, you might want to do something such as this

```
\foreach \i in {-100,-50,...,300} \node [red,inner sep=2.5pt] at (chronos year \i) {};
```

This seems reasonable, but will fail if `chronos year 0` doesn’t exist.

`Chronos` attempts to solve these problems by handling the ‘era switch’ as a special case. First, if there is no year zero, it will create *two* coordinates at the switch, provided you have asked it to mark something at this point. `chronos year 0` will exist, as far as `chronos` is concerned, at the same point as `chronos year 1`. This means you can loop over the era switch in the normal way and expect sensible output, but you can *also* refer to `chronos year 1`, even if you only asked every hundredth year to be marked from 100 BCE.

Second, `chronos` provides a special option for configuring what is marked on the timeline at the switch of eras.

`timeline/mark at era switch = true|false`
boolean key

Whether to use a mark rather than a year at the era switch. If false, the year (e.g. ‘1’) is used; if true, a mark is used instead (illustrated in fig. 18b, though the format will depend on how the timeline is configured).

Default: `true`

Initially: `false` (if showing every year)

Initially: `true` (otherwise)

Note that this option only configures what is marked if something is. If you ask `chronos` to mark every hundredth year from 150 BCE to 400 CE, nothing will be marked at the era switch (but `chronos` will write a warning to the log). `Chronos` won’t do that by default, but, if you insist, it will take you at your word.

`timeline/year at era switch = true|false`
boolean key

Whether to use a year rather than a mark at the era switch. This is simply a convenience key which does the opposite of `mark at era switch`.

Default: `true`

Initially: see `mark at era switch`.

8.3 Basic Colours

`Chronos` uses (or may use) two basic colours: one for foreground and one for background elements.

`background = <colour name>`
colour key

This is the ‘main background colour’ for the picture as a whole. This colour is accessible within the `chronos` environment as `chronos main background colour` or `chronos main background color`. Whether it is used and, if so, how, depends on other settings. By default, it is used to determine the colours for the timeline itself and is the basis for the colours used in some tags. It is also used in some standard `chronos styles`.

Default: `white`

```
\begin{chronos}
[
  background=magenta,
]
\end{chronos}
```

`foreground = <colour name>`
colour key

This is the ‘main foreground colour’ for the picture as a whole. This colour is accessible within the `chronos` environment as `chronos main colour` or `chronos main color`. Whether it is used and, if so, how, depends on other settings. By default, it is used to determine the colours for the timeline itself and is the basis for the colours used in some tags. It is also used as the default colour for connections, lines and text tags and in some standard `chronos styles`.

Default: `black`

```
\begin{chronos}
[
  foreground=red,
]
\end{chronos}
```

For other colours, see sections 8.4.5 and 8.8.

8.4 Timeline

See section 6.1 for an overview of the timeline’s components and construction.

Placing different elements on different layers enables the same basic building blocks to result in different styles, but the blocks may also be configured directly. The layers on which the connections and lines of items connected to the timeline are drawn also affects the appearance. For example, putting connections behind the border results in circular chronos connectors appearing as semicircles. Chronos’s use of layers is explained in sections 6.4 and 10.

`connections on` = background|middle ground|main|foreground|overlay
`lines on`
`timeline/timeline on` Which layer each type of element should be placed on. Aside from main these are not standard layers. In particular, background is not the standard TikZ background layer, but instead refers to the chronos background layer.
`timeline/border on`
choice key

Default: dependent on other options

See section 6.4.

The timeline should be configured using the following key.

`timeline` = $\{(key\text{-value list})\}$
key
 $\langle key\text{-value list} \rangle$ should be a list of chronos keys from the timeline configuration options. These keys may also be accessed more verbosely as `/chronos/timeline/⟨key name⟩` or, in the *chronos preamble* or in `\chronosset` as `timeline/⟨key name⟩`. Some may also work without the `timeline/` prefix, but *this is not guaranteed and may break without notice in future releases*.

```
\begin{chronos}
[
  timeline={% timeline configuration
    dates={1310-02-03}:{1350-06-07},
    timeline foreground=black,
    timeline background=gray,
    minor years,
    timeline height=5pt,
    timeline width=\textwidth,
    timeline era margin=10pt,
    major step font=\sffamily\bfseries,
    minor step font=\sffamily\bfseries\small,
    timeline minor marks,
    timeline marks,
    timeline years=above,
  },
]
\end{chronos}
```

Timeline configuration keys are prefixed with `timeline/` in this manual.

8.4.1 Timeline Dates

`timeline/dates` = $\langle start\ date \rangle:\langle end\ date \rangle$
date key

The first and last date to be represented on the timeline. Dates must be specified as explained in section 8.2. This key offers a more compact syntax as an alternative to the keys `start date` and `end date` (or `start` and `end`) explained below. That is

```
\begin{chronos}
[
  timeline={%
```

```

    dates={1310-02-03}:{1350-06-07},
    % equivalent to
    start date={1310-02-03},
    end date={1350-06-07},
    % equivalent to
    start={1310-02-03},
    end={1350-06-07},
  },
]
\end{chronos}

```

`timeline/start date` = `{(date)}`

`timeline/start`
date key

The first date to represent on the timeline, specified as explained in section 8.2.

```

\begin{chronos}
[
  timeline={%
    start date={1310-02-03},
    % equivalent to
    start={1310-02-03},
  },
]
\end{chronos}

```

`timeline/end date` = `{(date)}`

`timeline/end`
date key

The last date to represent on the timeline, specified as explained in section 8.2.

```

\begin{chronos}
[
  timeline={%
    end date={1350-06-07},
    % equivalent to
    end={1350-06-07},
  },
]
\end{chronos}

```

8.4.2 Timeline Dimensions

See note 8.1.5.1.

The dimensions of the timeline line and border are illustrated in fig. 19.

The total height of the timeline is a function of the dimensions `timeline height` and `timeline border height`:

$$\text{timeline height} + 2 \cdot \text{timeline border height}$$

The total width is `timeline width`. The width includes the width used to represent the time covered by the timeline and twice the `timeline margin`. If era labels are used, the width also includes the space used for these²² and the `timeline era margins`.

For example,

```

\begin{chronos}
[
  timeline={%
    timeline height=10mm,
    timeline border height=2.5mm,

```

²²I am grateful to Martin Scharrer for providing the code implementing this at [TeX StackExchange: 56405](https://tex.stackexchange.com/questions/56405).

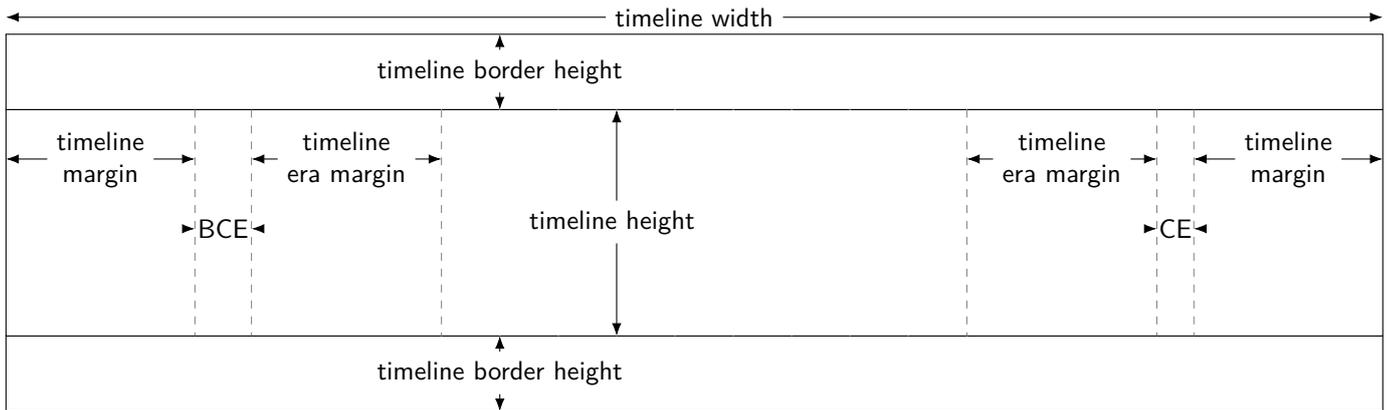


Figure 19: Timeline dimensions.

```

    timeline width=200mm,
    timeline mark eras,
    timeline margin=5mm,
    timeline era margin=2.5mm,
    dates={-200}:2000,
  },
]
\end{chronos}

```

would result in a total timeline height of 15mm and a total timeline width of 200mm. The width used to represent the years from 200 BCE to 2000 CE would be

$$200\text{mm} - 2 \cdot 5\text{mm} - 2 \cdot 2.5\text{mm} - \text{width of BCE label} - \text{width of CE label}$$

that is,

$$185\text{mm} - \text{width of BCE label} - \text{width of CE label}$$

`timeline/timeline height` = $\langle dimension \rangle$

`timeline/height`
dimension key

The height of the timeline excluding any border.

Default: dependent on other options

For example,

```

timeline={
  timeline height'=10mm,% we can use ' here
},

```

`\timelineht` *macro* The height of the timeline. This macro is available *only at the end of the $\langle chronos preamble \rangle$ and can be considered reliable only within the $\langle timeline specification \rangle$* ²³. Despite its unreliability, early availability is essential to some chronos styles definitions. In these cases, the chronos style is responsible for ensuring accuracy (or compensating for inaccuracy). In standard cases, this happens automatically, even though it is not guaranteed. However, if you neither load a chronos style nor configure dimensions explicitly, you should not try to use this macro before the timeline is constructed.

`timeline/timeline border` = $\langle dimension \rangle$

`height`
dimension key

The height of each of the upper and lower borders.

²³Note that the unreliability applies to the internal macro, too.

Default: dependent on other options

For example,

```
timeline={
  timeline border height'+=2.5pt,% we can use ' here
},
```

`\timelineborderht` macro The height of the border. This macro is available *only within the* `\timeline` specification).
`timeline/timeline width` = `\langle dimension \rangle`

`timeline/width` dimension key The total width of the timeline, including margins.

Default: `\textwidth`

For example,

```
timeline={
  timeline width=.75\paperheight,% we cannot use ' here
  timeline width'-=10mm,% we can use ' here
},
```

`\timelinewd` macro The width of the timeline. This macro is available *only within the* `\timeline` specification).
`timeline/timeline margin` = `\langle dimension \rangle`

`timeline/margin` dimension key The horizontal space to allow at each of the two ends of the timeline.

Default: 15pt

For example,

```
timeline={
  timeline margin'+=-2.5pt,% we can use ' here
},
```

`timeline/timeline era` = `\langle dimension \rangle`

`margin` dimension key The horizontal space to allow between the first/last point on the timeline and the era labels.

Default: 15pt

For example,

```
timeline={
  timeline era margin+=0.05,% we can't use ' here
},
```

The following keys determine dimensions of the chronos picture as a whole. They do not affect the dimensions of the timeline itself.

`headings border` = `\langle dimension \rangle`
dimension key

The distance between the top of the highest level and the top of the space used for headers.

Default: 15pt + `\langle headings drop \rangle` + `\langle upper subheadings drop \rangle` + `\langle lower subheadings drop \rangle` (if there are one or more levels above the timeline)

Default: 5pt + `\langle headings drop \rangle` + `\langle upper subheadings drop \rangle` + `\langle lower subheadings drop \rangle` (otherwise)

`headings drop` = `\langle dimension \rangle`
dimension key

The distance between the top of the border and the headings.

Default: 0pt (if headings are omitted)

Default: 15pt (if headings are used)

Note that you should set this explicitly to 0pt if using subheadings without headings.

`subheadings drops` = $\{\langle dimension 1 \rangle\}:\{\langle dimension 2 \rangle\}$
dimension key

The distances between the headings and upper subheadings and between the tops of the upper subheadings and lower subheadings.

Default: 0pt:0pt (if headings are omitted)

Default: 12pt:10pt (if headings are used)

Note that you should set this explicitly to 0pt:0pt, $\langle dimension \rangle:0pt$ or $0pt:\langle dimension \rangle$ if using headings without upper subheadings and/or lower subheadings or only one of upper subheadings or lower subheadings.

`headings drops'` = $\{\langle dimension 1 \rangle\}:\{\langle dimension 2 \rangle\}:\{\langle dimension 3 \rangle\}$

`headings'+`
`headings'-`
dimension key

A convenience key equivalent to setting `headings drop'` to $\langle dimension 1 \rangle$ and `subheadings drops'` to $\langle dimension 2 \rangle$ and $\langle dimension 3 \rangle$. *Note that only the ' forms are available.* For pgfmath support, use `headings drop` and `subheadings drops`.

`outer border` = $\langle dimension \rangle$

dimension key

If a frame is created, this is the outer border. In effect, the bounding box will be set to be this distance from the frame, less half the line width used to draw it.

Default: 5pt

`borders'` = $\{\langle dimension \rangle\}:\{\langle dimension \rangle\}:\{\langle dimension \rangle\}:\{\langle dimension \rangle\}:\{\langle dimension \rangle\}:\{\langle dimension \rangle\}$

`borders'+`
`borders'-`
dimension key

Sets the headings border, top border, right border, bottom border, left border and outer border in one go. *Note that only the ' forms are available.* For pgfmath support, use `top border`, `right border`, `left border`, `bottom border` and `headings border`.

If you're not sure what this key does or uncertain whether to use it, it is not the key you are looking for. Setting the `outer border` and `headings border` suffices in most cases.

`top border` = $\langle dimension \rangle$

`right border`
`bottom border`
`left border`
dimension key

If the frame does not use the bounding box, these dimensions determine the internal margin between each of the top of the headings, the timeline's right end, the bottom of the lowest level, the timeline's left end and the frame, less half the line width used to draw the frame.

Default: 0pt

Most people should let the frame use the bounding box, which is the default, and leave these dimensions alone.

8.4.3 Timeline Marks and Years

Chronos offers two primary styles of timeline. In one, the line has sufficient vertical depth (`timeline height`) for years, era labels and marks to be drawn on the timeline itself. In the other, the timeline may be much thinner, with marks, era labels and years drawn above or below the line. In this case, the marks appear to grow out from the line and the year labels float slightly above or below.

It is also possible to use `chronos` to draw a line with neither marks nor years. Alternatively, you might want to create 'invisible' marks or years, which may be useful for placement purposes²⁴. Figure 19 shows a timeline in which this has been done by setting the foreground and background colours equal. The nodes are used to place the arrows and labels illustrating the various dimension keys.

²⁴You don't need this simply to connect elements to the timeline. `chronos` doesn't depend on the creation of marks or years for that purpose.

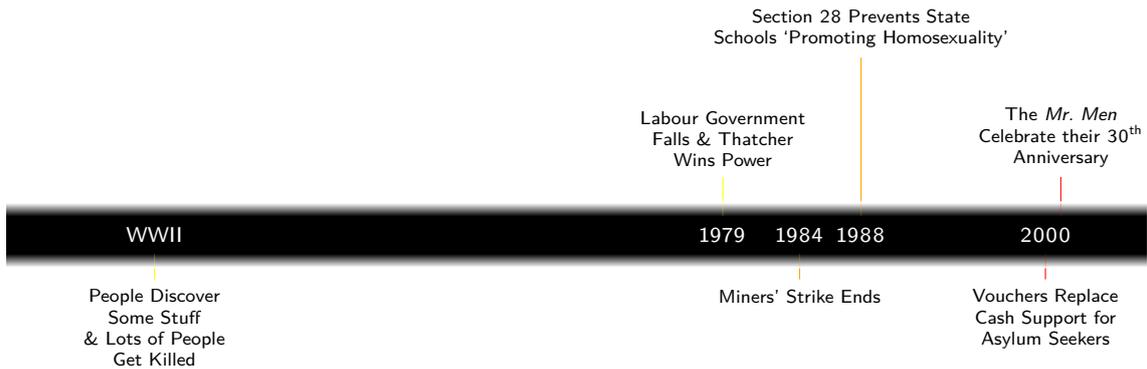


Figure 20: Illustration of event years on line.

`timeline/timeline years` = on line|off line|above|below|none
choice key

Whether years (and any era labels and marks) should be created on the timeline, off it or not at all and, if they should be off the timeline, whether they should be above or below it. The options are mutually exclusive, except that `off line` implies either `above` or `below`. See also `minor years`, `timeline marks`, `timeline minor marks` and `timeline bare marks`, which further determine what exactly is shown.

Default: none

Initially: on line

it may actually make sense to write something like

```
\begin{chronos}
[
  timeline={%
    timeline years=off line,
    timeline years=none,
  },
]
\end{chronos}
```

if one wants an off-line style of line with no years or marks. I don't know why one *would* want such a thing, but the possibility is there.

`none` is actually intended to support a particular style of event-only timeline, in which the dates are created on the line itself.

`event years on line` *key* Don't create regular year labels or marks on the timeline itself. Instead, put the years of subsequently added events onto the line. This option creates a timeline suitable for showing years on the timeline, but doesn't create any labels when drawing the line itself.

Assuming `timeline years` is not set to `none`, as it is if `event years on line` is enabled, the following keys determine how and where `chronos` represents time on (or off) the timeline itself. The primary concepts here are those of `major steps` and `minor steps`. The space available to represent time on the timeline (see section 8.4.2) is divided into `major steps` and, optionally, further divided into `minor steps`. These can be highlighted with `timeline marks` and `timeline minor marks` and are set using `step major year` and `step minor year`.

In addition to years, `timeline bare marks` may be used to create unlabelled subdivisions at intermediate points. In the standard case, the value of `step divisions` is used to divide the distance equally. For example, if you specify 5, `chronos` will use 4 lines to subdivide each. No attempt is made to place these so they correspond to any particular date: if you request 12, `chronos` will not make the division for February smaller than the one for December.

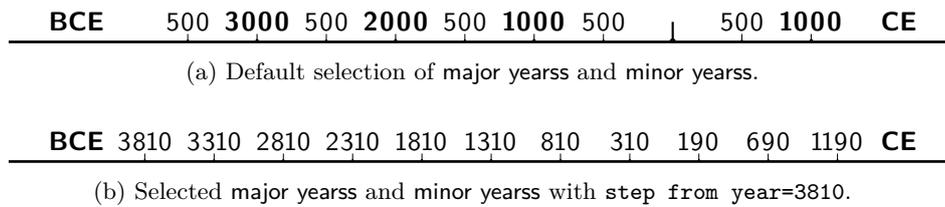


Figure 21: Default (fig. 21a) and non-default (fig. 21b) selection of major years and minor years when `dates={-3814}:1213`, `step major year=1000` and `step minor year=500`.

However, if a timeline is short, `chronos` proceeds differently. ‘Short’ refers to temporal duration rather than dimension and includes any timeline which begins and ends in the same year or in consecutive years.

`timeline/minor years` = `true|false`
boolean key

Whether to label minor years, in addition to major years.

Default: `true`

Initially: `true`

`timeline/step major year` = `{(positive integer)}`
`timeline/step major years`
key

How often to label major years on the timeline if showing them. Use this key if you want a larger or bolder font and/or a different date format and/or thicker or longer marks to be used for some year labels. You can also use this key if you want all year labels on the timeline to use the same format. For example, you might want to print the full 4 digits of the year each thousand years.

Default: dependent on other options

`timeline/step minor year` = `{(positive integer)}`
`timeline/step minor years`
key

How often to label minor years on the timeline if displaying them. The idea is that you might want a smaller or lighter font and/or a different date format and/or thinner or shorter marks to be used for intermediate year labels. For example, you might want to print full years only every millennium and the last 3 digits of the year each century.

Default: dependent on other options

`Chronos` labels minor years only if labelling major years. Although the package attempts to correct the result if only minor years are requested, it is better to use `step minor year` only in conjunction with major years.

`timeline/step year` = `{(positive integer)}`
`timeline/step years`
key

How often to label years on the timeline, if you want them all to be formatted in the same way. This key sets `step major years` internally and unsets `step minor year`.

Default: dependent on other options

`Chronos` tries to label years *modulo* the `step major year` and `step minor year` (or `step year`). This means you can start the timeline at 3,814 BCE, request major years every millennium and minor years every half millennium without worrying about which year should be the first (labelled) year. Figure 21 illustrates `chronos`’s default choices in this case. Note that the first year is *not* determined by the start date alone in fig. 21a, but is determined in conjunction with `step major year` and `step minor year` so that -1 BCE ends (and 1 CE begins) at a major year and the turn of millennia generally occur at major years, while the first minor year is 3,500 BCE.

`timeline/step from year` = `{(integer)}`
key

Do not use this key unless `chronos` produces undesirable results by default. If for some reason you do *not* want years on the timeline to be determined modulo `step major year` and `step`

minor year, you may tell `chronos` where to begin stepping from. In this case, `chronos` will issue a warning, but it will implement your choice.

Default: dependent on other options

Note that fig. 21b effectively includes no major yearss because `chronos` tests whether the current year is modulo the `step major year` when deciding how to format the year label and marks.

`chronos year <YYYY>` Every major year and minor year receives a name: a `node` or `coordinate` is created with the name `chronos year <YYYY>` for CE and `chronos year -<YYYY>` for BCE. No zeros are added, so years with fewer than four digits get nodes or coordinates with names such as `chronos year -1`. `Chronos` creates all years at the beginning of the year i.e. 1st January. (This is analogous to a ruler which marks each centimetre at its beginning.)

`chronos origin` If the timeline spans the switch of eras from BCE to CE *and* the years represented on the timeline are modulo an additional coordinate named `chronos origin` is created at the era switch point, `chronos year 1`.

`chronos year 0` If `year zero` is `false`, as it is by default, a third coordinate named `chronos year 0` is created at `chronos origin`²⁵.

`timeline/step divisions` = `{<positive integer>}`
key

Whether the timeline should be further subdivided between major and/or minor years using bare marks and, if so, how many sub-divisions should be made. These are simple subdivisions of the distance between points. Unlike the labels/marks made for years, they do not involve calculations involving dates and are not named.

Default: dependent on other options

`timeline/timeline year` = `{<key-value list>}`
key

Adds `<key-value list>` to the common style used when putting major years and minor years onto the timeline. Do not specify `font` or `anchor` here as they will be overridden. Although both major and minor years use the same general style, they may and, by default do, use different fonts and date format keys.

Example: `timeline/timeline year=fill=chronos timeline background colour`

Default: `text=<timeline foreground>`, `text opacity=1`, `align=center`, `fill opacity=.75` (off line)

Default: `text=<timeline foreground>`, `anchor=center` (on line)

`timeline/timeline years` = `{<text>}`
anchor
key

The TikZ `anchor` to use when creating the nodes for years on or off the timeline. *Do not set this option unless you know you need to.* In most cases, `chronos` will pick a sensible default. The key is provided primarily for cases where you want to rotate the year labels in styles which place them off the line. Even then, you should not need to change the setting if using a style designed for rotation, unless you need to change the angle.

Default: dependent on other options

`timeline/timeline marks` = `true|false`
boolean key

Whether to draw vertical marks on or off the timeline at major years using the style set with `timeline mark`.

Default: `true`

Initially: `true`

`timeline/timeline minor` = `true|false`
marks
boolean key

²⁵So the non-existent year zero is marked at the same point as the existent year one. This avoids complications in `\foreach` loops.

Whether to draw vertical marks on or off the timeline at minor years using the style set with `timeline minor mark`.

Default: `true`

Initially: `true`

`timeline/timeline show` = `true|false`

`years`
boolean key

Whether to represent years on or off the timeline at all. If false, neither labels nor marks will be added when the timeline is constructed. This is useful if you wish to use a style such as `event years on line`, but is the nuclear option otherwise.

Default: `true`

Initially: `true`

`timeline/timeline bare` = `true|false`

`marks`
boolean key

Whether to draw bare marks on or off the timeline in between years²⁶ using the style set by `timeline bare mark`. If you specify `step divisions`, this key will be automatically enabled. If you don't want bare marks, don't set/set to zero `step divisions`.

Default: `true`

Initially: `false`

`timeline/timeline mark` = `{(key-value list)}`

key

Adds to the style used for the vertical lines drawn when `chronos` labels a major year on or off the timeline and `timeline marks` is true. These correspond to the major steps at which `chronos` puts years.

Example: `timeline mark=thick`

Default: `draw=(timeline foreground), Triangle[width=Opt 3,reversed,length=Opt 1.5]-, thin, shorten >=-2.5pt (off line)`

Default: `draw=(timeline foreground) (on line)`

`timeline/timeline minor` = `{(key-value list)}`

`mark`
key

Adds `(key-value list)` to the style used for the vertical lines drawn when `chronos` labels a minor year on or off the timeline and `timeline minor marks` is true. These correspond to the minor steps at which `chronos` puts years.

Example: `timeline mark=thin, shorten >=-2pt`

Default: `draw=(timeline foreground), Triangle[width=Opt 3,reversed,length=Opt 1.5]-, very thin, shorten >=-2.5pt (off line)`

Default: `draw=(timeline foreground), thin (on line)`

`timeline/timeline bare mark` = `{(key-value list)}`

key

Adds `(key-value list)` to the style used to draw lines at `step divisions`, provided `timeline marks` is true.

Example: `timeline bare mark=thin, <-`

Default: `draw=(timeline foreground), Triangle[width=Opt 3,reversed,length=Opt 1.5]-, very thin, shorten >=-1.5pt (off line)`

Default: `draw=(timeline foreground), thick (on line)`

`timeline/timeline all marks` = `{(key-value list)}`

key

²⁶If your timeline is very short and 12 `step divisions` are set, `chronos` will actually mark months. In other cases, marks simply divide the available space and are not placed by date.

Adds to the styles used to draw lines at major years, minor years and step divisions. This is equivalent to passing \langle key-value list \rangle to each of `timeline mark`, `timeline minor mark` and `timeline bare mark`.

- `event year on line` *style* The style used to mark years on the timeline if `event years on line` is enabled. By default, the style otherwise used for years when on the line is used. Redefine this if you wish, but you could also use `timeline years`, since no other years will be set on the line anyway.
- `event year on line skip` *key* Don't put this particular event's year on the timeline. This can be used if the line would otherwise become too crowded. See section 9.3.
- `timeline/era switch off` *line style* The style to use if years are 'off line' and `mark at era switch` is true. With the standard settings, you would get a small mark at the switch, no different from other intermediate marks. Likely you want something more similar in stature to the year labels. Redefine or supplement using standard TikZ techniques.

Default: `thick, shorten >=0pt`

```
\begin{chronos}
[
  timeline={%
    era switch off line/.append style={ultra thick},% retain undoing of shortening
in default, but make mark thicker
    era switch off line/.style={ultra thick, shorten>=-2pt},% make mark thicker and
longer
    era switch off line/.style={shorten>=-2pt},% make mark longer but use whatever
thickness is used for other marks
  },
]
\end{chronos}
```

8.4.4 Timeline Fonts

- `major step font` = \langle key-value list \rangle *key*
The font used for major years.

Default:

```
\begin{chronos}
[
  timeline={%
    major step font=\sffamily,
  },
]
\end{chronos}
```

- `timeline/minor step font` = \langle key-value list \rangle *key*
The font used for minor years.

Default:

```
\begin{chronos}
[
  timeline={%
    minor step font=\sffamily\small,
  },
]
\end{chronos}
```

- `timeline/eras font` = \langle key-value list \rangle *key*

The font used for era labels on the timeline.

Default:

```
\begin{chronos}
[
  timeline={%
    eras font=\sffamily\bfseries\large,
  },
]
\end{chronos}
```

8.4.5 Timeline Colours

`timeline/timeline border` = *<colour name>*

`inner colour`

`timeline/timeline border`

`inner color`

colour key

The innermost colour used for the gradient used to shade the timeline borders, if any. This colour is accessible within the `chronos` environment as `chronos timeline border inner colour` or `chronos timeline border inner color`.

Default: the `timeline background colour`, which is itself `black` by default.

```
\begin{chronos}
[
  timeline={%
    timeline border inner colour=blue,
  },
]
\end{chronos}
```

`timeline/timeline border` = *<colour name>*

`outer colour`

`timeline/timeline border`

`outer color`

colour key

The outermost colour used for the gradient used to shade the timeline borders, if any. This colour is accessible within the `chronos` environment as `chronos timeline border outer colour` or `chronos timeline border outer color`.

Default: the `background colour`, which is itself `white` by default.

```
\begin{chronos}
[
  timeline={%
    timeline border outer colour=green!5!white,
  },
]
\end{chronos}
```

`timeline/timeline border` = *<colour name>*

`middle colour`

`timeline/timeline border`

`middle color`

colour key

The middle colour used for the gradient used to shade the `idx post=colour configuration[type=element,idx as=timeline border]timeline` borders, if any. This colour is accessible within the `chronos` environment as `chronos timeline border middle colour` or `chronos timeline border middle color`.

Default: a 50-50 mix of the `timeline border outer colour` and `timeline border inner colour`.

```
\begin{chronos}
[
  timeline={%
    timeline border middle colour=blue!20!green,
  },
]
\end{chronos}
```



Figure 22: Cumulative effect of colour settings given as examples in sections 8.4.5 and 8.8.

`timeline/timeline` = \langle colour name \rangle

`background`
colour key

The colour used for the background of the central part of the timeline. This colour is accessible within the `chronos` environment as `chronos timeline background colour` or `chronos timeline background color`.

Default: the foreground colour, which is itself `black` by default (if putting years/marks on the line).

Default: the background colour, which is itself `white` by default (otherwise).

```
\begin{chronos}
[
  timeline={%
    timeline background=blue,
  },
]
\end{chronos}
```

`timeline/timeline` = \langle colour name \rangle

`foreground`
colour key

The colour used for the foreground of the central part of the timeline. This colour is accessible within the `chronos` environment as `chronos timeline foreground colour` or `chronos timeline foreground color`.

Default: the background colour, which is itself `white` by default (if putting years/marks on the line).

Default: the foreground colour, which is itself `black` by default (otherwise).

```
\begin{chronos}
[
  timeline={%
    timeline foreground=green!5!white,
  },
]
\end{chronos}
```

The cumulative effect of the colour settings given in the examples in this section, together with the `background` and `foreground` from section 8.8 is shown in fig. 22.

8.4.6 Timeline Style

The timeline's overall style can be customised using the following keys, which should (and, by default, do) utilise colours from the colour scheme (see section 13.2). Unless you are creating a `chronos` style, it is best to *add to* rather than *replacing* the existing configuration. For example, if you wish the line to take the form of an arrow, you can simply add the use of an appropriate arrow tip, without modifying the colours, dimensions or markings.

`timeline/timeline line` = $\{(key-value list)\}$

`timeline/timeline line'`
`timeline/timeline line+`
key

The style of the timeline line. `timeline/timeline line+` adds to the current list; `timeline/timeline line'` and `timeline/timeline line+` replace it.

Default: `empty`

Initially: dependent on other options

This key makes it possible to override the default drawing or filling of the timeline lines.

For example, `blues` below includes the following in its timeline configuration,

```

timeline={%
  ...
  timeline line={Bar-Latex,chronos timeline foreground colour,double=chronos timeline
background colour,line width=\timelineht/3,double distance=\timelineht/3,shorten <=-\
timelineht/3,shorten >=-3pt-2.1\timelineht},
  timeline config+={\pgfqkeys{/chronos/timeline}{timeline width-={3pt+2.43\timelineht}}},
  ...
}

```

To make the timeline line into an arrow, without otherwise modifying the existing style, use, for example,

```

timeline={%
  ...
  timeline line+={shorten >={-10mm}, -{Triangle Cap[length=10mm]}},
  timeline config+={\pgfqkeys{/chronos/timeline}{timeline width-=10mm}},
  ...
}

```

The adjustments are required to ensure that the tapered part is not counted when `chronos` calculates how much of the total `timeline width` is available to represent time.

`timeline/timeline arrow` = true|false
boolean key

Whether the timeline should be or have an arrow or arrows.

Default: true

Initially: false

Whether this has any effect depends entirely on the `chronos` style. With the default settings, it does nothing but trigger a warning, since `on line` styles cannot have arrows.

`timeline/no timeline arrow` A convenience key which sets `timeline/timeline arrow` false. *Whether this has any effect depends entirely on the `chronos` style.*
key

`timeline/timeline border` = {(key-value list)}

The style of the timeline border. `timeline/timeline border+` adds to the current list; `timeline/timeline border` and `timeline/timeline border'` replace it.
key

Default: empty

Initially: dependent on other options

This key makes it possible to override the default gradients used to fill the borders.

8.5 Frame

`frame` = true|false
boolean key

Whether to draw a frame. This is initially false, but use of `main/frame` will automatically set it to true.

Default: true

Initially: false

`frame uses bb` = true|false
boolean key

Whether the bounding box should be used to determine any frame at the end of the `chronos` environment. This is true by default and almost certainly what you want unless you are smuggling code into the end of the environment or using the frame for nefarious purposes.

Default: `true`

Initially: `true`

`main/frame` = $\{(key\text{-value list})\}$

`main/frame'` The style of the TikZ node used to draw the frame. This may be freely redefined as desired.

`main/frame+`
key Default: empty

Example: `main/frame={draw=black,ultra thick,inner sep=5pt}`

Example: `main/frame+={double=blue}`

The second form may be useful if you wish to modify, rather than replace, a style defined by a `chronos` style. `main/frame` and `main/frame'` replace any current list; `main/frame+` adds to it.

8.6 Placing Things: Levels & Coordinates

Knowing where to put things may get tricky in complicated or densely-packed timelines. `Chronos` offers several techniques to help. The simplest is to simply use existing items as reference points. `Chronos` names coordinates and nodes routinely and predictably, as explained throughout this documentation. However, sometimes this isn't quite enough. Levels and `chronos` coordinates offer additional help with vertical and horizontal placement respectively.

8.6.1 Levels

Levels are not (generally) visible elements. They are instead part of the structure behind-the-scenes. They are, if you like, minimal stage-hands.

The idea is to tell `chronos` how many tiers (approximately) of elements you will create above and below the timeline. For each of these levels, `chronos` creates a standardised node or placeholder based on the settings used for elements of type `life` when the timeline is constructed. Each of these nodes is named: `level 1`, `level 2`, ... above the timeline and `level -1`, `level -2`, ... below²⁷. The first node in each direction is shifted `2pt` from the timeline. Subsequent nodes are created directly above each other, with no separation between.

Together with points on the timeline, you then have a crude system for placing things horizontally and vertically. It also enables you to 'stack' text tags, but create them in any order.

`levels` = $\{(number\ above)\}:\{(number\ below)\}$

key $\langle number\ above \rangle$ and $\langle number\ below \rangle$ should be non-negative integers specifying how many levels to create above and below the timeline respectively.

Default:

no number of `levels` are created by default (not even zero).

```
\begin{chronos}
[
  levels=4:4,
]
\end{chronos}
```

`levels at` = $\{(coordinate)\}$

key

²⁷You can also refer to the nodes above as `u1`, `u2` etc. and those below as `i1`, `i2` etc.

Although they are not intended to be visible in the timeline, placeholder nodes may be rendered visible for debugging or development purposes. As such, it may be useful to move them from their default location.

Default: `chronos mid`

```
\begin{chronos}
[
  levels at=chronos year -200,% make sure this exists!
]
\end{chronos}
```

To render the nodes temporarily visible, see section 14.

8.6.2 Chronos Coordinates

In addition to the coordinates and nodes shown in fig. 3, `chronos` names a coordinate or node `chronos year <year>` for each year represented on the timeline. However, depending on your preferred style, this may not provide sufficient horizontal reference points. In that case, you can create additional coordinates. Like `levels`, `chronos` coordinates are not ordinarily visible; unlike `levels`, there is nothing there to see²⁸.

`chronos coords` = `{<comma-separated list of years>}`
comma-separated list key

For each `<year>` in `<comma-separated list of years>`, `chronos` will place a single coordinate named `chronos year <year>` at the appropriate point on the timeline. These may be used together with `levels` to specify coordinates e.g. `(chronos year <year> |- level <n>)` is the point vertically aligned with `level <n>` and horizontally aligned with `chronos year <year>`.

Default: `empty`

8.6.3 Miscellaneous

`\chronosbaselineskip` The `chronos` environment sets this macro equal to the current `\baselineskip`. It may be used to fine-tune placement in the same way you might use `\baselineskip` outside a `tikzpicture`.
macro

8.7 Headings

`headings` = `{<text>/<coordinate 1>/<coordinate 2>,<text>/<coordinate 1>/<coordinate 2>,...}`
`headings+`
`headings'`
comma-separated list key

List of value triplets in the format used by PGF's `\foreach`. The list should consist of one or more triplets where `<text>` is used in capitalised form for the content of a node which will be aligned with `chronos main headings` vertically and placed midway between the horizontal positions of `<coordinate 1>` and `<coordinate 2>`. `headings` and `headings+` add to the current list; `headings'` replaces it.

Default: `none`

See section 8.7.1 for an example.

`heading` = `{<text>}{<coordinate 1>}{<coordinate 2>}`
`heading+`
`heading'`
key

Add or set a single heading. These forms require the same information as `headings`, `headings+` and `headings'` but as three separate arguments.

Default: `none`

See section 8.7.1 for an example.

`subheadings` = `{<text>/<coordinate 1>/<coordinate 2>/<coordinate 3>,<text>/<coordinate 1>/<coordinate 2>/<coordinate 3>,...}`
`subheadings+`
`subheadings'`
comma-separated list key

²⁸You could label them, of course, but they are just regular PGF/TikZ coordinates and so naturally invisible.

List of value quadruplets in the format used by PGF's `\foreach`. The list should consist of one or more quadruplets where $\langle text \rangle$ is used in capitalised form for the content of a node which will be aligned with $\langle coordinate 4 \rangle$ vertically and placed midway between the horizontal positions of $\langle coordinate 1 \rangle$ and $\langle coordinate 2 \rangle$. $\langle coordinate 4 \rangle$ should be either `chronos upper subheadings` or `chronos lower subheadings`. `subheadings` and `subheadings+` add to the current list; `subheadings'` replaces it.

Default: none

See section 8.7.1 for an example.

```
subheading = { $\langle text \rangle$ }{ $\langle coordinate 1 \rangle$ }{ $\langle coordinate 2 \rangle$ }{ $\langle coordinate 3 \rangle$ }
subheading+
subheading'
key
```

Add or set a single subheading horizontally aligned with the midpoint between the horizontal positions of $\langle coordinate 1 \rangle$ and $\langle coordinate 2 \rangle$ and vertically aligned with $\langle coordinate 3 \rangle$. $\langle coordinate 3 \rangle$ should be either `chronos lower subheadings` or `chronos upper subheadings`, though this is not enforced. These forms require the same information as `subheadings`, `subheadings+` and `subheadings'` but as four separate arguments.

Default: none

See section 8.7.1 for an example.

```
century subheadings = { $\langle number list \rangle$ }{ $\langle text \rangle$ }
century subheadings+
century subheadings'
comma-separated list key
```

Create a subheading aligned with `chronos lower subheadings` for each of the centuries specified in $\langle number list \rangle$, using $\langle text \rangle$ as the superscript for each. Note that for the n th century `chronos year` coordinates much exist for both the year $n00$ and the year $(n+1)00$. `century subheadings` and `century subheadings+` add to the current list; `century subheadings'` replaces it.

Default: none

See section 8.7.1 for an example.

```
century subheading = { $\langle number \rangle$ }{ $\langle text \rangle$ }
century subheading+
century subheading'
key
```

Add or set a single century subheading. These forms require the same information as `century subheadings`, `century subheadings+` and `century subheadings'` but expect a single $\langle number \rangle$.

Default: none

See section 8.7.1 for an example.

8.7.1 Example

For example, here's an excerpt from the code used for fig. 2 which demonstrates the use of keys to create headings and subheadings.

```
\begin{chronos}
[
  timeline={%
    dates={-500}:1500,
  },
  chronos coords={-500,-450,...,1500},
  headings={heading/chronos year 800/chronos year 1500,another heading/chronos year
-450/chronos year 1,a third heading/chronos year 100/chronos year 800},
  subheadings={subheading on upper level/chronos year -250/chronos year 500/chronos
upper subheadings,subheading on lower level/chronos start/chronos year -100/chronos
lower subheadings,another subheading/chronos year 1000/chronos year 1500/chronos upper
subheadings,yet another subheading/chronos year 500/chronos year 1000/chronos lower
subheadings},
  century subheadings={12,13,...,15}{th},
  century subheading={1}{st},
]
```

```
\end{chronos}
```

Note the use of `chronos coords` to add coordinates for years which may not be visibly represented on the timelines. This ensures the `chronos year` coordinates needed to place headings, subheadings and century subheadings exist. It is permissible for coordinates to lie beyond the timeline's end date, though you may get strange results if you create coordinates too distant from the endpoint.

8.7.2 Headings Configuration

```
headings style = {(key-value list)}
```

```
headings style+
```

```
headings style'
key
```

PGF/TikZ options to apply to headings. `headings style` and `headings style'` replace the current list; `headings style+` replaces it.

Default: empty

Example: `headings style={align=center, anchor=base, inner sep=0pt, outer sep=0pt, color=chronos main colour, opacity=.8, font=\bfseries}`

Although the style is empty by default, `anchor=base` is passed to the node prior to the style. If you do not want this alignment, therefore, you must specify an alternative anchor.

```
subheadings style = {(key-value list)}
```

```
subheadings style+
```

```
subheadings style'
key
```

PGF/TikZ options to apply to subheadings. `subheadings style` and `subheadings style'` replace the current list; `subheadings style+` replaces it.

Default: empty

Example: `subheadings style={align=center, anchor=base, inner sep=0pt, outer sep=0pt, font=\bfseries\itshape\footnotesize, color=chronos main colour!75!chronos main background colour, opacity=.8}`

Although the style is empty by default, `anchor=base` is passed to the node prior to the style. If you do not want this alignment, therefore, you must specify an alternative anchor.

8.8 Colours

For timeline colours, see section 8.4.5. For basic colours, see section 8.3.

The *easiest* way to customise colours is to load a colour scheme as explained in section 7.2.

The *simplest* way to make use of colours is to specify colours for elements manually. Defaults can be configured in the timeline setup.

```
life/default colour = (colour name)
```

```
event/default colour
```

```
period/default colour
```

```
theory/default colour
```

```
info/default colour
```

```
life/default color
```

```
event/default color
```

```
period/default color
```

```
theory/default color
```

```
info/default color
```

```
colour key
```

Sets the default colour for elements of the specified type. This provides a fall-back colour and ensures some colour is always found, even when none is specified.

Default: `chronos main colour`

See foreground in section 8.4.5. For example,

```
\begin{chronos}
[
  life/default colour=chronos timeline foreground colour,
  event/default colour=chronos timeline foreground colour!50!chronos main colour,
  period/default colour=chronos main colour,
  theory/default colour=chronos timeline background colour,
  info/default colour=chronos main colour!50!chronos main background colour,
]
\end{chronos}
```

Alternatively or in addition, colours can be set on a per-element basis (sections 9.3 to 9.5).

8.8.1 Colour Rotation

More complex configuration can be achieved using lists of colours from which `chronos` selects when adding elements to the timeline. If you wanted to typeset all elements of type `life` in the colours of the rainbow taken in order, for example, it would be error prone and inflexible to assign colours manually. Instead, we would like `chronos` to select the colours in turn, keep track of which colour is used for which element and automatically adjust the assignments if items are inserted or removed from the timeline.

To achieve this, `chronos` supports colour rotation for text tags, connections and lines of type `life`, `event`, `period` and `theory`.

`Chronos` assigns all elements belonging to tags `life`, `event`, `period`, `theory` and `info` a colour with a predictable colour name. `Chronos` determines the colour to assign to the element as follows.

1. First, `chronos` checks whether a `colour` has been specified for the element.
 - ↳ If it has, that `colour` is assigned.
2. If not, `chronos` checks whether colour rotation is enabled for the relevant type of element.
 - ↳ If it is, `chronos` assigns the next colour from the specified colour list for the type of element in question and according to whether the element will be placed above or below the timeline. That colour is then moved to the bottom of the list.
3. If rotation is not enabled, a configurable `default` colour is assigned instead.

8 sets of colours can be configured which correspond to material placed above and below the timeline for each of `default`, `life`, `event` and `period`. See section 8.8.3 for details.

8.8.2 Using Colours

There are at least two things you might want `chronos` to tell you about elements' colours. First, you might want to know the `colour` assigned to a particular element *after* the element is created. Second, you might want to know the `colour` assigned to the current element during creation. Note 8.8.2.1 addresses the first, note 8.8.2.2 the second.

8.8.2.1 Colours by Element Name Regardless of how the colour assigned to an element ends up being determined, `chronos` assigns the colour a name derived from the element so that it can be used later, if required.

The result of this is that, assuming we have created an element of type `life` with `name=donald knuth`, we can write

```
\draw [chronos connect=life:donald knuth] (text tag connector donald knuth1) -- (text tag connector metafont2);
```

to connect Donald Knuth with an element named `metafont`, which might be of type `theory`. The code used to draw the connection will use the same style and colour as any connection drawn between Donald Knuth and the timeline²⁹. This colour can also be (and, by default, is) passed to the text tag. For example, a darker shade might be used for the text and outline of the node, and a paler one as a filling. The colour may also be accessed directly using `colour donald knuth`, `color donald knuth` or, if simple colour names are enabled³⁰, simply `donald knuth`.

`colour` *<name>* Colour names assigned to the element created with `name= <name>`. *life, event, period, theory, info*
`color` *<name>*
colour *<name>* Note these names cannot be used during the element's creation in `\chronos{tag}`.

²⁹See section 9.6

³⁰See sections 5 and 8.8.4.

`<name>` An additional name for colour `<name>`. *life, event, period, theory, info*
colour
 Requires simple colour names.

8.8.2.2 The Current Tag Colour You may also wish to refer to an element's assigned colour while creating it.

`chronos current tag colour` The colour assigned to the current element during creation. *life, event, period, theory, info*

`chronos current tag color`
colour This colour is available when creating an element belonging to an appropriate tag i.e. inside the tag context setup when using `\chronoslif`, `\chronosevent`, `\chronosperiod` or `\chronostheory`. Outside a tag context, `chronos current tag colour` and `chronos current tag color` are equivalent to `chronos main colour`.

Example: `\hypersetup{urlcolor=chronos current tag colour}`

Figure 1 uses this code within a `figure` to override the colour of URL links locally in such a way that each hyperlink's colour is the colour of the text tag to which it belongs.

8.8.3 Colour Lists

The lists of colours for colour rotation (section 8.8.1) may be loaded from provided styles, specified directly.

No specific lists are provided for *theory*, but you can obviously reserve the default lists for this type, if you want distinct lists for everything.

`colours above` = *<list of colour names>*

`colors above`
colour list key When given in the *<chronos preamble>* or to `\chronosset`, sets the default colour list for use above the timeline to *<list of colour names>*.

Default: Red,Orange,Yellow,Green,Blue,MidnightBlue,Violet

`colours below` = *<list of colour names>*

`colors below`
colour list key When given in the *<chronos preamble>* or to `\chronosset`, sets the default colour list for use below the timeline to *<list of colour names>*.

Default: Red,Orange,Yellow,Green,Blue,MidnightBlue,Violet

`colour rotation` = true|false

`color rotation`
boolean key When given in the *<chronos preamble>* or to `\chronosset`, determines whether colours are rotated by default or not.

Default: true

This key does not override tag-specific settings. Depending on other settings, therefore, using this key may have no effect or it may enable colour rotation for everything.

`rotate all colours` When given in the *<chronos preamble>* or to `\chronosset`, enables both default colour rotation and colour rotation for all supported tags. This key overrides tag-specific settings.

`rotate all colors`
key
`no colour rotation` When given in the *<chronos preamble>* or to `\chronosset`, disables default colour rotation. This key does not override tag-specific settings. Depending on other settings, therefore, using this key may have no effect or it may prevent colour rotation completely.

`no colour rotation`
key
`rotate no colours` When given in the *<chronos preamble>* or to `\chronosset`, disables both default colour rotation and colour rotation for all tags. This key overrides tag-specific settings.

`rotate no colors`
key
 Note that, like many `chronos` keys, the effect of setting these depends on the current key path. That means that using a key when creating a tag of type *life*, for example, the key will have a different effect from using in in the *<chronos preamble>*.

`life/colours above` = *<list of colour names>*

`life/colors above`
colour list key Sets the colour list for use with elements of type `life` placed above the timeline to *<list of colour names>*.

Default: `empty`

`life/colours below` = *<list of colour names>*

`life/colors below`
colour list key Sets the colour list for use with elements of type `life` placed below the timeline to *<list of colour names>*.

Default: `empty`

`event/colours above` = *<list of colour names>*

`event/colors above`
colour list key Sets the colour list for use with elements of type `event` placed above the timeline to *<list of colour names>*.

Default: `empty`

`event/colours below` = *<list of colour names>*

`event/colors below`
colour list key Sets the colour list for use with elements of type `event` placed below the timeline to *<list of colour names>*.

`period/colours above` = *<list of colour names>*

`period/colors above`
colour list key Sets the colour list for use with elements of type `period` placed above the timeline to *<list of colour names>*.

Default: `empty`

`period/colours below` = *<list of colour names>*

`period/colors below`
colour list key Sets the colour list for use with elements of type `period` placed below the timeline to *<list of colour names>*.

Default: `empty`

8.8.4 Simple Colour Names

If you wish to enable or disable `simple colour names` (see sections 5 and 8.8) for a particular timeline, use one of the following two options.

`simple colour names` = `true|false`

`simple color names`
boolean key Enable or disable `simple colour names`.

Default: `true`

Initially: `true`

Example: `simple colour names=false,`

See section 5 for details, but note that the keys here are implemented differently.

`no simple colour names` Disable `simple colour names`.

`no simple color names`
key Example: `no simple colour names,`

See section 5 for details, but note that the keys here are implemented differently. In particular, unlike both `simple colour names` and the load-time option, `no simple colour names` does *not* take an argument.

9 Adding Elements to the Timeline

See section 6.2 for an overview of the components available for use in the `timeline`'s *(`timeline additions specification`)*.

Seven macros are provided for adding elements to the `timeline`. Conceptually, these are always 'above' or 'below', though they could also be created to the left or right. For an overview of the way these commands work, see section 6.

9.1 Adding Connectable Elements

The most important kinds of additions `chronos` supports are those which can be connected to the `timeline` itself.

9.1.1 Timeline-Connectable Elements

`\chronoslife` *{(key-value list)}*
macro

life

Create an element of type `life`. The *(key-value list)* should specify values for `chronos` keys and may include arbitrary `TikZ` keys. At a minimum, `name` and `birth` must be specified for a living person. If the person is dead, both `birth` and `death` or `dates` should be given. If no date of death is specified, `chronos` assumes the person is living and uses the current date when placing the element on the `timeline`.

Table 5 summarises the `chronos` keys supported by elements of type `life`, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing, naming and connecting several components. These are described in table 6 for a typical case, but note that additional connectors require `connectors` to be set, the connection is drawn only if `connect` is `true` and some components may be rendered invisibly.

For example,

```
\chronoslife{%
  name=leslie lampport,
  birth={1941-02-07},
  at=leslie lampport |- u1.north,
  connectors=east,
  tag anchor=west,
  xshift=10pt,
}
```

This will create a text node (text tag) named `tag leslie lampport` with two connectors, `10pt` to the right of coordinate (`leslie lampport |- u1.north`), using the settings for `life`. The main connector, named `main connector leslie lampport` or `connector leslie lampport0`, will be at the `TikZ` anchor `west`. This will be used as the `TikZ` anchor when placing the node and used to connect it to the `timeline`. A second connector, named `connector leslie lampport1` will be created at the `east`, which may be used to connect the text tag to other elements.

A `chronos` connector, named `chronos connector leslie lampport` will be created on the `timeline` at the midpoint between `1941-02-07` and today's date. A line will also be marked on the `timeline` border, on the `timeline` or near the `timeline`, between these dates.

Note that the coordinate `leslie lampport` need not (and generally should not) exist when this command is given. A coordinate of this name will be created on the `timeline` midway between the birth and death dates (or, in this case, between the birth date and today's date) prior to creation of the text tag. However, `u1` must exist. In this case, it refers to a node created using the `levels` option. `u1` is also known as `level 1` and refers to the first level above the `timeline`. `Lampport` will be a bit higher because the text tag's `west` anchor will be aligned with the north of node `level 1`.

Table 5: Keys which are enabled (✓) and disabled/ineffective (-) for tag contexts associated with chronos macros.

Option	life	event	period	theory	theory circle	info	main	copyright copyleft
primarily per item configuration	name	✓	✓	✓	✓	✓	✓	✓
	as is	✓	✓	✓	✓	-	-	-
	at	✓	✓	✓	✓	✓	✓	✓
	at aux	✓	-	✓	-	-	-	-
	tag anchor	✓	✓	✓	✓	-	✓	✓
	colour color	✓	✓	✓	✓	-	✓	-
	connect	✓	✓	✓	-	-	-	-
	connectors connectors+ connectors'	✓	✓	✓	✓	-	-	-
	place above	✓	✓	✓	✓	-	-	-
	place below	✓	✓	✓	✓	-	-	-
	dates	✓	-	✓	-	-	-	-
	date	-	✓	-	-	-	-	-
	birth	✓	-	-	-	-	-	-
	death	✓	-	-	-	-	-	-
	start	-	-	✓	-	-	-	-
	end	-	-	✓	-	-	-	-
	dates content	✓	✓	✓	-	-	-	-
	name content	✓	✓	✓	✓	-	✓	✓
	text content	✓	✓	✓	✓	-	✓	-
	event year on line skip	-	✓	-	-	-	-	-
	caption	-	-	-	-	-	✓	-
	labels	-	-	-	-	✓	-	-
	circle texts	-	-	-	-	✓	-	-
	sizes	-	-	-	-	✓	-	-
	author	-	-	-	-	-	-	-
	copyleft	-	-	-	-	-	-	-
	notice	-	-	-	-	-	-	-
	rotate	-	-	-	-	-	-	-
	year	-	-	-	-	-	-	-
primarily all-of-type-tag configuration	date format	-	✓	-	-	-	-	-
	date formats	✓	-	✓	-	-	-	-
	full dates	✓	✓	✓	-	-	-	-
	only years	✓	✓	✓	-	-	-	-
	show eras	✓	✓	✓	-	-	-	-
	without eras	✓	✓	✓	-	-	-	-
	only text	✓	✓	✓	-	-	-	-
	tag tag+	✓	✓	✓	✓	-	✓	-
	connection connection+	✓	✓	✓	✓	-	-	-
	line line+	✓	-	✓	-	-	-	-
	line add yshift	✓	-	✓	-	-	-	-
	text tag text tag+	✓	✓	✓	✓	-	✓	-
	default colour color	✓	✓	✓	✓	-	✓	-
	colours colors above	✓	✓	✓	✓	-	-	-
	colours colors below	✓	✓	✓	✓	-	-	-
colour color rotation	✓	✓	✓	✓	-	-	-	
text tag yshift	✓	✓	✓	✓	-	-	-	

Table 6: Components of elements of tag types life and period.

Element	Name	Description	TikZ Type
–	$\langle name \rangle$	Point on timeline midway between $\langle birth \rangle$ and $\langle death \rangle$ (life) or $\langle start \rangle$ and $\langle end \rangle$ (period).	coordinate
line	–	Line or rectangle on or near timeline or timeline border from $\langle birth \rangle$ to $\langle death \rangle$ (life) or $\langle start \rangle$ to $\langle end \rangle$ (period).	$\backslash path$
chronos connector text tag	chronos connector $\langle name \rangle$ tag $\langle name \rangle$	Connection point midway along line. Main box representing element. By default, contains dates above capitalised $\langle name \rangle$ (life) or capitalised $\langle name \rangle$ above dates (period).	node node
main connector connection	main connector $\langle name \rangle$ –	Connection point at TikZ anchor of text tag. Line between the chronos connector and main connector.	node $\backslash draw$
connectors	connector $\langle name \rangle n$	Secondary connection point(s) at TikZ anchor(s) of text tag, named in order with $n = 1, 2, \dots$	node

Since the `text tag` is shifted right, the connection will be drawn using `|–` rather than `--`. If more complex paths are required, `connect=false` may be used and the `text tag` connected to the timeline manually. A `chronos connector`, `chronos connector leslie lampport`, would then be created on the timeline, as would the `connectors` on the text tag, but the connection itself would be omitted.

In addition, a colour named `colour leslie lampport` or `color leslie lampport` will be created. This is typically used in the styles responsible for the appearance of the `text tag`, `line`, `connection` and `connectors` and may be referenced and reused later. If simple colour names or simple color names are used, it may also be referenced as `leslie lampport`.

`\chronosevent` $\{ \langle key-value list \rangle \}$
macro

event

Create an element of type `event`. This is intended for events spanning no more than a day. The $\langle key-value list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `name` and `date` should be specified.

Table 5 summarises the `chronos` keys supported by elements of type `event`, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing, naming and connecting several components. These are described in table 7 for a typical case, but note that additional `connectors` require `connectors` to be set, the connection is drawn only if `connect` is `true` and some components may be rendered invisibly.

For example,

```
\chronosevent {%
  name=\emph{Common Sense},
  as is,
  yshift=5pt,
  date=1776,
  text=WildStrawberry,% will affect text for the element itself but not drawing,
  filling or the assigned colour
  place below,% does nothing because the positive yshift pushes the element above the
  timeline
}%
```

Note the use of `as is` to prevent errors trying to capitalise `\emph`. `place below` has no effect here: the item still ends up above the timeline due to `yshift=5pt`. Note the use of only a year in

Table 7: Components of an element of tag type event.

Element	Name	Description	TikZ Type
–	$\langle name \rangle$	Point on timeline at $\langle date \rangle$.	coordinate
line	–	Line from timeline to the edge of timeline border at $\langle date \rangle$.	$\backslash path$
chronos connector	<code>chronos connector</code> $\langle name \rangle$	Connection point at end of line.	node
text tag	<code>tag</code> $\langle name \rangle$	Main box representing element. By default, contains the date above the capitalised $\langle name \rangle$.	node
main connector	<code>main connector</code> $\langle name \rangle$	Connection point at TikZ anchor of text tag.	node
connection	–	Line between the chronos connector and main connector.	$\backslash draw$
connectors	<code>connector</code> $\langle name \rangle n$	Secondary connection point(s) at TikZ anchor(s) of text tag, named in order with $n = 1, 2, \dots$	node

`date`. If you only specify years, you probably want to configure your `timeline` to avoid printing full dates or you will end up with everything happening on January 1st. See section 8.2.2.

`\chronosperiod` $\{ \langle key-value list \rangle \}$
macro

period

Create an element of type `period`. This is intended for extended events spanning more than one day. The $\langle key-value list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `name` and `start` must be specified for an ongoing period. If the extended event has ended, both `start` and `end` or `dates` should be given. If no end date is specified, `chronos` assumes the period is ongoing and uses the current date when placing the element on the timeline.

Table 5 summarises the `chronos` keys supported by elements of type `period`, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing, naming and connecting several components. These are described in table 6 for a typical case, but note that additional connectors require `connectors` to be set, the connection is drawn only if `connect` is `true` and some components may be rendered invisibly.

For example,

```
\chronosperiod {%
  dates={476-01-01}:{476-10-31},
  name=Fall of the\Roman Empire,
  colour=blue,
  line+={draw=gray},% draw ugly grey border around line
}
```

This will construct an element analogous to the one created for `Lamport`. Note that the names of nodes and coordinates will be based on `Fall of theRoman Empire` because `chronos` will remove the `\` and the capitalisation won't change. `colour` `Fall of theRoman Empire` will be `blue` and the line representing the period on the timeline will be drawn in `gray` but potentially filled in `blue`. This is because `line+` adds to any existing style rather than replacing it.

9.1.2 Adding Other Connectable Elements

Of the remaining elements, only those of type `theory` are connectable. While they cannot be connected to the timeline³¹, `chronos` can create connectors for them to enable easy connections to other elements.

`\chronostheory` $\{ \langle key-value list \rangle \}$
macro

theory

³¹At least, `chronos` won't connect them for you.

Table 8: Components of an element of tag type theory.

Element	Name	Description	TikZ Type
–	$\langle name \rangle$	Alias for <code>text tag</code> .	node
text tag	<code>tag</code> $\langle name \rangle$	Main box representing element. By default, contains the capitalised $\langle name \rangle$.	node
main connector	<code>main connector</code> $\langle name \rangle$	Connection point at TikZ anchor of text tag.	node
connectors	<code>connector</code> $\langle name \rangle n$	Secondary connection point(s) at TikZ anchor(s) of text tag, named in order with $n = 1, 2, \dots$	node

Table 9: Components of an element of tag type theory circle.

Element	Name	Description	TikZ Type
–	$\langle name \rangle$	A (rectangular!) box containing all other components.	node
–	<code>label above</code> $\langle name \rangle$	Label above the ring.	nodes
–	<code>label below</code> $\langle name \rangle$	Label below the ring.	nodes
–	$\langle name \rangle 1$	Centre of the ring.	coordinate

Create an element of type theory. The $\langle key-value list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `name` must be specified, but `at` is required for placement. If left unspecified, `chronos` will place the theory at `chronos origin` and issue a warning.

Table 5 summarises the `chronos` keys supported by elements of type theory, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing and naming components of up to two kinds. These are described in table 8 for a typical case, but note that a `connector` requires `tag anchor` or `connectors` to be set. Connectors may be rendered invisibly.

9.2 Adding Non-Connectable Elements

The remaining elements are non-connectable.

`\chronostheorycircle` $\{ \langle key-value list \rangle \}$ *theory circle*
macro

Create a theory circle. The $\langle key-value list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `name` must be specified, but `at` is required for placement.

Table 5 summarises the `chronos` keys supported by elements of type theory circle, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing and naming components of several kinds. Depending on the style, the element is intended to consist of a ring with text placed on the upper and lower semicircles and labels above and below. A symbol or picture can then be placed at the centre. The components are described in table 9 for a typical case, but note that these are style-dependant. In practice, this element could be used in other ways since it depends primarily on re-definable styles. However, in that case, there's no reason to avoid — and every reason to prefer — a new name.

For example,

```
\chronostheorycircle{
  name=gutenberg revolution,
  at=chronos end |- printing press.center,
  sizes=15pt:9pt,
  circle texts=Gutenberg:Revolution,
  labels=15\textsuperscript{th}c.\thinspace \celabel:21\textsuperscript{st}c.\thinspace \celabel,
```

Table 10: Components of an element of tag type info.

Element	Name	Description	TikZ Type
–	$\langle name \rangle$	Alias for text tag.	node
text tag	tag $\langle name \rangle$	Main box representing element. Empty by default.	node
caption	caption $\langle name \rangle$	By default, contains the capitalised $\langle name \rangle$.	node

Table 11: Components of an element of tag type main.

Element	Name	Description	TikZ Type
text tag	$\langle name \rangle$	By default, contains the capitalised $\langle name \rangle$.	node

```
}

```

`\chronosinfo` $\{(key\text{-}value\ list)\}$ *info*
macro

Create an element of type `info` i.e. an information box with a distinct caption. The $\langle key\text{-}value\ list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `name` and `at` must be specified.

Table 5 summarises the `chronos` keys supported by elements of type `info`, with detailed usage information provided in sections 9.3 and 9.5.

Creating the element involves constructing and naming two components. These are described in table 10 for a typical case.

For example,

```
\chronosinfo{%
  name=syllogism,
  at=chronos year 200 |- u4,
  text content={All men are\[-.25em]\hspace*{1.5em}mortal.\Socrates is a\[-.25em]
] \hspace*{1.5em}man.\$\therefore$ Socrates is\[-.25em]\hspace*{1.5em}mortal.},
  anchor=north,
  caption=A Syllogism,
}
```

Note the use of `caption` to override the default reuse of `name`. This allows the box to be captioned ‘A Syllogism’, while allowing references simply to `syllogism`.

`\chronosmaintitle` $\{(key\text{-}value\ list)\}$ *main*
macro

Create the main title. The $\langle key\text{-}value\ list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `name` and `at` must be specified.

Table 5 summarises the `chronos` keys supported by elements of type `main`, with detailed usage information provided in sections 9.3 and 9.5.

The result is simply a TikZ node, as described in table 11.

`\chronoscopyright` $\{(key\text{-}value\ list)\}$ *copyleft, copyright*
macro

Create a `copyleft` or `copyright` notice. The $\langle key\text{-}value\ list \rangle$ should specify values for `chronos` keys and may include arbitrary TikZ keys. At a minimum, `at` should be specified to avoid a warning.

Table 5 summarises the `chronos` keys supported by elements of type `copyleft` and `copyright`, with detailed usage information provided in sections 9.4 and 9.5.

The result is simply a TikZ node, as described in table 12.

`\chronoscopyleft` $\{(key\text{-}value\ list)\}$ *copyleft, copyright*
macro

Table 12: Components of an element of tag type `copyleft` and `copyright`.

Element	Name	Description	TikZ Type
text tag	<code><name></code>	By default, contains a standard copyright or copyleft notice utilising whatever details are provided or default values and dummy texts.	node

Create a copyleft notice. Sets `copyleft true` before passing `{<key-value list>}` to `\chronoscopyright`.

9.3 Additional Elements: Local Configuration

These keys are designed for use when creating specific elements. That is, they should be used in the argument of a `chronos` command such as `\chronoslife`, `\chronosevent`, `\chronosperiod`, `\chronostheory`, `\chronosinfo`, `\chronostheorycircle`, `\chronosmaintitle`, `\chronoscopyleft` or `\chronoscopyright`. If used globally (e.g. in `\chronosset` or the `<chronos preamble>`), they will determine defaults for all elements (belonging to the relevant tag). Where this makes sense, the possibility is noted below; where it is not noted, global usage is unsupported.

name = `<text>` *life, event, period, theory, info, theory circle, main, copyleft, copyright*
key

The base name of the element. Except for `\chronosmaintitle`, `\chronoscopyleft` and `\chronoscopyright`, **this key is required**.

Default: `main title` (main)

Default: `copyleft and copyright` (copyleft and copyright)

By default, `<text>` is used multiple times.

First, it is capitalised and used for (part of) the content created for the element added to the timeline. `as is` prevents capitalisation. In the case of `life`, `event` and `period`, it is used for the non-date part of the content. In the case of `theory` and `main`, it is used for the whole content of the title. In the case of `info`, it is used to create the caption. In the case of `copyleft` and `copyright`, it is used as the author's name if `author` is unset. It is not used to create content in the case of `theory circle`.

Second, it is processed to create multiple names for different parts of the element e.g. names for `connectorss`, `text tags` etc. Processing attempts to remove some things which would be problematic when used as part of the names for coordinates and nodes, but markup can still cause problems. In this case, use `name content` or `text content` for the marked-up version and give `<name>` a suitably simplified version.

as is = `true|false` *life, event, period, theory*
boolean key

Whether to skip capitalisation of `name` if using it in the textual content of the element. If true, the `name` will *not* be capitalised; if false, it will be. Capitalisation is never used when setting the names of coordinates, nodes etc.

Default: `false`

at = `<coordinate>` *life, event, period, theory, info, theory circle, main, copyleft, copyright*
key

Where to place the element. This key is mandatory for `theory circle`, `info`, `main`, `copyleft` and `copyright`.

For `life`, `event`, `period` and `theory`, the key is optional. By default, the text tag will be placed at `<name>`, which is a point on the timeline calculated according to date, offset vertically by either `yshift` or `text tag yshift`. Since `theory` text tags do not have dates, they are placed at the `(chronos origin)` and a warning is issued.

Example: `at=<name> |- level -2`

This will align $\langle name \rangle$ horizontally with its placement point on the timeline and vertically with `level -2`, assuming at least two levels exist below the timeline. See section 8.6.

`at aux` = $\langle text \rangle$ *life, period*
key

A wrapper around `at` which appends a space followed by the sanitised name of the current element to $\langle text \rangle$ before passing the result to the key.

At present, the only values of $\langle text \rangle$ which make sense are `birth` or `death` (for `life`) and `start` or `end` (for `period`). These options allow the `text` tag to be placed relative to the end points of the line, rather than at its centre.

This key is intended for use with `off line` styles, especially those utilising `line add yshift`.

`tag anchor` = $\langle node anchor \rangle$ *life, event, period, theory, info, main, copleft, copyright*
key

The PGF/TikZ anchor to use for the element's main connector. This is the point `chronos` uses to connect `life`, `event` and `period` text tags to the timeline. By default, this anchor is also used when placing the text tag. That is, `tag anchor` is used as the TikZ `anchor`. If you want different anchors to be used for the connection point and for placement, you can use both keys.

```
\chronoslife{%
  name=friedrich gottlob koenig,
  dates={1774-04-17}:{1833-01-17},
  at=friedrich gottlob koenig |- i1.north,
  tag anchor=east,
  anchor=north east,
  xshift=-5pt,
}
```

Default[for elements below the timeline]north Default[for elements above the timeline]south These defaults may be overridden on a per-tag basis by setting the key globally. For example,

```
\begin{chronos}[%
  life/tag anchor=50,
  event/tag anchor=north east,
  period/tag anchor=south,
]
\end{chronos}
```

`colour` = $\langle colour name \rangle$ *life, event, period, theory, info*
color
colour key

The colour to assign to the element. The effect depends on the type of element being created and other settings. To modify the default colours, see sections 8.8 and 9.5.

`connect` = `true|false` *life, event, period*
boolean key

Whether to connect the element to the timeline.

Default: `true`

`connectors` = $\langle list of node anchors \rangle$ *life, event, period, theory*
`connectors+`
`connectors'` *key*

Connection points to create on the element's text tag. Applies to `life`, `event`, `period` and `theory`. `connectors` and `connectors+` *add* to the existing list (if any). `connectors'` *replaces* it.

Default: empty

```
connectors={north,south,east,west},
connectors'={north},
connectors+={south},
connectors={east},
```

This code would result in connection points at the node's `north`, `south` and `east` anchors.

Note that one connection point is always created if the element is of a kind which could be connected to the timeline.

default colour Use the default colour assigned to elements of this tag type. *life, event, period, theory, info, main*
default color *key* This key does something quite different if used in a global configuration context. See section 9.5 and section 8.8 for details. For example,

```
\begin{chronos}
[
  life/colour rotation=true,
  life/default colour=gray,
]
\chronoslife{% use colour from life's colours above colour list
  name=chris,
  dates={1038-01-10}:{1066-11-19},
  at=u2 -| chris,
}
\chronoslife{% use gray
  name=sandy,
  dates={1345-11-23}:{1378-12-24},
  at=u3 -| sandy,
  default colour,
}
\chronoslife{% use blue
  name=alex,
  dates={1246-09-22}:{1295-02-07},
  at=u5 -| alex,
  colour=blue,
}
\chronoslife{% use colour from life's colours below colour list
  name=hilary,
  dates={1156-06-12}:{1201-04-01},
  at=i4 -| hilary,
}
\end{chronos}
```

Note the lack of an argument when used locally.

Note that there is no reason to use this key unless you wish to override colour rotation for a particular element. It suffices not to specify a colour.

place below = true|false *life, event, period, theory*
boolean key

By default, `chronos` alternates putting elements of a particular type above and below the timeline, but you may wish to put everything above or below, all elements of particular type above or below. Furthermore, you may wish to override the default for particular elements. Densely-packed timelines, especially, can require considerable intervention in order to make best use of the space while arranging things in a clear and (hopefully) visually appealing way.

```
\chronosevent {%
  name=red letter day,
  date=1750,
  place below=false,
}
```

Default: true

Initially: dependent on other options

place above A convenience key equivalent to `place below=false`. *life, event, period, theory*
key

Thus the previous code could be rewritten as

```
\chronosevent {%
  name=red letter day,
  date=1750,
  place above,
}
```

`line add yshift` = `{<dimension>}` life, period

`line add yshift+`

`line add yshift'` Additional vertical displacement of lines from the timeline. This is *added* to the default vertical displacement.

`line add yshift-`
dimension key

Cf. `line yshift`.

`\lineyshift` The `line yshift`. This macro is available *only within the <timeline specification>*.

macro

`dates` = `{<birth date>}:{<death date>}`

life

date key

=`{<start date>}:{<end date>}`

period

Dates of a life or period, specified as explained in section 8.2. The second date may be empty for a living person or ongoing occurrence. This key offers a more compact syntax as an alternative to the keys `birth` and `death` or `start` and `end` explained below. That is

```
dates={1310-02-03}:{1350-06-07},
```

is equivalent to

```
birth={1310-02-03},
death={1350-06-07},
```

for life or

```
start={1310-02-03},
end={1350-06-07},
```

for period.

By default, these dates are used for both placement on the timeline and the date content of the element's text tag, but see `dates content`.

`birth` = `{<birth date>}`

life

date key

The date of birth for a life, specified as explained in section 8.2. See `dates` above.

`death` = `{<death date>}`

life

date key

The date of death for a life, specified as explained in section 8.2. See `dates` above.

`start` = `{<start date>}`

period

date key

The start date of a period, specified as explained in section 8.2. See `dates` above.

`end` = `{<end date>}`

period

date key

The end date of a period, specified as explained in section 8.2. See `dates` above.

`date` = `{<date>}`

event

date key

The date of an event, specified as explained in section 8.2. By default, the date is used for both placement on the timeline and the date content of the element's text tag, but see `dates content`.

`event year on line skip` Don't put this particular event's year on the timeline. event

key

This can be used if the line would otherwise become too crowded when using `event years on line`. Cf. `special date`. See section 8.4.3. Figure 20 illustrates the effect of using this key.

special date = $\{(text)\}$ *event*
key
 Use $\langle text \rangle$ rather than the **date** for a particular event when using **event years on line**. Cf. **event year on line skip**. See section 8.4.3. Figure 20 illustrates the effect of using this key.

dates content = $\{(text)\}$ *life, event, period*
key
 Override the use of specified dates when creating content for the element's text tag. This is intended for 'special' cases e.g. uncertain, approximate or non-standardly specified dates. By default, the value is derived from **dates** or **date**.

Example: `dates content={c600-1450\, \celabel}`

name content = $\{(text)\}$ *life, event, period, theory, info, main*
key
 Override the use of the element's name when creating content for the element's text tag. This might be necessary if special markup is required. For example,

```
name content=\LaTeX3 Hummingbird,
```

It may also be desirable where longer content would render reuse of a **name** unwieldy.

text content = $\{(text)\}$ *life, event, period, theory, info*
key
 Override the use of both element's **name** and **dates** when creating content for the element's text tag.

```
name=block printing,  
text content={Block printing, originally used to print pictures and text onto cloth,  
developed into a method of printing books on paper.},
```

phantom = true|false *life, event, period*
boolean key
 Create a 'phantom' element. Phantoms have assigned colours, require **names** and potentially feature lines, but they do not have text tags or connections. Note that these components are not invisible; *they are not constructed at all*.

Default: true

Initially: false

Example: `\chronosperiod{\name=c17,dates=1600:1699,colour=cyan,phantom}`

This key may be used globally to set a different tag-specific default.

```
\begin{chronos}[%  
  period/phantom,% make periods are phantoms by default  
  event/phantom=true,% make events are phantoms by default  
  life/phantom=false,% make lives non-phantoms by default (this matches the package  
default)  
  ]  
\end{chronos}
```

For example, this key may be used to colour stretches of time without visibly labelling them, in conjunction with non-phantom lives or events³².

```
\begin{chronos}[% https://tex.stackexchange.com/a/701743/  
  ...  
  period={%  
    phantom,  
    colours below={orange,cyan,green,green},  
  },
```

³²Based on my answer at [TeX StackExchange: 701743](https://tex.stackexchange.com/a/701743/).

```

    ...
  ]
  % these **must** be named, even though they invisible, detached phantoms
  \chronosperiod{dates=2018:2019,name={n1}}
  \chronosperiod{dates=2019:2022,name={n2}}
  \chronosperiod{dates=2022:2023,name={n3}}
  \chronosperiod{dates=2023:2024,name={n4}}
  ...
\end{chronos}

```

`caption` = $\langle\text{text}\rangle$ info

key

The caption for an element of type info.

`labels` = $\langle\text{upper label}\rangle:\langle\text{lower label}\rangle$ theory circle

key

Labels to be placed above and below a theory circle.

`circle texts` = $\langle\text{upper text}\rangle:\langle\text{lower text}\rangle$ theory circle

key

The text to place in the upper and lower parts of a theory circle. By default, this uses `text effects along path`, so the content must be consistent with the restrictions imposed by use of this TikZ decoration.

`sizes` = $\langle\text{outer circle dimension}\rangle:\langle\text{inner circle dimension}\rangle$ theory circle

dimension key

The sizes of the inner and outer circles used to create a theory circle.

Default: 15pt:9pt

The difference between the two dimensions gives the thickness of the ring around which text is placed; the size of the inner circle gives the dimension of the hole in which a symbol or similar may be placed. This key may be used globally to set defaults.

```

\begin{chronos}[%
  theory/circles/sizes'+=10pt:5pt,
]
\end{chronos}

```

9.4 Additional Elements: Local/Global Configuration

Although you will generally want to use the following keys in the $\langle\text{chronos preamble}\rangle$ or in `\chronosset`, they can also be used to influence the format of a particular element.

$\langle\text{tag}\rangle/\text{date format}$ = $\langle\text{date format specification}\rangle$ event

date format key

Use $\langle\text{date format specification}\rangle$ to format date.

```

\chronosevent{%
  ...,
  date format={!a, !d !b},% show short day of week, day of month and short month
}
\end{chronos}

```

See section 8.2 for details and defaults.

$\langle\text{tag}\rangle/\text{date formats}$ = $\langle\text{date format spec.}\rangle:\langle\text{date format spec.}\rangle:\langle\text{date format spec.}\rangle$ life, period

date format key

Use $\langle\text{date format spec.}\rangle$ s to format date range.

```

\chronosevent{%
  ...,
  date formats={!d}:{!d !B},% show day of month for start/birth date and day of month
  and month name for end/death date
}

```

```
}
\end{chronos}
```

See section 8.2 for details and defaults.

full dates Show full dates. *life, event, period*
 <tag>/full dates

```
key \chronoslife{%
    ...,
    full dates,
}
\end{chronos}
```

See section 8.2 for details and defaults.

only years Show only years. *life, event, period*
 <tag>/only years

```
key \chronoslife{%
    ...,
    only years,% use only years in all dates
    event/full dates,% override to use full dates for events
}
\end{chronos}
```

See section 8.2 for details and defaults.

show eras Show eras. *life, event, period*
 <tag>/show eras

```
key \chronoslife{%
    ...,
    show eras,% show eras in all text tags
}
\end{chronos}
```

See section 8.2 for details and defaults.

without eras Omit eras. *life, event, period*
 <tag>/without eras

```
key \chronoslife{%
    ...,
    without eras,% omit eras in all text tags
    life/show eras,% override to show eras in life text tags
}
\end{chronos}
```

See section 8.2 for details and defaults.

only text Omit all date information. *life, event, period*
 <tag>/only text

key Default: disabled

```
\chronoslife{%
    ...,
    only text,% omit all dates from all tags
}
\end{chronos}
```

The following six sets of keys all work in the same way³³. If used when creating a specific element, they affect that element. If set in the $\langle\text{chronos preamble}\rangle$ or $\backslash\text{chronosset}$ with a `tag` prefix, they set the `tag`-specific setting and will affect all elements belonging to that tag unless overridden locally.

Note these keys require a `tag` prefix if used in a global context, such as the $\langle\text{chronos preamble}\rangle$. They do not need a prefix if used when creating a particular element. For example,

```
\begin{chronos}
  [
    event/line+={semithick},% prefix required ; event/ explicit
  ]
  \chronosevent{%
    name=dydd dewi sant,
    date={1982-03-01},
    line+={double},% no prefix ; event/ implicit
  }
\end{chronos}
```

$\langle\text{tag}\rangle/\text{connection}$ = $\{(key\text{-value list})\}$ life, event, period, theory

$\langle\text{tag}\rangle/\text{connection+}$ $\langle\text{tag}\rangle/\text{connection}'$ $\langle\text{tag}\rangle/\text{connection}$ $\langle\text{key-value list}\rangle$ to apply to this element's connection. This affects the line drawn between the element's connector on the timeline and the text tag's main connector. This is intended for arbitrary TikZ keys; it should *not* be used for chronos keys as they may not be processed correctly. $\langle\text{tag}\rangle/\text{connection}$ and $\langle\text{tag}\rangle/\text{connection}'$ replace any current list; $\langle\text{tag}\rangle/\text{connection+}$ adds to it.

$\langle\text{tag}\rangle/\text{line}$ = $\{(key\text{-value list})\}$ life, event, period

$\langle\text{tag}\rangle/\text{line+}$ $\langle\text{tag}\rangle/\text{line}'$ $\langle\text{tag}\rangle/\text{line}$ $\langle\text{key-value list}\rangle$ to apply to this element's line on or parallel to the timeline. This is the line representing the temporal extension of a life or period. This is intended for arbitrary TikZ keys; it should *not* be used for chronos keys as they may not be processed correctly. $\langle\text{tag}\rangle/\text{line}$ and $\langle\text{tag}\rangle/\text{line}'$ replace any current list; $\langle\text{tag}\rangle/\text{line+}$ adds to it.

Default: `fill=##1,fill opacity=.25,draw=none` (on line, life/period)

Default: `draw=##1,fill=none,opacity=.25` (on line, event)

Default: `draw=##1,thick,fill opacity=.75` (off line, life/period)

Default: `draw=##1,draw opacity=.75,fill=none` (off line, event)

`line yshift` = $\{(dimension)\}$ life, period

`line yshift+` `line yshift'` `line yshift-` Default vertical displacement of lines from the timeline. Whether the displacement is reckoned from the centre or border of the timeline depends on the default placement.

Cf. `line add yshift`.

`\lineyshift` The `line yshift`. This macro is available *only within the $\langle\text{timeline specification}\rangle$* .

$\langle\text{tag}\rangle/\text{text tag}$ = $\{(key\text{-value list})\}$ life, event, period, theory, info

$\langle\text{tag}\rangle/\text{text tag+}$ $\langle\text{tag}\rangle/\text{text tag}'$ $\langle\text{tag}\rangle/\text{text tag}$ $\langle\text{key-value list}\rangle$ to apply to this element's text tag. This is intended for arbitrary TikZ keys; it should *not* be used for chronos keys as they may not be processed correctly. $\langle\text{tag}\rangle/\text{text tag}$ and $\langle\text{tag}\rangle/\text{text tag}'$ replace any current list; $\langle\text{tag}\rangle/\text{text tag+}$ adds to it.

```
\chronosset{%
  life/text tag+={font=\scshape\small},
  event/text tag+={font=\scshape\footnotesize},
  period/text tag+={font=\itshape\footnotesize},
```

³³There is a seventh set, $\langle\text{tag}\rangle/\text{tag}$, $\langle\text{tag}\rangle/\text{tag+}$ and $\langle\text{tag}\rangle/\text{tag}'$, which may be of interest to advanced users. These keys are also potentially destructive. Not only $\langle\text{tag}\rangle/\text{tag}'$, but also $\langle\text{tag}\rangle/\text{tag}$ and even $\langle\text{tag}\rangle/\text{tag+}$, can overwrite default settings for such things as colour rotation.

}

See also `<tag>/date font` and `<tag>/text font`.

`<tag>/chronos connector` = `{(key-value list)}` *life, event, period*

`<tag>/chronos connector+`
`<tag>/chronos connector'`
key Specify TikZ settings to be used when creating `chronos` connectors on the timeline. Note that `<tag>/chronos connector` *adds* options to the current list. If, for some reason, you want to override this, you must do so explicitly. In general, it does *not* make sense to change this base option, so consider carefully whether you wish to do so.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

`<tag>/text tag connector` = `{(key-value list)}` *life, event, period, theory*

`<tag>/text tag connector+`
`<tag>/text tag connector'`
key Specify TikZ settings to be used when creating `text tag` connectors on the timeline. Note that `<tag>/text tag connector` *adds* options to the current list. If, for some reason, you want to override this, you must do so explicitly. In general, it does *not* make sense to change this base option, so consider carefully whether you wish to do so.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

`<tag>/main text tag connector` = `{(key-value list)}` *life, event, period, theory*

`<tag>/main text tag connector+`
`<tag>/main text tag connector'`
key Specify *additional* TikZ settings to be used when creating the main connectors on `text tags`. `<tag>/main text tag connector` and `<tag>/main text tag connector'` replace any current list; `<tag>/main text tag connector+` adds to it. The 'main' connector is the one which connects (or would connect) the `text tag` to the timeline. These keys are rarely needed because, usually, you want all the `text tag` connectors to look the same. Only use one of these three keys rather than one from the previous set if you *don't* want `<key-value list>` to apply to all of them. You do *not* need to duplicate settings here.

Note that `<tag>/main text tag connector` *adds* options to the current list. If, for some reason, you want to override this, you must do so explicitly. In general, it does *not* make sense to change this base option, so consider carefully whether you wish to do so.

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

`<tag>/label` = `{(key-value list)}` *info, theory circle*

`<tag>/label'`
`<tag>/label+`
key Style to apply to the caption of an element of `tag` type `info` or the labels of an element of type `theory circle`. In the latter case, the style applies to both the upper and lower label.

Default: empty

`label` and `label'` replace the current list; `label+` replaces it.

`<tag>/title` = `{(key-value list)}` *main*

`<tag>/title'`
`<tag>/title+`
key Style to apply to the main title, an element of `tag` type `main`.

Default: empty

`main/title` and `main/title'` replace the current list; `main/title+` replaces it.

`<tag>/title lines` Place main title between two parallel lines aligned to the width of the text. *main*
style

This style is available when creating a `text tag` of type `main` and draws lines along the northern and southern sides of the node. It is used in `somewhat plain` and `date centric`.

`<tag>/author` = `{(text)}` *copyleft, copyright*
key

The author's name for a `copyleft` or `copyright` notice. This is used only if `name content` is unset.

Default: `Author` (as a last resort)

If `author` and `name content` are unset, `chronos` first tries to figure out a suitable author. If `name` is set, a capitalised version is used. Otherwise, if `\svnauthor` is defined,

`\svnFullAuthor`{`\svnauthor`} is used, if `\svnFullAuthor` is available, or `\svnauthor`, if it is not. If `chronos` still hasn't found an author, `Author` is used.

`<tag>/copyleft` = `true|false` *copyleft, copyright*
boolean key

Whether a `copyleft` or `copyright` notice should specify `copyleft` or `copyright`.

Default: `false` (`\chronoscopyright`)

Default: `true` (`\chronoscopyleft`)

`\chronoscopyright` respects the global default, so if you set `<tag>/copyleft true` with `\chronosset`, both macros will make `copyleft` notices unless overridden in the *<key-value list>* of options they absorb when executed. `\chronoscopyleft` always creates a `copyleft` notice, regardless of any global settings, unless `copyleft` is explicitly set `false` when invoked.

`<tag>/notice` = `{<macro definition>}` *copyleft, copyright*
key

Template for a `copyleft` or `copyright` notice. It is used as the definition of the macro used for the content of the notice and should absorb two arguments: year and author.

Default: `{Copyleft \textcopyleft{} #1 #2}` (if `<tag>/copyleft` is true)

Default: `{Copyright \textcopyright{} #1 #2}` (if `<tag>/copyleft` is false)

For example,

```
\begin{chronos}
[
  copyright/notice={Created by #2 in the year #1 of the Great Debacle at the behest of
  His Gracious Grasp Full Acre Fanfare the Nineteenth.},
]
```

`<tag>/rotate` = `<angle>` *copyleft, copyright*
key

The angle to rotate the node containing a `copyleft` or `copyright` notice.

Default: 90

`<tag>/year` = `<text>` *copyleft, copyright*
key

The year of publication for a `copyleft` or `copyright` notice.

Default: `\svnyear` (if available)

Default: `\today` (otherwise)

9.4.1 Specialist Fonts for Text Tags

`<tag>/date font` = `{}` *life, event, period*
key

Set font macros to be applied to the date content of text tags.

Default:

```
\chronosset{%
...
event/date font=\itshape\bfseries\small,
life/date font=\sffamily\large,
period/date font=\upshape\normalsize\mdseries,
}
```

Note that if you want to alter the font for the entire contents of the `text tag`, it is better to just use `<tag>/text tag+=font={<>}`. Use `date font` to modify those settings specifically for date(s). Note that if era label are included, they will not be affected.

`<tag>/text font` = `{}` *life, event, period*
key

Set font macros to be applied to the text content of text tags.

Default:

```
\chronosset{%
  ...
  event/text font=\uishape\large,
  life/text font=\sffamily\Large,
  period/text font=\small\bfseries,
}
```

Note that if you want to alter the font for the entire contents of the text tag, it is better to just use `<tag>/text tag+=font={<>}`. Use `text font` to modify those settings specifically for names.

9.5 Additional Elements: Global Configuration

Except where otherwise noted, the keys in this section should not be used locally. The following keys are intended for use in the `<chronos preamble>` or in `\chronosset`. They are not intended for use when creating particular elements. For example, `default colour` should *not* be used for particular elements, unless you wish to *use* the existing default, as opposed to setting it. Instead, use `colour` to override default settings.

See section 8.8 for further information about colour keys and colour list keys.

`life` = `{<(key-value list)>}` *life, event, period, theory*
`event`
`period`
`theory`
`key`

Equivalent to prefixing each item in `<(key-value list)>` with `<tag>`.

```
\begin{chronos}
[
  life={%
    full dates,
    without eras,
    text tag+={font=\sffamily},
    text font=\bfseries,
    date font=\small,
    colours above={red,orange,blue},
    colours below={darkgray,gray,black,magenta},
  },
  period={%
    only years,
    text tag+={opacity=.75},
  },
  event={%
    text tag+={double=blue},
  },
]
\end{chronos}
```

`<tag>/default colour` = `<(colour name)>` *life, event, period, theory, info*
`<tag>/default color`
`colour key`

The default colour to use for all elements of type `<tag>`, as explained in section 8.8. *This key does something quite different if used when creating a specific element. See section 9.3 for details.* For example,

```
\begin{chronos}[
  life/default colour=blue,
  event/default colour=green,
  period/default colour=red,
]
\end{chronos}
```

See section 8.8 for details and defaults.

`colours above` = `{(colour list)}` *life, event, period, theory*
`colors above`
`<tag>/colours above` The default and tag-specific colour lists for all susceptible elements above the timeline. *These keys*
`<tag>/colors above` *should never be used when creating specific elements.*
colour list key

```
\begin{chronos}[
  colours above={gray,blue,green},
  life/colours above={magenta,pink,purple},
]
\end{chronos}
```

See section 8.8 for details and defaults.

`colours below` = `{(colour list)}` *life, event, period, theory*
`colors below`
`<tag>/colours below` The default and tag-specific colour lists for all susceptible elements below the timeline. *These keys*
`<tag>/colors below` *should never be used when creating specific elements.*
colour list key

```
\begin{chronos}[
  colours below={red,orange,magenta},
  theory/colours below={black,gray},
]
\end{chronos}
```

See section 8.8 for details and defaults.

`colour rotation` = `true|false` *life, event, period, theory*
`color rotation`
`<tag>/colour rotation` Whether colour rotation is enabled by default.
`<tag>/color rotation` Default: `true`
boolean key

```
\begin{chronos}[
  colour rotation=false,
]
\end{chronos}
```

See section 8.8 for details and defaults.

`copyleft` = `{(key-value list)}` *copyleft, copyright*
`copyleft'`
`copyleft+` Style to apply to the copyleft or copyright, an element of tag type copyleft / copyright.
`copyright` Default: empty
`copyright'`
`copyright+` `copyleft`, `copyleft'`, `copyright` and `copyright'` replace the current list; `copyleft+` and
key `copyright+` replace it.

`event dates split` = `true|false` *event*
boolean key

Create two text tags for each event, one above and one below the timeline. The formatted `date` or `dates` content goes into one and the formatted `name` or `name` content goes into the other. *This key has no effect on text tags belonging to other tags, such as life or period.*

Default: `true`

Initially: `false`

`event date split` Additional style applied to text tags of type event if `event dates split` is `true`. *event*
style

This style is provided primarily for use *outside* the `chronos` environment, in case you want some timelines with split events and some without. It is *not* intended to support both split and unsplit events on the same timeline.

Default: empty

The next twelve sets of keys fall into two groups, corresponding to the five sets of corresponding keys explained in section 9.4. *None of these keys should be used when creating specific elements.*

The first set of six consists of plural forms, as opposed to the singular forms used for tag-specific configuration. These are available in the `<chronos preamble>` and `\chronoset`.

`text tags = {(key-value list)}` *life, event, period, theory, info*
`text tags+`
`text tags'`
key Set or modify the global default `<key-value list>` to be applied to text tags in the absence of a tag-specific setting (section 9.4). `text tags` and `text tags'` replace the current value; `text tags+` replaces it.

Default: `outer sep=0pt, text=#1!75!black`

The key are passed a single argument specifying the current element's assigned colour, which may be used in the usual way i.e. by writing `#1` everywhere you would like the colour to be used.

Note that, when checking if a more fine-grained value is set, *the lists of <key-value> pairs are regarded as a whole. They are not treated on a <key>-by-<key> basis.* So if you write

```
\begin{chronos}
[
  event/text tag={},
  text tags+={fill=green},
]
\end{chronos}
```

you will *not* get green text tags for events. Nor will you get the package option default. Instead, no style whatsoever will be applied when creating event text tags.

`connections = {(key-value list)}` *life, event, period, theory*
`connections+`
`connections'`
key Set or modify the global default `<key-value list>` to be applied to connections in the absence of a tag-specific setting (section 9.4). `connections` and `connections'` replace the current value; `connections+` replaces it.

Default: `draw=#1`

These keys are related to the tag-specific `<tag>/connection`, `<tag>/connection+` and `<tag>/connection'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to `<tag>/text tag`, `<tag>/text tag+` and `<tag>/text tag'`. Please see above for details.

`lines = {(key-value list)}` *life, event, period*
`lines+`
`lines'`
key Set or modify the global default `<key-value list>` to be applied to lines in the absence of a tag-specific setting (section 9.4). `lines` and `lines'` replace the current value; `lines+` replaces it.

Default: none (see section 9.4 for tag-specific defaults.)

These keys are related to the tag-specific `<tag>/line`, `<tag>/line+` and `<tag>/line'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to `<tag>/text tag`, `<tag>/text tag+` and `<tag>/text tag'`. Please see above for details.

`chronos connectors = {(key-value list)}` *life, event, period, theory*
`chronos connectors+`
`chronos connectors'`
key Set or modify the global default `<key-value list>` to be applied to chronos connectors in the absence of a tag-specific setting (section 9.4). `chronos connectors'` replaces the current value; `chronos connectors` and `chronos connectors+` replace it.

Default: `anchor=center, inner sep=0pt, outer sep=0pt`

These keys are related to the tag-specific `<tag>/chronos connector`, `<tag>/chronos connector+` and `<tag>/chronos connector'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to `<tag>/text tag`, `<tag>/text tag+` and `<tag>/text tag'`. Please see above for details.

`text tag connectors` = `{(key-value list)}` *life, event, period, theory*

`text tag connectors+`
`text tag connectors'` Set or modify the global default `<key-value list>` to be applied to text tag connectors in the absence of a tag-specific setting (section 9.4). `text tag connectors'` replaces the current value; `text tag connectors` and `text tag connectors+` replace it.

key

Default: `anchor=center,inner sep=0pt,outer sep=0pt`

These keys are related to the tag-specific `<tag>/text tag connector`, `<tag>/text tag connector+` and `<tag>/text tag connector'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to `<tag>/text tag`, `<tag>/text tag+` and `<tag>/text tag'`. Please see above for details.

`main text tag connectors` = `{(key-value list)}` *life, event, period, theory*

`main text tag connectors+`
`main text tag connectors'` Set or modify the global default `<key-value list>` to be applied to main text tag connectors in the absence of a tag-specific setting (section 9.4). `main text tag connectors'` replaces the current value; `main text tag connectors` and `main text tag connectors+` add to it.

key

Default: empty

These keys are related to the tag-specific `<tag>/main text tag connector`, `<tag>/main text tag connector+` and `<tag>/main text tag connector'` in just the same way as `text tags`, `text tags+` and `text tags'` are related to `<tag>/text tag`, `<tag>/text tag+` and `<tag>/text tag'`. Please see above for details.

The next six sets of keys are convenience keys which set or modify the global defaults and the corresponding keys for all tags at once.

`every text tags` = `{(key-value list)}` *life, event, period, theory, info*

`every text tags+`
`every text tags'` A convenience key equivalent to setting the same `<key-value list>` for all of `text tags`, `life/text tag`, `event/text tag`, `period/text tag`, `theory/text tag` and `info/text tag` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

key

`every connections` = `{(key-value list)}` *life, event, period, theory*

`every connections+`
`every connections'` A convenience key equivalent to setting the same `<key-value list>` for all of `connections`, `life/connection`, `event/connection`, `period/connection` and `theory/connection` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

key

`every lines` = `{(key-value list)}` *life, event, period*

`every lines+`
`every lines'` A convenience key equivalent to setting the same `<key-value list>` for all of `lines`, `life/line`, `event/line` and `period/line` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

key

`every chronos connectors` = `{(key-value list)}` *life, event, period, theory*

`every chronos connectors+`
`every chronos connectors'` A convenience key equivalent to setting `<key-value list>` for all of `chronos connectors`, `life/chronos connector`, `event/chronos connector`, `period/chronos connector` and `theory/chronos connector` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

key

`every text tag connectors` = `{(key-value list)}` *life, event, period, theory*

`every text tag connectors+`
`every text tag connectors'` A convenience key equivalent to setting the same `<key-value list>` for all of `text tag connectors`, `life/text tag connector`, `event/text tag connector`, `period/text tag connector` and `theory/text tag connector` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

key

`every main text tag connectors` = `{(key-value list)}` *life, event, period, theory*

`every main text tag connectors+`

`every main text tag connectors'`

`every main text tag connectors'`

key

A convenience key equivalent to setting the same \langle key-value list \rangle for all of `main text tag connectors`, `life/main text tag connector`, `event/main text tag connector`, `period/main text tag connector` and `theory/main text tag connector` or the `+` or `'` variants. *This key should never be used when creating a specific element.* See section 9.4 and above for details and defaults.

`every theory circle circle` = \langle key-value list \rangle *theory circle*
`every theory circle circle'`
`every theory circle circle+` Configuration of the base ring for elements of tag type `theory circle`. The ring consists of two circles with the smaller forming a hole in the centre by default. Changing or deleting the filling rule will eliminate the hole.

Default: `fill= \langle chronos main colour \rangle` , `draw= \langle chronos main colour \rangle` , even odd rule

`every theory circle circle` and `every theory circle circle+` add to the current \langle key-value list \rangle ; `every theory circle circle'` replaces it.

`every theory circle text` = \langle key-value list \rangle *theory circle*
`every theory circle text'`
`every theory circle text+` Style applied to the texts used in constructing elements of tag type `theory circle`. By default the texts are placed along the semicircular paths corresponding to the upper and lower halves of the ring formed by the `theory circle circles`. This means the colour used here should differ from that used to fill the circles, given the default styles.

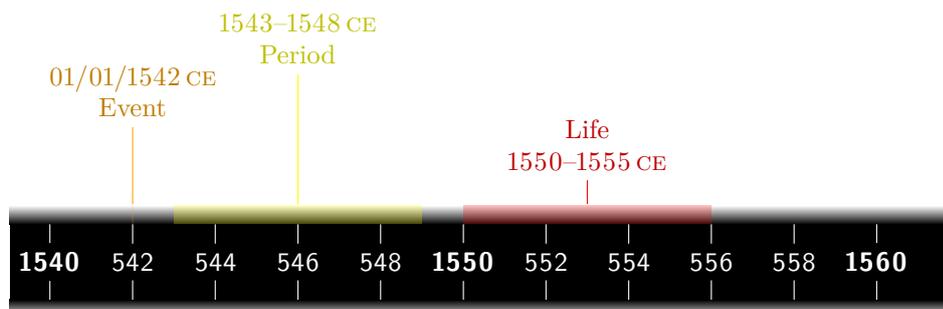
Default: `decoration={text effects along path, text={##1}, text effects/.cd, fit text to path, text=chronos@prifliw@cefndir, characters={text along path, font=\scriptsize\s}`
`decorate`

`every theory circle text` and `every theory circle text+` add to the current \langle key-value list \rangle ; `every theory circle text'` replaces it.

`text tag yshift` = \langle dimension \rangle *life, event, period, theory*
dimension key

The `yshift` to apply when placing the `text tag` if `yshift` is otherwise `0pt` and `at` is unset. You should probably never use this key in the context of a particular element, because `yshift` works just as well and will probably be more reliable and certainly faster. Moreover, unlike `yshift`, which can be used to adjust a position set with `at`, `text tag yshift` cannot. If `at` is used, `text tag yshift` is ignored. It makes sense to set this globally if you want all elements or all elements belonging to a particular tag to be shifted by some specified distance from the timeline. For example,

```
\begin{chronos}[
  life/text tag yshift=10pt,
  event/text tag yshift=30pt,
  period/text tag yshift=50pt,
  theory/text tag yshift=70pt,
]
\end{chronos}
```



Theory

The following keys take the form `{every} <tag>`, optionally followed by prime or plus. *They should not be used to configure elements for which other global keys exist, such as colours, connections, connectors, date formats, lines or text tags.* Generally, these keys should be unnecessary and are best avoided, although they may occasionally be convenient.

<pre>every life = {(key-value list)} every life' every life+ key</pre>	<pre>life</pre> <p>Additional configuration for all elements of tag type life. These do much the same as <code>life/tag</code>, <code>life/tag+</code> and <code>life/tag'</code>, but should <i>never</i> be used when creating a specific element. <code>every life</code> and <code>every life+</code> add to the current <code><key-value list></code>; <code>every life'</code> replaces it.</p>
<pre>every event = {(key-value list)} every event' every event+ key</pre>	<pre>event</pre> <p>Additional configuration for all elements of tag type event. These do much the same as <code>event/tag</code>, <code>event/tag+</code> and <code>event/tag'</code>, but should <i>never</i> be used when creating a specific element. <code>every event</code> and <code>every event+</code> add to the current <code><key-value list></code>; <code>every event'</code> replaces it.</p>
<pre>every period = {(key-value list)} every period' every period+ key</pre>	<pre>period</pre> <p>Additional configuration for all elements of tag type period. These do much the same as <code>period/tag</code>, <code>period/tag+</code> and <code>period/tag'</code>, but should <i>never</i> be used when creating a specific element. <code>every period</code> and <code>every period+</code> add to the current <code><key-value list></code>; <code>every period'</code> replaces it.</p>
<pre>every theory = {(key-value list)} every theory' every theory+ key</pre>	<pre>theory</pre> <p>Additional configuration for all elements of tag type theory. These do much the same as <code>theory/tag</code>, <code>theory/tag+</code> and <code>theory/tag'</code>, but should <i>never</i> be used when creating a specific element. <code>every theory</code> and <code>every theory+</code> add to the current <code><key-value list></code>; <code>every theory'</code> replaces it.</p>
<pre>every info = {(key-value list)} every info' every info+ key</pre>	<pre>info</pre> <p>Additional configuration for all elements of tag type info. These do much the same as <code>info/tag</code>, <code>info/tag+</code> and <code>info/tag'</code>, but should <i>never</i> be used when creating a specific element. <code>every info</code> and <code>every info+</code> add to the current <code><key-value list></code>; <code>every info'</code> replaces it.</p>

9.6 Adding Connections, Using Colours and Accessing Styles

To access the colour list used for the timeline etc., see sections 8.3 and 8.4.5. For details of the way colour list are assigned to elements, see section 8.8.

Life, event, period and theory elements are designed to be connected not only, in the case of those which are connectable, to the timeline, but also to each other. To ensure consistent styling, this requires the use of `chronos` styles in `TikZ` commands.

In addition, densely-packed timelines sometimes require non-standard paths be used to connect a minority of elements to the timeline in an efficient way. Again, this requires access to `chronos` styles.

`chronos connect` = $\{ \langle tag \rangle \} : \{ \langle element name \rangle \}$ *life, event, period, theory*
style

This sets the style used for connections belonging to elements of type $\langle tag \rangle$ with the colour assigned to $\langle element name \rangle$ (section 8.8). For example,

```
\draw [chronos connect=life:johannes gutenberg] (connector johannes gutenberg) -- (
connector printing press) (connector johannes gutenberg2) -|- (connector movable type) (
connector johannes gutenberg3) -- ++(5pt,0pt) |-| (connector gutenberg bible);
```

This will draw a line using the style for connections of tag type `life` and the colour assigned to the element named `johannes gutenberg`. Note the use of connectors on both the element's own text tag and on other elements' text tags. In this case, tag `johannes gutenberg` is being connected to tag `printing press`, tag `movable type` and tag `gutenberg bible`.

The following four keys provide analogous access to the styles and colour list used for `chronos` connectors, text tag connectors, lines and text tags and are used in the same way.

`chronos create chronos` = $\{ \langle tag \rangle \} : \{ \langle element name \rangle \}$ *life, event, period*
connector

style This sets the style used for `chronos` connectors belonging to elements of type $\langle tag \rangle$ with the colour assigned to $\langle element name \rangle$.

`chronos create text tag` = $\{ \langle tag \rangle \} : \{ \langle element name \rangle \}$ *life, event, period, theory*
connector

style This sets the style used for text tag connectors belonging to elements of type $\langle tag \rangle$ with the colour assigned to $\langle element name \rangle$.

`chronos mark line` = $\{ \langle tag \rangle \} : \{ \langle element name \rangle \}$ *life, event, period*
style

This sets the style used for lines (on or near the timeline) belonging to elements of type $\langle tag \rangle$ with the colour assigned to $\langle element name \rangle$.

`chronos text tag` = $\{ \langle tag \rangle \} : \{ \langle element name \rangle \}$ *life, event, period, theory, info*
style

This sets the style used for text tags belonging to elements of type $\langle tag \rangle$ with the colour assigned to $\langle element name \rangle$.

We can also use the colour assigned to `johannes gutenberg` directly. Perhaps, for example, we'd like to put a book symbol near this element in the appropriate colour.

Example: `\node [colour johannes gutenberg, above left=5pt and 10pt of tag johannes gutenberg.north west, anchor=south east, inner sep=0pt] { \langle book-symbol \rangle };`

10 Drawing on Chronos Layers

See section 6.4.

`on chronos background layer` Apply to a scope to draw everything inside on layer `chronos background`.
style

```
\begin{scope}[on chronos background layer]
  \node {Something};% in between the regular background and chronos middle ground
\end{scope}
```

`on chronos middle ground` Apply to a scope to draw everything inside on layer `chronos middle ground`.
layer

```
\begin{scope}[on chronos middle ground layer]
  \node {Something};% behind the main layer and chronos background
\end{scope}
```

`on chronos foreground layer` Apply to a scope to draw everything inside on layer `chronos foreground`.
style

```
\begin{scope}[on chronos foreground layer]
  \node {Something};% above the main layer but behind chronos overlay.
\end{scope}
```

`on chronos overlay layer` Apply to a scope to draw everything inside on layer `chronos overlay`.
style

By default, `chronos` puts only debugging information on `chronos overlay`, which means drawing on this layer should always draw on top of anything constructed by the package code.

```
\begin{scope}[on chronos overlay layer]
  \node {Something over everything else.};
\end{scope}
```

In addition, `chronos` never puts anything on the non-`chronos` PGF/TikZ `background` layer and it would be difficult to persuade it to do so without rewriting internal code. Drawing on *this* layer, therefore, is almost guaranteed to end up behind everything constructed by the package code³⁴.

```
\begin{scope}[on background layer]% fill area below the timeline
  \fill [blue!25!white] (chronos pre |- chronos bottom) rectangle (chronos post-foot);
\end{scope}
```

11 Externalising Chronos Timelines with Memoize

As explained in section 15, `chronos` timelines cannot be externalised with PGF/TikZ's `external`. Since PGF/TikZ, in general, and `chronos`, in particular, can be rather slow to compile, this is serious issue. If you only have a two or three small timelines, the compilation time will be negligible. But if you have a large, densely packed timeline or many timelines, compilation time will quickly become excessive.

Fortunately, `chronos` environments *can* be externalised. Moreover, they can be externalised more conveniently, more robustly and more securely, without the need for a separate compilation for each `chronos`. This means compilation is only a little slower when the timelines are being compiled (whereas compilation would be far slower with the external `pgf/ti\emphkz` library, even if it worked) and subsequent compilations are fast.

Sašo Živanović's `memoize` has no trouble compiling this documentation and externalising its timelines. `Memoize` is a little more trouble to set up initially than the external `pgf/ti\emphkz` library, but requires far less fine-tuning once configured.

To externalise `chronos` timelines, *you must first setup memoization as explained in* `memoize's documentation`³⁵.

`Chronos` supports automemoization out-of-the-box³⁶: to enable automatic memoization of `chronos` environments, simply load `memoize` early in your preamble. `Chronos` will then enable 'automemoization' for all timelines³⁷.

All `chronos` styles (except `default`) and all colour schemes (except `default`) are defined so that modification will automatically trigger the recompilation of all `chronos` timelines which use them.

³⁴Unless nefarious TeXnicians have interfered with your installation. It is even quite unlikely a bug would cause this kind of problem, though bugs will doubtless cause many others.

³⁵By default, `memoize` uses `perl` and requires the installation of a couple of libraries. If you use Linux or have `python` already installed, I'd recommend using this method as it requires only a single extra library, is faster and more robust. If you do not wish to use either `perl` or `python`, you can use TeX, but I have not personally tested this method as it is slower and less secure.

³⁶This fantastic feat was accomplished by copying a line of code from `memoize's` manual and substituting `chronos` for the appropriate word. Even I managed to achieve this without major incident.

³⁷Of course, memoization can be disabled permanently or temporarily for some or all timelines. See `memoize's` documentation for details or look at the code for this document, which disables memoization for `fig. 1` to prevent destruction of hyperlinks.

12 Deferring Code

If you don't know why you might want to use the keys in this section, you don't need to use them.

```
timeline config = {<code>}
```

```
timeline config'
timeline config+
  key
```

Execute additional *<code>* after `chronos` has processed the keys at the start of the `chronos` environment, but before further processing the resulting configuration and constructing the timeline. These keys are provided primarily for use in `chronos` style definitions, but may occasionally be useful elsewhere. `timeline config` and `timeline config+` add to the current code; `timeline config'` replaces it. Note that `timeline config'` is destructive: it obliterates any existing code `chronos` has installed, which may be entirely unrelated to the code now being stored. `Chronos` style authors should never use this form. Even if the code is for purely private use in a locked room with no internet access, you should stick to the additive forms unless your memory is infallible and you always remember to use it.

12.1 Additional TikZ

Generally, you can mix arbitrary TikZ code freely into the body of the `chronos` environment. For example, this is how to add connections between text tags or to decorate your timeline with symbols or ornaments.

However, sometimes you might want to add something *after* `chronos` has finished. You might, for example, want to do something after the frame is drawn or place something relative to headings or subheadings. Two sets of keys are provided for this purpose. One set enable you to execute arbitrary TikZ code within the picture's bounding box; the other enables you to do so outside. Generally, it is the first set you will want to use; the second are useful in a narrower range of cases and for debugging purposes.

```
chronos tikz' = {<TikZ commands>}
```

```
chronos tikz
chronos tikz+
  key
```

Commands to execute after the *<timeline additions specification>* and any frame, headings and subheadings are drawn, but before debugging information is added (see section 14). `chronos tikz` and `chronos tikz+` add to current material; `chronos tikz'` replaces it. Material added with these keys is included in the final picture's bounding box. If you draw outside the frame and outer border, for example, the final bounding box expands to accommodate it. *If you aren't sure which set of keys to use, choose these.*

```
chronos tikz outside bb' = {<TikZ commands>}
```

```
chronos tikz outside bb
chronos tikz outside bb+
  key
```

Commands to execute after the *<timeline additions specification>* and any frame, headings and subheadings are drawn, but before debugging information is added (see section 14). `chronos tikz outside bb` and `chronos tikz outside bb+` add to current material; `chronos tikz outside bb'` replaces it. Material added with these keys is excluded when the final picture's bounding box is determined. If you draw outside the frame and outer border, for example, `TEX` will treat it as if it didn't exist and you will need to ensure adequate space is available to accommodate it manually. *If you aren't sure which set of keys to use, avoid these.*

Finally, you might want to add material at some specific point in the construction of the picture (e.g. after headings but before the frame). The following sets of keys facilitate such additions.

```
before headings' = {<TikZ commands>}
```

```
before headings
before headings+
  key
```

Commands to execute after the *<timeline additions specification>*, but before constructing any headings. `before headings` and `before headings+` add to current material; `before headings'` replaces it.

```
before drawing frame' = {<TikZ commands>}
```

```
before drawing frame
before drawing frame+
  key
```

Commands to execute after the *<timeline additions specification>* and any headings and subheadings are drawn, but before constructing any frame. `before drawing frame` and `before drawing frame+` add to current material; `before drawing frame'` replaces it.

13 Custom Schemes and Styles

The macros and keys explained in this section enable you to define custom colour schemes and chronos styles. These may then be used in the same way as those provided by `chronos` (section 7).

Customisation is a two-stage process. Chronos styles should not define colours definable by colour schemes.

Colour schemes are straightforward to define; chronos styles are a bit trickier.

13.1 Defining Chronos Colour Schemes

As explained in section 7.2, in addition to the default colours, `chronos` currently provides `blues`, `contninety`, `cronoleg`, `lavender`, `modern`, `offlinebasic`, `offlinealt`, `sobriety` and `xcolseries`³⁸ (table 2). `xcolseries` demonstrates the use of `xcolor` colour series in chronos colour lists. `contninety`, `modern`, `offlinebasic` and `offlinealt` illustrate the use of colour schemes to support chronos styles which require minimal modifications of other colour schemes.

New colour schemes should follow the examples in `chronos-lib-colschemes.sty`³⁹. For instance, here's the code to set up `blues`:

```
\chronosnewcolourscheme{blues}{% chronos-lib-colschemes.sty
  timeline foreground=DodgerBlue4,
  timeline background=DodgerBlue2,
  default below={Cerulean!50!DodgerBlue4,Cerulean!50!DodgerBlue3,Cerulean!50!DodgerBlue2,
Cerulean!50!DodgerBlue1,Cerulean},
  default above={Cerulean!50!DodgerBlue4,Cerulean!50!DodgerBlue3,Cerulean!50!DodgerBlue2,
Cerulean!50!DodgerBlue1,Cerulean},
  foreground=DodgerBlue4,
  background=white,
}
```

This is intended for ‘off line’ timelines so it doesn’t include colours for a timeline border, though `chronos` will derive such colours anyway, as explained below.

There are two pitfalls in defining a colour scheme. First, definitions cannot utilise other chronos colours at this stage. You cannot, therefore, define the middle border colour, for example, in terms of the outer and inner colours.

Second, scheme names must consist of letters only as they are used to create new macros.

```
\chronosnewcolourscheme [(existing scheme)] {<name>} {<key-value list>}
macro
\chronosnewcolourscheme [(existing scheme)] {<name>} {<key-value list>}
macro
```

If *<existing scheme>* is specified, it should be the name of an existing colour scheme; otherwise, a default set of colours is loaded. *<name>* is the name of the new colour scheme and must be a unique string of alphabetic characters suitable for use in a macro name. *<key-value list>* is a list of key-value pairs from the list in table 13.

Schemes need not use all keys⁴⁰. It is sufficient to specify the required deviations from *<existing scheme>*. For example, here's the code to set up `offlinealt`,

```
\chronosnewcolourscheme[cronoleg]{offlinealt}{%
  timeline foreground=blue!40,
```

³⁸Note that `xcolseries` uses the `hsb` colour model, which is not supported by PGF/TikZ. If loading this set of colours directly, add `/utils/exec=` to `chronos`'s optional argument. This is not necessary if loading a `chronos` style which utilises `xcolseries`. In either case, all colours in the current `chronos` environment will be converted to `rgb`.

³⁹For historical reasons, `cronoleg` is non-standardly defined as it was the default scheme during most `chronos` development. The current implementation of this scheme is officially internal. The implementation — as opposed to the scheme — is highly likely to change in backwards-incompatible ways without notice. This warning does not apply to *usage* of the colour scheme, but you should not take it as a model for a new scheme, except to pass it as an option to `\chronosnewcolourscheme`.

⁴⁰In fact, they need not use any, though a colour scheme which uses none would serve no purpose.

Table 13: Keys for `\chronosnewcolourscheme`. Note that neither ‘colour’ nor ‘color’ appears in any key.

Key	Expected Argument Type	Example
foreground	<i><colour name></i>	chronosblack
background	<i><colour name></i>	chronoswhite
timeline foreground	<i><colour name></i>	chronosCerulean
timeline background	<i><colour name></i>	chronosDodgerBlue4!50!chronosblack
timeline border outer	<i><colour name></i>	chronoswhite
timeline border inner	<i><colour name></i>	chronosCerulean
timeline border middle	<i><colour name></i>	chronosDodgerBlue4!50!chronosblack
life/default	<i><colour name></i>	chronosDodgerBlue4
event/default	<i><colour name></i>	chronosDodgerBlue4
period/default	<i><colour name></i>	chronosDodgerBlue4
theory/default	<i><colour name></i>	chronosDodgerBlue4
info/default	<i><colour name></i>	chronosDodgerBlue4
default above	<i><list of colour names></i>	chronosRed, chronosOrange, chronosYellow, chronosGreen, chronosBlue, chronosMidnightBlue, chronosViolet
default below	<i><list of colour names></i>	chronosCerulean!50!chronosDodgerBlue4, chronosCerulean!50!chronosDodgerBlue3, chronosCerulean!50!chronosDodgerBlue2, chronosCerulean!50!chronosDodgerBlue1, chronosCerulean
life/above	<i><list of colour names></i>	chronosDeepPink2, chronosDarkOrange1, chronosFirebrick1, chronosPurple0, chronosWildStrawberry, chronosOrangeRed1, chronosDarkGoldenrod1, chronosDarkOrchid3
life/below	<i><list of colour names></i>	chronosDodgerBlue3, chronosGreen3, chronosBlue3, chronosSpringGreen4, chronosDeepSkyBlue2, chronosForestGreen, chronosPeriwinkle, chronosSeaGreen3
event/above	<i><list of colour names></i>	chronosThistle4, chronosThistle4!.5!chronosThistle3, chronosThistle3, chronosThistle3!.5!chronosThistle2, chronosThistle2
event/below	<i><list of colour names></i>	chronosSeashell4, chronosSeashell4!.5!chronosSeashell3, chronosSeashell3, chronosSeashell3!.5!chronosSeashell2, chronosSeashell2
period/above	<i><list of colour names></i>	chronosMistyRose4, chronosMistyRose4!.5!chronosMistyRose3, chronosMistyRose3, chronosMistyRose3!.5!chronosMistyRose2, chronosMistyRose2
period/below	<i><list of colour names></i>	chronosIvory4, chronosIvory4!.5!chronosIvory3, chronosIvory3, chronosIvory3!.5!chronosIvory2, chronosIvory2
theory/above	<i><list of colour names></i>	xcolor s2!![0],xcolor s2!![1],xcolor s2!![2],xcolor s2!![3],xcolor s2!![4],xcolor s2!![5],xcolor s2!![6],xcolor s2!![7],xcolor s2!![8],xcolor s2!![9],xcolor s2!![10],xcolor s2!![11], xcolor s2!![12],xcolor s2!![13],xcolor s2!![14],xcolor s2!![15]
theory/below	<i><list of colour names></i>	xcolor g2!![0],xcolor g2!![1],xcolor g2!![2],xcolor g2!![3],xcolor g2!![4],xcolor g2!![5],xcolor g2!![6],xcolor g2!![7],xcolor g2!![8],xcolor g2!![9],xcolor g2!![10],xcolor g2!![11],xcolor g2!![12],xcolor g2!![13],xcolor g2!![14],xcolor g2!![15]

}

13.1.1 How Colour Schemes are Processed

When a colour scheme is loaded, `chronos` processes the settings in six stages.

1. The specified (*existing scheme*) or defaults are loaded.
2. Keys for the ‘core’ colours `foreground` and `background` are set and flipped to provide default settings for the ‘core derivative’ colours `timeline foreground` and `timeline background`.
3. Keys for the ‘core derivative’ colours `timeline foreground` and `timeline background` are set and the resulting four colours used to derive default settings for the ‘core border’ colours `timeline border inner`, `timeline border middle` and `timeline border outer`. In particular, `timeline border inner` is set to match `timeline background`, `timeline border outer` is set to `background` and `timeline border middle` is set to a 50-50 mix of the two.
4. Keys for the ‘core border’ colours `timeline border inner`, `timeline border middle` and `timeline border outer` are set. The main `foreground` colour is assigned to the ‘elemental’ default colours `life/default`, `event/default`, `period/default`, `theory/default` and `info/default`.
5. Keys for the ‘elemental’ default colours `life/default`, `event/default`, `period/default` and `theory/default` are set.
6. *Much later*, after the user configuration for the `chronos` environment has been read, `chronos` potentially flips the ‘core derivative’ colours `timeline foreground` and `timeline background`. See section 13.2.

Only after this sixth stage are the ‘public’ names listed in table 14 assigned to the final set of colour scheme-definable colours.

13.2 Defining Chronos Styles

The current method for creating `chronos` styles is straightforward in theory, but potentially hazardous in practice. Here’s an example from `chronos-lib-styles.sty`.

```
\pgfqkeys{/chronos}{%
  blues below/.style={%
    /chronos/.cd,
    blues below/.meaning to context,
    colour scheme=blues,
    rotate all colours,
    timeline={%
      timeline years=above,
      timeline marks,
      timeline minor marks,
      step minor year=50,
      step divisions=10,
      step major year=100,
      dates=1550:2050,
      timeline height'=3pt,
      timeline line={chronos timeline foreground colour,double=chronos timeline
background colour,line width=\timelineht/3,double distance=\timelineht/3},
      timeline arrow,
      conditional timeline arrow={%
        timeline/timeline line+={Bar-Latex,shorten <=-\timelineht/3,shorten >=-3pt-2.1\
timelineht},
        timeline/timeline width--={3pt+2.43\timelineht},
        before headings+={\path (chronos post) -- ++(3pt+2.1\timelineht,0pt) (chronos
pre) -- ++(-\timelineht/3,0pt)};}
```

```

    }{,
    timeline mark={chronos timeline foreground colour,line width=.6pt,shorten >=-4pt},
    timeline minor mark={chronos timeline foreground colour,line width=.5pt,shorten
>=-3.5pt},
    timeline bare mark={chronos timeline foreground colour,line width=.3pt,shorten
>=-2.5pt},
    timeline year={fill=none,text=chronos timeline foreground colour,rotate around
={45:(chronos year \chronosyeari |- chronos top)}},
    major step font=\sffamily\footnotesize\tlstyle,
    timeline years anchor=south west,
    minor step font=\sffamily\scriptsize\tlstyle,
    timeline margin'=17.5pt,
  },
  minor year format={!Y},
  every event below,
  every life below,
  every period below,
  levels=0:3,
  headings style+={text=chronos main colour!75!chronos main background colour,font=\
small\itshape\bfseries},
  subheadings style+={text=chronos main colour!75!chronos main background colour,font
=\footnotesize\itshape},
  main/title+={font=\LARGE,text=chronos timeline foreground colour,draw=chronos
timeline background colour,semithick},
  main/frame+={thick,draw,chronos timeline foreground colour,double=chronos timeline
background colour},
  copyright={font=\footnotesize\sffamily, inner sep=0pt, outer sep=0pt, text=chronos
timeline foreground colour!50!chronos main background colour},
  copyright/rotate=90,
  copyright/tag anchor=north west,
},
}

```

This definition is chosen because it is one of the most technically complex examples. This complexity is a function of several factors: it uses *off-line* years; the year labels are rotated; the line involves two arrow tips; and the line is drawn with `double`.

Note the following:

1. colours listed in table 13 are used but not defined;
2. instead, a custom colour configuration is set by loading an appropriate colour scheme;
3. there is a weird looking `\chronosyeari` in the definition of `timeline year`;
4. `timeline/timeline arrow` and `timeline/conditional timeline arrow` enables use of arrow tips to be toggled off;
5. `dates` are defined, even though they are almost certainly wrong in most cases;
6. `.meaning to context` is used, even though the user might not have loaded `memoize`, which defines it.
7. some fonts use a non-standard command `\tlstyle`.

Item 7 need not concern us here. If certain packages are loaded, it ensures tabular, lining figures; if not, `chronos` provides a command with this name at the end of the preamble by simply `\letting` it to `\upshape`.

Regarding item 5, the standard `chronos` styles all define `dates`, but whether they should do so is another question. On the one hand, if they are not defined (as they are not if no `chronos` style is loaded), `chronos` will generate an error, alerting the user to the deficiency. Since it is highly unlikely any default choice will suit any user, let alone most of them, an error might be considered

appropriate. On the other hand, some chronos styles are far more suitable for some temporal ranges than others. For example, consider this excerpt from the definition of `contemporary` 90:

```

timeline={%
  timeline marks,
  timeline minor marks,
  timeline mark={ultra thick},
  timeline minor mark={thick},
  step divisions=4,
  step major years=2,
},

```

This is fine for a timeline of a decade or two, but quite unsuitable for one representing either the period 3,000 BCE–2025 CE or the first half of 1857. While a user can always modify these settings, along with the `dates`, a default range provides a sense of the temporal duration the chronos style is suitable for ‘out-of-the-box’.

The author of this package has found a comfortable spot on a convenient fence and intends to stay there, whatever the provided chronos styles might suggest. The reader is warned to make the most of the fences available here, as there are none whatsoever in the next section.

13.2.1 How (Not) to Customise Colours

Items 1 and 2 are the most important. *Chronos styles MUST NOT set core, core derivative or core border colours, where ‘core, core derivative and core border colours’ refer to those listed in tables 13 and 14.* In many cases, violating this rule may appear to work, but in others doing so will produce weird results or errors.

Moreover, *chronos styles should not set any other colour key or colour list directly.* In many cases, violating this rule may appear to work, but in others doing so will cause things not to work as expected.

To summarise, *if it can be done by a colour scheme, it should be done by a colour scheme*⁴¹.

The reason for this restriction is that the colours are not finalised and the public colour names are not defined when the colour scheme and/or chronos style are read. Initially, `chronos` assigns colours only to internal names. When the user configuration in the `<chronos preamble>` has been read, `chronos` starts the `tikzpicture` environment and further processes the configuration before drawing the timeline. As part of this processing, `chronos` makes changes to colours in specified circumstances.

In particular, the colours assigned to the `timeline foreground` and `background` are switched if three conditions are satisfied.

1. The internal colour names for `chronos timeline foreground colour` and `chronos timeline background colour` evaluate to the same colour specification.
2. One of the specifications is identical to the colour specification for `white`.
3. `timeline years` is not on line.

Condition 3 cannot be determined until the complete configuration has been read. In particular, it is not known when colour schemes and chronos styles are read. While it is recommended users select a chronos style congruent with their preferred setting for `timeline years`, this is intended to make configuration easier and is not a requirement.

Only *after* colours are potentially switched are the public names listed in table 14 assigned, long after colour schemes and chronos styles have been read.

It is nonetheless possible, indeed recommended, to *use* the public names in chronos styles, though they cannot be used in colour schemes. It is only *defining* them at this stage which is problematic.

⁴¹That is, ‘can implies ought’.

Table 14: Keys and names for chronos colours. Note that neither ‘colour’ nor ‘color’ appears in any key in the first column, but in every key in the second. In the second column, ‘color’ may be substituted for ‘colour’ in any name.

		Colour Schemes Key	Later Accessible As		
MUST NOT define!	CORE	core { foreground	chronos main colour	CORE	
		background	chronos main background colour		
		derivative { timeline foreground	chronos timeline foreground colour		
	background	chronos timeline background colour			
	border	timeline border outer	chronos timeline border outer colour		
		inner	chronos timeline border inner colour		
		middle	chronos timeline border middle colour		
	Should NOT touch!	ELEMENTAL	default colours { life/default		-
			event/default		-
period/default			-		
theory/default			-		
info/default			-		
colour lists		default above	-		
		default below	-		
		life/above	-		
		life/below	-		
		event/above	-		
		event/below	-		
		period/above	-		
		period/below	-		
		theory/above	-		
theory/below	-				

Here is an example from the definition of `modern` in `chronos-lib-styles`:

```
✓ timeline line={chronos timeline background colour, opacity=1},
  period/line={fill=chronos timeline foreground colour, blend mode=overlay},
  life/line={fill=chronos timeline foreground colour, blend mode=overlay},
  event/line={draw=chronos timeline foreground colour, thick, blend mode=overlay},
  every text tags={fill=chronos main background colour, text=####1, fill opacity=.75,
text opacity=1, draw=none, rounded corners, align=center, font=\sffamily\footnotesize},
```

This is perfectly proper⁴². However, if you were to include something such as

```
✗ timeline border middle colour=chronos timeline border inner colour!50!chronos timeline
border outer colour,
```

you would get an error complaining about the use of undefined colours. The definition of `timeline border middle colour` is the prerogative of the colour scheme and shouldn't feature in a chronos style at all, but this particular definition is illegitimate in any case because neither `chronos timeline border inner colour` nor `chronos timeline border outer colour` yet exists.

But why shouldn't chronos styles include colour definitions of the kind permitted in colour schemes? Because `chronos` processes the definitions in colour schemes as they are read (section 13.1.1). If you put

```
✗ foreground=SlateBlue4,
background=Snow1,
```

in a chronos style, *only* these colours will be set. In particular, neither the `timeline` nor any default colours will be affected at all. But if you put this into a colour scheme, `chronos` will derive colours for the `timeline` and set default colours for elements belonging to the various tags. If no other changes are made, the result will be a white-on-blue `timeline` with blue-to-white `timeline` borders and blue as the fallback colour for `tag` elements. (This is probably wrong for `off line` and `chronos` won't correct you because `Snow1` isn't exactly `white`, but that's why colour schemes should do either a bit more or a bit less than this.)

If you wish, your chronos style can load a colour scheme of its own. This is what many of the standard chronos styles do. For instance, here is the sum total of `modern`'s `modern` colour scheme,

```
✓ \chronosnewcolourscheme{modern}{%
  timeline foreground=chronosSilver,
}
```

13.2.2 How to Rotate Years

Item 3 is a function of this style's rotation of the year labels created for the `timeline`. The easiest way to do this is to `rotate around` one of the anchors belonging to the node containing the relevant year. Obviously, we can't do this for each node. We don't know how many there are or what they are named. Instead, we need a hook into the `\foreach` loop `chronos` uses when creating the year labels.

`\chronosyeari`
macro refers to the current year *inside the \foreach loop used to mark years on the timeline*. (`chronos year \chronosyeari`) isn't actually the node, but the point representing the date on the timeline, but the node starts there, so we can use it provided `timeline years anchor` is set appropriately.

```
timeline year={rotate around={45:(chronos year \chronosyeari |- chronos top)}},
timeline years anchor=south west,
```

⁴²At least, it is fine as far as `chronos` goes. Whether it is proper `TikZ` code is not for me to judge.

13.2.3 Hashes

You may have noticed the following line in the excerpt from `modern`'s definition above.

```
every text tags={fill=chronos main background colour, text=####1, fill opacity=.75,
text opacity=1, draw=none, rounded corners, align=center, font=\sffamily\footnotesize},
```

Anywhere you'd normally use a single hash (e.g. `#1`) in defining a TikZ style, you need two (`##1`) because you're nesting that definition within the definition of another style. So it is not surprising to find lines such as

```
connections={draw=##1, {Triangle[width=Opt 3,reversed,length=Opt 1.5]}--{Triangle[width
=Opt 5,reversed,length=Opt 2.5]}}},
```

in `modern`'s definition, but why *four*?

Certain keys require one or more additional doublings of hashes. Anytime you use an `every` key, you need to double. Double double makes four, so we get `text=####1`⁴³.

Elsewhere, a single doubling is generally sufficient, as shown in these lines from the definition of `plain arrow`

```
period/line+={line width=2pt,draw=##1},
life/line+={line width=2pt,draw=##1},
```

Incidentally, PGF doesn't complain if you quadruple the hashes here, though it does so if you make the same mistake elsewhere. So silence does not always indicate correctness. This is important if you're debugging: don't assume because a pattern generates no error in one case, it cannot be the source of an error in another.

Note also that if you say

```
✗ text tags={draw=####1,sharp corners,text opacity=1,fill opacity=1,draw opacity=1,
drop shadow},
```

T_EX will give you an error suggesting you haven't used *enough* hashes,

```
! Illegal parameter number in definition of \tikz@temp.
<to be read again>
```

```
1
```

```
l.113 ]
```

```
? h
```

```
You meant to type ## instead of #, right?
```

```
Or maybe a } was forgotten somewhere earlier, and things
are all screwed up? I'm going to assume that you meant ##.
```

```
?
```

If you double the hashes *again* (`#####1`), you'll get the same error. The actual problem is that you've used too many.

```
✓ text tags={draw=##1,sharp corners,text opacity=1,fill opacity=1,draw opacity=1,drop
shadow},
```

is correct in a `chronos` style definition i.e. twice the number required in the `<chronos preamble>`. If you reduce the hashes to one (`#1`), you'll get no error but the wrong output as the element's colour won't be used.

⁴³For real fun with hashes, may I recommend `chronos` or `forest`?

Despite this, chronos styles should always use chronos keys and hashes for colours.

Hashes are essential for two reasons.

1. Hard-coding colours breaks colour rotation. In order for colours to be not just assigned in rotation, but used for the elements they are assigned to, chronos style definitions must use the colour names passed to them. So hashes are essential when defining the properties of tag elements subject to colour rotation.
2. Chronos *cannot track colours it doesn't know about and it doesn't know about colours passed directly to PGF/TikZ keys*. Hard-coding colours breaks the system of colour names chronos provides. Chronos will assign colour names to colours regardless, but the names will not refer to the colours actually used. They will merely refer to the colours assigned. Chronos styles are responsible for ensuring assigned colours are used so chronos colour names work correctly. Suppose a chronos style includes `event/text tag+={text=red},event/connection+={draw=red}`. Chronos will keep assigning colours to elements of tag type event, but it will not assign 'red' except by happy chance.

Example: `\draw [chronos connect=period:red letter day] ...`

will still work, but may well use black or navy blue rather than the pillar box red expected. Since this referencing system works for some elements not subject to colour rotation at all, such as those belonging to tag info and applies even when colour rotation is disabled completely, it constitutes a more general reason to avoid hard-coding colours, even if the effects may be less immediately noticeable in some timelines.

13.2.4 Timeline Arrow

Chronos styles must decide whether to support timelines with and/or without one or more arrow tips and/or line caps. In deciding this, note the following points.

- Only `off line` styles can support these features.
- Adding, removing or modifying a tip or cap requires adjusting the `timeline width`. This is because the length available for representing time is reduced when some proportion of the timeline line is used for a tip or cap. Chronos adjusts automatically for `timeline margins` and `timeline era margins`, but styles are responsible for other adjustments.
- Supporting both arrowed and non-arrowed variants therefore requires conditionalised code.
- Each arrow tip and line cap requires a bespoke adjustment, even if used in default form.
- Users may legitimately use `timeline/timeline arrow` and `timeline/no timeline arrow` after loading a chronos style.
- Chronos styles may legitimately ignore these keys.
- Chronos styles must delay finalising the content of `timeline` until the end of the `<chronos preamble>` if they wish to support variants with and without tips and/or caps.

See `timeline/timeline arrow` and `timeline/no timeline arrow`.

```

timeline/conditional = {(<key-value list if arrow/cap>)}key-value list otherwise
timeline arrow
  key

```

This key expects two arguments: `<key-value list if arrow/cap>` should be a list of key-values to be executed if `timeline/timeline arrow` is true; `<key value list otherwise>` should be a list of key-values to be executed if it is false. Chronos will switch the key path to `/chronos/` prior to using the list, but the `timeline` prefix must be specified if required. The effect is to add code to the style `timeline/do timeline arrow` which executes `<key-value list if arrow/cap>` if `timeline arrow` is true and `<key-value list otherwise>` otherwise. More specifically, the code used to implement this mechanism is equivalent to

```

conditional timeline arrow/.code 2 args={%
  \pgfqkeys{/chronos}{%
    lline1 amser/.cd,

```

```

    timeline@arrow/.style={/chronos/.cd,#1},
    no@timeline@arrow/.style={/chronos/.cd,#2},
    do timeline arrow/.add code={%
      \ifchronostimelinearrow
        \tikzset{/chronos/lilinell amser/timeline@arrow}%
      \else
        \tikzset{/chronos/lilinell amser/no@timeline@arrow}%
      \fi
    },
  }%
},

```

If the timeline uses off line yearss, `\pgfqkeys{/chronos/timeline}{do timeline arrow}` is executed after `timeline/timeline height` is finalised.

Example: See below.

`timeline/do timeline arrow`
key

Chronos styles are expected to set this *via* `timeline/conditional timeline arrow`, which causes it to be executed in `timeline` config, but they could also execute it explicitly if required.

Default: dependent on other options

For example, `lines on line` supports arrowed and non-arrowed variants using

```

lines on line/.style={% https://tex.stackexchange.com/a/324453/
  /chronos/.cd,
  ...
  timeline={%
    timeline width'=120mm,
    ...
    timeline arrow,
    conditional timeline arrow={%
      timeline/timeline width'=-20mm,
      timeline/timeline line+={shorten >=-20mm, -{Triangle Cap[length=20mm]}},
      before headings+={%
        \path (chronos post) -- +(20mm,0pt);
      },
    }{ },
  },
  ...
},

```

`timeline arrow` requests an arrow by default, but does nothing else. `conditional timeline arrow` sets up the style keys to execute if `timeline arrow` is still enabled when `do timeline arrow` is executed. At this stage, then, no actual changes are applied to the style to be applied to the timeline.

The actual effects on the timeline's style are determined only at the end of *(chronos preamble)* when `timeline/do timeline arrow` is executed. Hence, the user may override the style's use of `timeline arrow` by writing `timeline/timeline arrow=false` or `timeline/no timeline arrow` after loading `lines on line`.

Styles which support timeline arrows must do the following to ensure correct results⁴⁴.

1. Set `timeline/timeline arrow` if an arrow, non-default line-cap or similar is to be default.

⁴⁴This is necessary because

`chronos` discards the bounding box which includes the arrows immediately after drawing them and it is not possible (as far as I can tell) to extract the required information, even though PGF has just performed all these calculations itself.

2. Use `timeline/conditional timeline arrow` if a non-arrow is to be supported and configure the arrow/cap/spacer(s) *only* using this conditional.
3. Decrease `timeline/timeline width` by the total length of arrows, caps and spacers. At the beginning of the `chronos` environment, this dimension must equal the actual length available for the `timeline era margins`, `timeline margins` and the representation of time, else marks and years may be placed onto arrows or caps.

The recommended way to do this at present is to

- (a) calculate the total length of arrows, caps and spacers by hand and use `timeline/timeline width' = {\total length}` to subtract it from the user-specified `width`⁴⁵;
 - (b) add `shorten >=` and/or `shorten <=`, as appropriate, to increase the length of the line just while it is being drawn.
4. Ensure the bounding box includes any arrows, caps and spacers.

One way to achieve this is to

- (a) use `before headings+` to place coordinates at the tip and very tail of the arrow/cap/spacer(s).
5. Calculations must account for `\pgflinewidth` and, if applicable, any use of `double`, in order to avoid overfull boxes.

13.2.5 Styles and Automemoization

It is recommended that `chronos` styles are configured so that externalised `chronos` timelines which use them are automatically recompiled if the styles' definitions change. This can be achieved by adding `<name of style>/.meaning` to `context` to each `chronos` style's definition. For example, the packaged styles all use the following template to begin their definitions.

```
\pgfqkeys{/chronos}{%
  <name of style>/.style={%
    /chronos/.cd,
    <name of style>/.meaning to context,
    ...
  },
}
```

This is safe, even if `memoize` isn't used, because `chronos` provides a fallback key handler, `.meaning` to `context` which does nothing.

13.3 Defining Styles for Additional Elements

Due to the way `chronos` manages `tag` contexts, creating custom styles to apply to the additional elements explained in section 9 is not necessarily straightforward.

If you only want to use non-`chronos` keys in your style, however, it *is* straightforward. Simply create whatever PGF/TikZ styles you wish and add them to particular elements as you deem appropriate.

The trouble starts if you want to define style which include `chronos` keys. More particularly, difficulties arise if you want to use keys which are specific to `tag` contexts such as `at` or `tag anchor`. For example, the `timeline` in fig. 1 uses three custom styles, `tag left`, `tag post` and `tag right` to place text tags. Consider the definition of `tag right`,

⁴⁵Accurate calculation requires knowledge of `\pgflinewidth`, any use of `double`, custom options passed to the arrow and details of the formula PGF uses to calculate the length for the specific types of arrow tips and/or line caps configured. In some cases, this information is included in the TikZ manual but, in most cases, you must consult the source of the `arrows.meta` `pgf/ti\emphkz` library.

```

at/.expand once=level -##1.south -| ##2,
tag anchor=north west,
anchor=south west,
xshift=5pt,
connectors=east,

```

It uses `at` and `tag anchor`, which are `tag`-specific `chronos` keys, as well as the `anchor` and `xshift` PGF/TikZ keys. A naïve approach would suggest

```

x tag right/.style 2 args={%
at/.expand once=level -##1.south -| ##2,
tag anchor=north west,
anchor=south west,
xshift=5pt,
connectors=east,
},

```

but this will fail. Less naïvely, you might fiddle with path prefixes, but this won't work reliably either because `chronos` effectively activates some `tag`-specific settings by installing them temporarily under `/chronos`. Meanwhile, it redefines a subset of both the global and `tag`-specific keys to ensure local element-specific settings don't 'leak'⁴⁶.

The result of all this is that you cannot generally use standard PGF/TikZ techniques to define styles involving `chronos` keys for use in creating `chronos` elements belonging to `tags`. Given the aims of `chronos`, this is a significant limitation only partially mitigated by the following workaround.

`Chronos` provides a PGF/TikZ key handler to facilitate the creation of straightforward styles, but the current version has significant limitations I've not been able to solve.

```

.chronos key maker = {(key name)}{(pgf key handler)}{(value)}
key handler

```

`<key name>` should be a name suitable for a PGF/TikZ key. `<pgf key handler>` should be a PGF key handler, without the leading dot, such as `style 2 args` or `ecode`. `<value>` should be the value or definition for `<key name>`. Only handlers which expect a single argument may be used. This limits the maximum number of arguments `<key name>` can absorb to two, since the only PGF key handlers capable of absorbing three or more arguments themselves require two or more.

The key handler is available in the `<chronos preamble>` and in `\chronosset`. It requires a single doubling of hashes.

Example:

Here are the definitions of `tag left`, `tag post` and `tag right` mentioned above.

```

tag right/.chronos key maker={tag right}{style 2 args}{%
at/.expand once=level -##1.south -| ##2,
tag anchor=north west,
anchor=south west,
xshift=5pt,
connectors=east,
},
tag left/.chronos key maker={tag left}{style 2 args}{%
at/.expand once=level -##1.south -| ##2,
tag anchor=north east,
anchor=south east,
xshift=-5pt,
text tag+={align=right},
},
tag post/.chronos key maker={tag post}{style}{%
at=level -##1.south -| chronos end,

```

⁴⁶PGF/TikZ has this type of containment down to a fine art. `Chronos`'s approach is altogether cruder.

```

tag anchor=north west,
anchor=south east,
connect=false,
connectors=east,
},

```

Note `tag post`'s use of the standard coordinate `chronos end` (fig. 3).

14 Debugging

*Note that many keys in this section draw on `chronos overlay layer`. They will typically draw **over** content you've created. This should not be a concern as they are not intended for use in the final document.*

`placeholders` = `on|off`

choice key

If enabled, any helper nodes created with `levels` will be visible rather than invisible⁴⁷ and vertical lines corresponding to headings will be drawn. This option is intended to assist in the creation of complex timelines.

Default: `on`

Initially: `off`

`placeholder lines` = `{(key-value list)}`

style

The style used to draw any lines created when `placeholders` is enabled. The style may be modified or replaced using the usual TikZ techniques, but the settings for nodes should not be altered in a way which changes their size e.g. by setting `line width` or similar.

```

\begin{chronos}
[
  placeholders,
  placeholder lines/.append style={thick},% for the default nodes and similar lines,
  but thicker
  placeholder lines/.style={thin,draw=magenta,<->},% for magenta double-arrowed
  lines with no changes to nodes
]
\end{chronos}

```

Default: `help lines, every node/.append style=rotate=-90,anchor=south,pos=.25,inner sep=0pt`

The following were created for use in developing the package, but some may be more generally useful. Those which seem most likely to be helpful are listed first.

Note that all of the keys which follow ignore the picture's bounding box. This means they will disappear (or partially disappear) with no warning if there is insufficient space. This may be a concern, but having half the timeline disappear from view is worse.

`show coords` = `true|false`

boolean key

Labels a selection of `chronos` coordinates, which may be useful for placement or trouble-shooting purposes.

Default: `true`

Initially: `false`

`show bounding box` = `true|false`

boolean key

Draws the bounding box of the `tikzpicture` containing the timeline.

⁴⁷I am grateful to Qrrbrbirlbel for providing the code implementing this at [TeX StackExchange: 694967](https://tex.stackexchange.com/questions/694967).

Default: `true`

Initially: `false`

`show nodes` = `true|false`

boolean key

If, and only if, `timeline mark eras` is explicitly enabled (as opposed to being enabled just because a timeline spans BCE and CE), draws and labels the nodes containing the era labels on the timeline.

Default: `true`

Initially: `false`

`debug` A convenience key which switches on all four of the options above.

key

```
\begin{chronos}
  debug,
\end{chronos}
```

The following keys are available to customise the output of the options in this section.

`show coordinate colour` = `<colour name>`

`show coordinate color`

colour key

Default: `red`

`show bb colour` = `<colour name>`

`show bb color`

colour key

Default: `green`

`show node colour` = `<colour name>`

`show node color`

colour key

Default: `blue`

`show coordinate` A style used to show coordinates. It is used both directly and indirectly by both `show coord` and `show node coord`. If you want to redefine it, it should take 5 arguments: a colour name, an angle, the name of the coordinate, a dimension and a (possibly empty) key-value list.

style

Default: `fill=#1, circle, anchor=center, inner sep=1pt, text=#1, pin=[#1, inner sep=0pt, pin edge={draw=#1}, pin distance=#4, #5]#2:#3`

`show coord` A style used to show coordinates. If you want to redefine it, it should take 2 arguments: the name of the coordinate and an angle.

style

Default: `/chronos/show coordinate={<chronos show coordinate colour>}{#1}{#2}{30pt}{}`

`show node coord` A style used to show particular points on nodes. If you want to redefine it, it should take 2 arguments: the name of the coordinate and an angle.

style

Default: `/chronos/show coordinate={<chronos show node colour>}{#1}{#2}{30pt}{}`

`\chronosshowcolour` [`<macroname>`]{`<colour name>`}

macro

`\chronosshowcolour*` [`<macroname>`]{`<colour name>`}

macro

`\chronosshowcolor` [`<macroname>`]{`<colour name>`}

macro

`\chronosshowcolor*` [`<macroname>`]{`<colour name>`}

macro

Extract the colour specification of `<colour name>` to the macro `<macroname>`. The starred forms show `<macroname>`; the remainder merely (re)define it. In case it is not obvious, don't use a `<macroname>` you care about as it will be overwritten without warning. By default, an internal macro is used and reused, so, if you don't specify `<macroname>`, you can only inspect one colour specification at a time.

Example: `\chronosshowcolour*{white}`

will show the colour specification of `white` on the terminal.

The remainder are unlikely to be helpful except in debugging `chronos` and no attempt has been made to render their output intelligible.

`\chronosshowpreset` *macro* Show non-default globalised options. This shows the properties⁴⁸ currently recorded as set by the user. This includes selected options set by `chronos` styles and options set with `\chronosset`, but not defaults set by `chronos` when loading. This list is used in deciding whether to change the current setting of an option during timeline configuration. For example, if a user specifically requests `off line years` with a `timeline height` of 50mm in white-on-blue, `chronos` won't override those settings. But if a user asks for `off line years` without specifying `timeline height` or changing the default colours, `chronos` will try to select something reasonable for `timeline height` and assume the user wants black-on-white rather than white-on-white.

The output of `\chronosshowpreset` is unlikely to prove especially enlightening unless debugging `chronos`. Here, for example, is the output when used at the start of a sample `chronos` environment,

```
The sequence \l__chronos_gosod_seq is empty
> .
```

and right after the optional argument has been processed,

```
The sequence \l__chronos_gosod_seq contains the items (without outer braces):
> {angor@blynyddoedd}
> {timeline@years}
> {@digwyddiad@llawn}
> {@byw@llawn}
> {@parhad@llawn}
> {markeras}
> {llinell}
> {cysylltiad}
> {llinell amser}
> {border}.
```

So this user didn't specify any non-default settings in the document preamble or with `\chronosset`, but has either set or specified a `chronos` style which set various options for this particular `chronos` environment, which `chronos` should respect. Note that the output tells us nothing about what has been chosen, but only *that* an explicit choice has been made. For example, `markeras` means the user has decided eras should or should not be marked on the timeline, but does not tell us which.

`\chronosshowfeatures` [*⟨tag⟩*] *life, event, period, theory, info*
macro

Shows properties⁴⁹ assigned to either the current or *⟨tag⟩* context. Note that the output uses the original names for tags, which differ from those documented in this manual. `life`, `event`, `period`, `theory` and `info` correspond to `byw`, `digwyddiad`, `parhad`, `theori` and `gwybodaeth`.

Without an argument, the default list of properties is shown if the command is executed outside a `tag` context; otherwise, the list for the current context is shown. With an argument, the list of properties for *⟨tag⟩* is shown regardless of execution context.

There is no list of properties associated with tag `main`.

Here's the output from `\chronosshowfeatures` inside a `chronos` environment, but outside any tag context,

```
The property list \l__chronos_prop contains the pairs (without outer braces):
> {@tag} => {{/chronos/troi lliwiau=false,/chronos/blynyddoedd yn
unig,/chronos/heb gyfnodau,/chronos/troi lliwiau=true}}
> {@cysylltwr@chronos} => {{coordinate}}
> {@cysylltwr@testun} => {{anchor=center,inner sep=0pt,outer
```

⁴⁸Specifically, the contents of the `expl3` sequence used to record the names of `chronos` properties.

⁴⁹Specifically, `expl3` property lists.

```
sep=0pt,circle, anchor=center, draw=none, fill=none, minimum
size=\pgflinewidth }}
> {@llinell} => {{}}
> {@testun} => {{fill=chronos main background colour, text=###1, fill
opacity=.75, text opacity=1, draw=none, rounded corners, align=center,
font=\sffamily \footnotesize ,draw=###1,sharp corners,text opacity=1,fill
opacity=1,draw opacity=1,drop shadow}}
> {@cysylltiad} => {{draw=###1, {Triangle[width=0pt 3,reversed,length=0pt
1.5]}-{Triangle[width=0pt 5,reversed,length=0pt 2.5]}}}}.
```

and from \chronosshowfeatures[event],

The property list \l__chronos_digwyddiad_prop contains the pairs (without outer braces):

```
> {@cysylltwr@chronos} => {{coordinate}}
> {@cysylltwr@testun} => {{circle, anchor=center, draw=none, fill=none,
minimum size=\pgflinewidth }}
> {@testun} => {{fill=chronos main background colour, text=##1, fill
opacity=.75, text opacity=1, draw=none, rounded corners, align=center,
font=\sffamily \footnotesize ,draw=##1,sharp corners,text opacity=1,fill
opacity=1,draw opacity=1,drop shadow}}
> {@tag} => {{/chronos/blynyddoedd yn unig,/chronos/heb
gyfnodau,/chronos/troi lliwiau=true}}
> {@llinell} => {{draw=chronos timeline foreground colour, thick, blend
mode=overlay}}.
```

Table 15: Public names for `chronos` internal macros defined locally within the *<timeline specification>*.

Public name	Chronos internal name
<code>\ceyearlabel</code>	<code>\chronos@yearce</code>
<code>\bceyearlabel</code>	<code>\chronos@yearbce</code>
<code>\celabel</code>	<code>\chronos@ce</code>
<code>\bcelabel</code>	<code>\chronos@bce</code>
<code>\timelineht</code>	<code>\chronos@height</code>
<code>\timelineborderht</code>	<code>\chronos@borderheight</code>
<code>\timelinewd</code>	<code>\chronos@width</code>
<code>\lineyshift</code>	<code>\chronos@llinell@yshift</code>

Table 16: Public names for `chronos` internal macros defined if undefined at the end of the preamble.

Public name	Chronos internal name
<code>\ceyearlabel</code>	<code>\chronos@yearce</code>
<code>\bceyearlabel</code>	<code>\chronos@yearbce</code>
<code>\celabel</code>	<code>\chronos@ce</code>
<code>\bcelabel</code>	<code>\chronos@bce</code>

15 Compatibility

`Chronos` timelines cannot be externalised using `TikZ`'s external `pgf/ti\emphkz` library⁵⁰.

`TikZ`'s `spy pgf/ti\emphkz` library also appears to be incompatible.

Arrow tips and line caps from `TikZ`'s `arrows pgf/ti\emphkz` library are not supported in `timeline`. Please use `arrows.meta` instead.

`Chronos` defines some commands without either marking them as internal or using a package-specific prefix. These commands are of the following kinds.

- They use Welsh rather than English (`\byw`, `\digwyddiad`, `\parhad`, `\gwybodaeth`, `\theori`, `\cylchtheori` and `\prifdeitl`). These all use `\NewDocumentCommand`. Should they already be defined, $\LaTeX 2_{\epsilon}$ will produce an error and existing definitions will not be overwritten.
- They are defined only locally within the *<timeline specification>*. These provide local access to `chronos` internals and do not use a package-specific prefix for reasons of convenience. These macros are listed in table 15. *Note that some of these macros are also defined conditionally at the end of the preamble. The local definitions described here are unconditional.*
- They are ‘throwaway’, extremely temporary macros such as `\tempa`. These are used only very, very locally. Any macro which needs to retain its definition for more than a few lines uses a `chronos@` prefix unless it is a variable in a `PGF \foreach` loop.
- They are defined only if undefined at the end of the preamble, so existing definitions are maintained without warning or error. This applies to cases where either `chronos` uses a command if it is available (e.g. `\uishape`), but needs a fallback otherwise, or a public macro is made available as a convenience, if the user is not using the name already (e.g. `\celabel`). These macros are listed in tables 16 and 17.
- They are differently-named replacements for a subset of `etoolbox` macros and tests⁵¹, which are defined only if they do not exist. If they already exist, `chronos` produces a warning and continues, hoping for the best. This set of macros is compatible with `etoolbox`, which `chronos` depends on for patching purposes.

⁵⁰However, `chronos` pictures *can* be ‘memoized’. Moreover, if `memoize` is loaded, `chronos` will set up ‘automemoization’ by default. See section 11.

⁵¹They are a response to advice not to mix `expl3` and `etoolbox`. Since I’d originally thought it was better to use `etoolbox` functions than create a slew of wrappers for `expl3` functions, these are the products of the resulting rewrite. Despite my best efforts, the dependency on `etoolbox` remains, but usage is confined to cases where `expl3` does not offer equivalent functionality.

Table 17: Fallback definitions for macros undefined at the end of the preamble.

Functionality used if defined	Chronos fallback definition
<code>\tlstyle</code>	<code>\let\tlstyle\upshape</code>
<code>\plstyle</code>	<code>\let\plstyle\upshape</code>
<code>\uishape</code>	<code>\let\uishape\itshape</code>
<code>\textui</code>	<code>\DeclareTextFontCommand{\textui}{\uishape}</code>
<code>\sishape</code>	<code>\DeclareRobustCommand\sishape{\itshape\scshape}</code>
<code>\textsi</code>	<code>\DeclareTextFontCommand{\textsi}{\sishape}</code>

Table 18: Approximate replacements for etoolbox macros.

etoolbox	chronos expl3 wrapper
<code>\ifundef</code>	<code>\IfFreeTF, \IfFreeT and \IfFreeF</code>
<code>\ifdef</code>	<code>\IfExistTF, \IfExistT and \IfExistF</code>
<code>\ifcsundef</code>	<code>\IfCSFreeTF, \IfCSFreeT and \IfCSFreeF</code>
<code>\ifcsdef</code>	<code>\IfCSExistTF, \IfCSExistT and \IfCSExistF</code>
<code>\undef</code>	<code>\Undefine</code>
<code>\csletcs</code>	<code>\CSletCS</code>
<code>\cslet</code>	<code>\CSlet</code>
<code>\ifboolexpr</code>	<code>\IfBooleanExprTF, \IfBooleanExprT and \IfBooleanExprF</code>
<code>bool</code>	<code>\LegacyBoolean</code>
<code>test</code>	<code>\CSFreeBoolean</code>
<code>\ifnumcomp</code>	<code>\IntCompareBoolean, \IfIntCompareTF, \IfIntCompareT and \IfIntCompareF</code>

However, they may be incompatible with packages I'm unaware of or which are not yet published, in which case the warnings may prove informative. These macros are listed in table 18.

15.1 Compatibility with Code from T_EX SE Answers

The CTAN release of `chronos` is not backwards compatible with versions published on [T_EX StackExchange](#). However, there are several methods you can use to update most timelines produced using code from answers there. Which approach is best depends on the specific case.

I suggest four possible approaches below. Of these, methods 1 and 2 are strongly recommended. The remaining methods 3(a) and 3(b) are for those keen for adventures in the typesetting hinterlands, desperate souls suffering in imminent-deadline hells and the perilously inquisitive with too much time on their hands. They are included because most of us, at one time or another, find ourselves in situations of the second type, even if we are too home-loving and incurious to dare the others.

Method 1: If you intend to develop work utilising code from T_EX SE answers further, I strongly recommend taking the time to switch to the new key-value interface and `chronos` environment. This method is the most work, but also the most reliable and flexible. There is no guarantee that either of the alternative methods methods 3(a) and 3(b) will work or continue to work with future `chronos` releases. Method 2 is an option, but if you are actively developing a timeline, the flexibility of `chronos` should make things easier and provide options otherwise unavailable. If you put more work in and then find the code you have insufficient to your needs, you will only have delayed and expanded the task of updating.

Method 2: If you don't intend to develop existing timelines further, I strongly recommend not loading `chronos`, renaming any existing file to avoid conflicts and doing an ultra-simple update so existing documents load the renamed file. This is the simplest, most straightforward option. Why fix what ain't broke? If the code you have works and you're satisfied with the results, you need this package like a head needs an ache. The only thing you should do — and you really *should* do this — is rename any conflicting package you created locally. That is, if you've stuck code from an SE answer in a file named `chronos.sty`, I strongly recommend renaming it to, for example, `chronos-se.sty` to avoid conflicts. Then you can use `chronos` in new documents and just change the `\usepackage` invocation to `chronos-se` in old ones.

Method 3: If methods 1 and 2 aren't options — if, say, you want to use this package for a new timeline in a document with existing timelines and you don't have time to update those, then one of the following pairs of definitions *may* produce more-or-less the same output from existing or slightly modified code. Note that there is no guarantee this will work in any particular case or, if it does, that it will continue to work with future releases of `chronos`. It may, however, provide a quick-and-dirty fix if you are stuck.

(a) This requires minimal changes to existing code. You will need to modify existing timelines to use the `chronos` environment if they are currently in `tikzpicture` environments. Then place the following code *into the preamble* of your document:

```
\usepackage{chronos}
\makeatletter
% The following definitions **MUST** be in the preamble.
% They will **NOT** work if placed after \begin{document}
% or before \usepackage{chronos}.
% BEGIN \chronosevent
\NewDocumentCommand \chronosevent { 0 {} m 0 {} +m D () { \chronos@testun@yshift } }
{% #1 [<connection options>]
% #2 [<date>]
% #3 [<text tag options>]
% #4 [<text>]
% #5 (<yshift>)
  \digwyddiad{%
    date=#2,
    name=#4,
    yshift=#5,
    text tag+={#3},
    connection+={#1},
  }%
}
% END \chronosevent
% BEGIN \chronosperiod
\NewDocumentCommand \chronosperiod { 0 {} m 0 {} m 0 {} +m D () { \chronos@testun@yshift } }
{% #1 [<line options>]
% #2 [<start date>]
% #3 [<connection options>]
% #4 [<end date>]
% #5 [<text tag options>]
% #6 [<text>]
% #7 (<yshift>)
  \parhad{%
    start=#2,
    end=#4,
    name=#6,
    yshift=#7,
    connection+={#3},
    text tag+={#5},
    line+={#1},
  }%
}
% END \chronosperiod
\makeatother
```

If you use this method, you *cannot* use the key-value versions of `\chronosevent` and `\chronosperiod`. Instead, you will need to use `\digwyddiad` for events and `\parhad` for periods when you wish to make use of the new features.

(b) Alternatively, update all existing environments to use `chronos` as explained in method 3(a), if re-

quired. Then replace every occurrence of `\chronosevent` and `\chronosperiod` with `\chronoslegacyevent` and `\chronoslegacyperiod` and place the following in your document preamble⁵²:

```
\usepackage{chronos}
\makeatletter
% BEGIN \chronoslegacyevent
\NewDocumentCommand \chronoslegacyevent { 0 {} m 0 {} +m D () { \chronos@testun@yshift } }
{% #1 [<connection options>]
% #2 {<date>}
% #3 [<text tag options>]
% #4 {<text>}
% #5 (<yshift>)
\chronosevent{%
  date=#2,
  name=#4,
  yshift=#5,
  text tag+={#3},
  connection+={#1},
}%
}
% END \chronoslegacyevent
% BEGIN \chronoslegacyperiod
\NewDocumentCommand \chronoslegacyperiod { 0 {} m 0 {} m 0 {} +m D () { \chronos@testun@yshift } }
{% #1 [<line options>]
% #2 {<start date>}
% #3 [<connection options>]
% #4 {<end date>}
% #5 [<text tag options>]
% #6 {<text>}
% #7 (<yshift>)
\chronosperiod{%
  start=#2,
  end=#4,
  name=#6,
  yshift=#7,
  connection+={#3},
  text tag+={#5},
  line+={#1},
}%
}
% END \chronoslegacyperiod
\makeatother
```

This allows you to use `\chronosevent` and `\chronosperiod` with the key-value interface in new timelines.

Change History

v0.7?	General: First repo release. (Ish.) Earlier versions were published informally on TeX StackExchange 105	v0.9.3	General: Add <code>at aux</code> and <code>line add yshift</code> etc. 61
			Add <code>at aux</code> 68
			Add <code>line add yshift</code> etc. 70
v0.9	General: First CTAN release. 105	v0.9.4	General: Update key names in example. 5

⁵²The location isn't crucial in this case, provided the definitions are read before you use them and after `chronos` is loaded, but it is bad practice to define new commands in the body of documents.

Index

Features are sorted by kind. Numbers written in bold italics refer to the pages containing the main descriptions of the corresponding entries. Numbers underlined refer to the code lines where the entries are defined. Upright numbers refer to pages containing additional comments, discussion or examples of usage or to line numbers for usage in the code⁵³. † indicates an example of usage.

Symbols	
' (prime)	33, 34
'+ (prime-plus)	34
'- (prime-minus)	34
+ (plus)	32
- (minus)	8
/ (forward slash)	32
-	63
# (hash)	93
--	63
A	
arrow tips	102
B	
BOOLEAN KEYS:	
<tag>/copyleft	76
as is	67
color rotation	59
colour rotation	59
connect	68
copyleft/copyleft	67
event dates split	78
event/as is	63
event/connect	63
event/place below	63
frame	
style	54
frame	53
frame uses bb	53
life/connect	61
no simple color names	10
no simple colour names	10
period/connect	64
phantom	71
place below	69
show bounding box	98
show coords	98
show nodes	99
simple color names	
Leslie Lamport†	63
simple color names	10, 60
simple colour names	
donald knuth†	58
Leslie Lamport†	63
simple colour names	10, 60
timeline/mark at era switch	40
timeline/minor years	47
timeline/timeline arrow	95
following chronos styles	94
use in blues below†	89
timeline/timeline arrow	53
timeline/timeline bare marks	49
timeline/timeline mark eras	
effect of enabling explicitly <i>vs.</i> implicitly on	
show nodes	99
timeline/timeline mark eras	38
timeline/timeline marks	48
timeline/timeline minor marks	48
timeline/timeline show years	49
timeline/year at era switch	40
timeline/year zero	39
C	
CHOICE KEYS:	
connections on	41
lines on	41
placeholders	
levels	54
style	98
placeholders	98
timeline/border on	41
timeline/timeline on	41
timeline/timeline years	90
above	46
below	46
none	46
off line	46, 100
on line	46
timeline/timeline years	46
CHRONOS STYLES:	
blues below	
example of	1
features (summary)	18
sample output†	25
use of timeline line†	53
use of blues colour scheme	29
blues below	19
contemporary 90	
features (summary)	18
sample output†	25
suitability for temporal ranges of	90
use of contninety colour scheme	29
contemporary 90	19

⁵³I am grateful to David Carlisle and Ulrike Fischer for help with indexing at [TeX StackExchange: 695555](https://tex.stackexchange.com/questions/695555).

cronoleg			
colour rotation	7		
features (summary)	18		
in example timeline†	5		
sample output†	20		
use of cronoleg colour scheme	29		
cronoleg	17		
date centric			
development	1		
features (summary)	18		
sample output†	21		
use of title lines	75		
date centric	17		
defining custom	86, 88		
event splitter			
development	1		
features (summary)	18		
sample output†	28		
event splitter	21		
flipping blues			
features (summary)	18		
sample output†	25		
use of blues colour scheme	29		
flipping blues	19		
lavender menace			
features (summary)	18		
sample output†	22		
use of lavender colour scheme	29		
lavender menace	17		
lines on line			
development	1		
features (summary)	18		
sample output†	28		
use of conditional timeline arrow†	95		
lines on line	21		
list of	18		
loading a colour scheme	92		
modern			
features (summary)	18		
quadrupling of hashes in	93		
sample output†	22		
use of modern colour scheme	29		
use of custom colour scheme in	92		
use of public colour names in	92		
modern	17		
modifying	54		
no-year	21		
event splitter†	21		
lines on line†	21		
plain arrow†	24		
off line	19		
blues below†	19		
contemporary 90†	19		
flipping blues†	19		
off line colour†	19		
off line colour alt†	19		
off line simple†	21		
rotated 45†	21		
simple arrow†	21		
somewhat plain†	21		
off line colour			
features (summary)	18		
sample output†	26		
use of offlinebasic colour scheme	29		
off line colour	19		
off line colour alt			
features (summary)	18		
sample output†	26		
use of offlinealt colour scheme	29		
off line colour alt	19		
off line simple			
development	1		
features (summary)	18		
sample output†	27		
use of offlinebasic colour scheme	29		
off line simple	21		
on line	17		
cronoleg†	17		
date centric†	17		
lavender menace†	17		
modern†	17		
rainbow serif†	17		
serif on line†	17		
sober judge†	17		
plain arrow			
doubling of hashes in	93		
features (summary)	18		
sample output†	28		
plain arrow	24		
rainbow serif			
features (summary)	18		
sample output†	23		
use of xcolseries colour scheme	29		
rainbow serif	17		
rotated 45			
features (summary)	18		
sample output†	27		
use of default colour scheme	29		
rotated 45	21		
serif on line			
features (summary)	18		
sample output†	23		
use of default colour scheme	29		
serif on line	17		
simple arrow			
features (summary)	18		
sample output†	27		
simple arrow	21		
sober judge			
features (summary)	18		
sample output†	24		
use of sobriety colour scheme	29		

sober judge	17	colours above	59
somewhat plain		colours below	59
features (summary)	18	event/colors above	60
sample output†	28	event/colors below	60
use of default colour scheme	29	event/colours above	60
use of title lines	75	event/colours below	60
somewhat plain	21	life/colors above	60
using	29	life/colors below	60
CLASSES:		life/colours above	60
happyholidays.cls	1	life/colours below	60
COLOUR KEYS:		period/colors above	60
<tag>/default color	77	period/colors below	60
<tag>/default colour	77	period/colours above	60
background	52	period/colours below	60
background	40	COLOUR LISTS:	
color	68	should not be set by	90
colour	69, 77	when to configure	30
in assignment of colour names	58	COLOUR SCHEME KEYS:	
colour	68	background	91
event/default color	57	default above	91
event/default colour	57	default below	91
foreground	52, 57	event/above	91
foreground	40	event/below	91
info/default color	57	event/default	91
info/default colour	57	colour derivation	88
life/default color	57	foreground	91
life/default colour	57	info/default	91
period/default color	57	colour derivation	88
period/default colour	57	life/above	91
should not be set by	90	life/below	91
show bb color	99	life/default	91
show bb colour	99	colour derivation	88
show coordinate color	99	period/above	91
show coordinate colour	99	period/below	91
show node color	99	period/default	91
show node colour	99	colour derivation	88
<tag>/default colour		processing	
applying to elements	69	background	88
in assignment of colour names	58	event/default	88
setting <i>vs.</i> using	77	foreground	88
theory/default color	57	life/default	88
theory/default colour	57	period/default	88
timeline/timeline background	52	theory/default	88
timeline/timeline border inner color	51	timeline background	88
timeline/timeline border inner colour	51	timeline border inner	88
timeline/timeline border middle color	51	timeline border middle	88
timeline/timeline border middle colour		timeline border outer	88
illegitimacy of definition in chronos style†	92	timeline foreground	88
timeline/timeline border middle colour		processing (delayed)	
	51	timeline background	88
timeline/timeline border outer color	51	timeline foreground	88
timeline/timeline border outer colour	51	theory/above	91
timeline/timeline foreground	52	theory/below	91
COLOUR LIST KEYS:		theory/default	91
colors above	59	colour derivation	88
colors below	59	timeline background	91

colour derivation	88	as instance of custom†	86
timeline border inner	91	use by <code>rainbow serif</code>	18
colour derivation	88	use of colour series	86
timeline border middle	91	COLOURS:	
colour derivation	88	$\langle name \rangle$	59
timeline border outer	91	accessing	58
colour derivation	88	assigned to <code>johannes gutenbergt</code>	83
timeline foreground	91	assignment by <code>chronos</code>	58
colour derivation	88	chronos current tag color	
COLOUR SCHEMES:		outside tag contexts	59
blues	29	chronos current tag color	59
as instance of custom†	86	chronos current tag colour	
definition	86	outside tag contexts	59
use by <code>blues below</code>	18	chronos current tag colour	59
use by <code>flipping blues</code>	18	chronos main background color	40
colour names	91	chronos main background colour	40
contninetv	29	use in <code>chronos styles</code>	40
as example of minimal modification to support		chronos main color	40
chronos styles	86	chronos main colour	40
as instance of custom†	86	as tag default	57
use by <code>contemporary 90</code>	18	chronos current tag colour as equivalent to	
creating		outside tag contexts	59
options (summary)	87	use in <code>chronos styles</code>	40
cronoleg	29	chronos timeline background colour	90
as instance of custom†	86	chronos timeline border inner colour	92
implementation internal	86	chronos timeline border outer colour	92
use by <code>cronoleg</code>	18	chronos timeline foreground colour	90
default		color $\langle name \rangle$	58
modification will not cause memoize recompila-		color leslie lamport†	63
tion	84	colour $\langle name \rangle$	58
defining custom	86	colour leslie lamport†	63
lavender	29	colour name	58
as instance of custom†	86	configuration	57
use by <code>lavender menace</code>	18	core	91
list of	29	core border	91
modern	29	core derivative	91
as example of minimal modification to support		current tag	59
chronos styles	86	default	30
as instance of custom†	86	elemental	91
use by <code>modern</code>	18	in tag context	59
offlinealt	29	leslie lamport†	63
as example of minimal modification to support		names	
chronos styles	86	assigned	58
as instance of custom†	86	simple colour names	
sufficient to define deviations from $\langle existing$		disabling	10
<i>scheme</i>	86	use in <code>chronos connect</code>	83
use by <code>off line colour alt</code>	18	use in <code>chronos create chronos connector</code>	83
offlinebasic	29	use in <code>chronos create text tag connector</code>	83
as example of minimal modification to support		83
chronos styles	86	use in <code>chronos mark line</code>	83
as instance of custom†	86	use in <code>chronos text tag</code>	83
use by <code>off line colour</code>	18	use in keys	79
use by <code>off line simple</code>	18	using	58
sobriety	29	using directly	83
as instance of custom†	86	white	90
xcolseries	29	with simple colour names	10

COMMA-SEPARATED LIST KEYS:

century subheadings	56
century subheadings+	56
century subheadings'	56
chronos coords	
to add coordinates for headings, subheadings and	
century subheadings	57
chronos coords	55
headings	55
headings+	55
headings'	55
subheadings	55
subheadings+	55
subheadings'	55

CONCEPTS:

<i><chronos preamble></i>	11, 43
setting normally local keys in	67
chronos style	17
authors should never use <code>timeline config'</code>	85
colour list	59
colour assignment from	58
rotation	58
colour rotation	58
breaking	94
Donald Knuth†	7
effect of colour rotation key	59
hashes essential	94
in assignment of colour names	58
colour scheme	29
as customisation	57
load existing	24
using	24
element	12
additional	10, 12
capitalisation, preventing	67
colour to assign	68
colour assignment to	58
components of life and period	63
components of event	64
components of theory	65
components of copleft and copyright	67
components of info	66
components of main	66
components of theory circle	65
connectable	14
connection points	68
global colour configuration	77
names of colours assigned to	58
non-connectable	14
placement of coordinate <code>jikji†</code>	6
specified in <i><chronos preamble></i>	12
timeline-connectable	14
tag	12, 59
colour assignment to elements	58
coordinate names	15

custom styles	96
effect on <code>\chronosshowfeatures</code>	100
fallback colour, problems	92
global defaults for all	80
hashes essential	94
in key specifications	32
node names	15
prefix required	74
prefix, influence on configuration	74
support for connectors	9
theory circle	81
use default colour assigned to elements be-	
longing to	69
timeline	11
combining package and T _E X SE code in single	
document	104
combining package and T _E X SE code in single	
document with legacy names	104
combining package and T _E X SE code in single	
document with minimal changes	104
completed using T _E X SE code	103
connectors	14
elements, additional, connectable	14
elements, additional, non-timeline-connectable	
	14, 14
elements, additional, timeline-connectable	14, 14
updating from T _E X SE code	103
updating with retained T _E X SE code	103
<i><timeline additions specification></i>	12
<i><timeline specification></i>	11, 43, 44, 44, 70, 74

COORDINATES:

<i>(chronos origin)</i>	67
chronos origin	48
default placement of theory	65
chronos origin	48
chronos year <i>-<YYYY></i>	48
chronos year 0	48
chronos year <i><YYYY></i>	48
leslie lamport†	61
<i><name></i>	
as component of life and period	63
<i><name>1</i>	
as component of theory circle	65

D

DATE FORMAT KEYS:

<i><tag>/date format</i>	72
<i><tag>/date formats</i>	72
date format	35
event/date format	35
event/show eras/full	36
event/show eras/only years	36
event/without eras/full	36
event/without eras/only years	36
every date format	37
life/date formats	36

- | | | | |
|---------------------------------------|--------|---|--------|
| life/show eras/full | 36 | left border | 45 |
| life/show eras/only years | 37 | line add yshift | 68, 74 |
| life/without eras/full | 37 | line add yshift | 70 |
| life/without eras/only years | 37 | line add yshift+ | 70 |
| minor year format | 38 | line add yshift- | 70 |
| period/date formats | 36 | line add yshift' | 70 |
| period/show eras/full | 36 | line yshift | 70 |
| period/show eras/only years | 37 | line yshift | 74 |
| period/without eras/full | 37 | line yshift+ | 74 |
| period/without eras/only years | 37 | line yshift- | 74 |
| year format | 37 | line yshift' | 74 |
| DATE KEYS: | | | |
| birth | 70 | outer border | 45 |
| date | | right border | 45 |
| effect of event dates split on use of | 78 | sizes | 72 |
| date | 70 | subheadings drops | 45 |
| dates | | text tag yshift | 67 |
| whether to define in chronos styles | 89 | text tag yshift | 81 |
| dates | 70 | timeline | |
| death | | timeline width | 53 |
| omission for living | 7 | timeline | |
| death | 70 | timeline era margin | 94 |
| end | 70 | timeline margin | 94 |
| event/date | 63, 64 | timeline width | |
| life/birth | 61 | as total width | 42 |
| life/death | 61 | timeline/height | 43 |
| period/dates | | timeline/timeline border height | 43 |
| as mandatory for completed | 64 | timeline/timeline era margin | 44 |
| period/end | | timeline/timeline height | 100 |
| as mandatory for completed | 64 | finalised before timeline/do timeline arrow | |
| period/start | | | 95 |
| as mandatory for completed | 64 | timeline/timeline height | 43 |
| as mandatory for ongoing | 64 | timeline/timeline margin | 44 |
| start | 70 | timeline/timeline width | 96 |
| timeline/dates | 41 | adjustments for arrow tips and line caps | 94 |
| timeline/end | 42 | timeline/timeline width | 44 |
| timeline/end date | 57 | timeline/width | 44 |
| timeline/end date | 42 | top border | 45 |
| timeline/start | 42 | dimensions | 33 |
| timeline/start date | 42 | | |
| DIMENSION KEYS: | | | |
| <dimension key> | 33 | E | |
| <dimension key>+ | 33 | ELEMENTS: | |
| <dimension key>- | 33 | additional | |
| <dimension key>'+ | 33 | and colour rotation | 58 |
| <dimension key>'- | 34 | connectable | 12 |
| <dimension key>' | 33 | non-connectable | 12 |
| borders' | 45 | phantoms | 71 |
| borders'+ | 45 | primary | 14 |
| borders'- | 45 | secondary (sub-) | 15 |
| bottom border | 45 | timeline-connectable | 12 |
| headings border | 44 | bare marks | 49 |
| headings drop | 44 | placement | 12 |
| headings drops' | 45 | setting | 48 |
| headings'+ | 45 | border | 9 |
| headings'- | 45 | using to change appearance of connectors | 41 |
| | | capitalisation of name | 67 |
| | | caption | 14 |

- style 75
- century subheadings
 - ensuring required coordinates exist 57
- chronos connector 14, 41, 63, 75
 - as component of life 63
 - as component of period 63
 - as component of event 64
 - configuring global defaults 79
 - style, using directly 83
 - use of colour `leslie lampport in†` 63
- chronos coordinates
 - cf. levels 55
 - help with placement 54
- chronos tikz
 - outer border† 85
- chronos tikz outside bb
 - outer border† 85
- colour
 - cf. `every <tag>` 82
- connected element 21
- connection 64, 74
 - chronos support for 8
 - absent in phantoms 71
 - adding between text tags 85
 - adding with `chronos connect` for `johannes gutenberg†` 83
 - and colour rotation 58
 - as component of life 63
 - as component of period 63
 - as component of event 64
 - between chronos connectors and text tag connectors 14
 - between `johannes gutenberg` and other elements† 83
 - cf. `every <tag>` 82
 - configuring global defaults 79
 - connectors as facilitating connections to theories 64
 - crossing nodes 7
 - default use of `chronos main colour` in 40
 - documentation† 32
 - Donald Knuth† 58
 - effect of drawing on different layers 41
 - on `chronos middle ground layer` 19
 - reducing visual clutter 19
 - style, using directly 83
 - up and left in `jikji†` 6
 - use of `|-` 63
 - use of colour `leslie lampport in†` 63
- connector
 - chronos support for 8
 - cf. `every <tag>` 82
 - `chronos connector jikji†` 6
 - connecting Knuth and \TeX † 8
 - connection 74
 - created for Knuth† 8
 - elements which support 9
 - main connector `jikji†` 6
 - main, identifying 75
 - required keys for theory 65
 - style in `cronoleg` 9
 - tags lacking support for 9
- connectors
 - use of `name` in 67
- copyleft
 - copyleft 14
 - copyright 14
 - style 78
- copyright
 - copyleft 14
 - copyright 14
 - style 78
- date format
 - cf. `every <tag>` 82
- default colour
 - setting 69
- documentation
 - timeline 5
- era label
 - location 12
- event 14
 - connectionconditions for drawing 63
 - event years on line 46
 - introduction to 5
 - line 14
 - text tag† 6
- frame
 - adding code after 85
 - adding code after outside bounding box 85
 - adding code before 85
 - and `outer border` 45
 - and bounding box 45
 - as secondary† 14
 - `cronoleg` as not using bounding box for† 17
 - determinants of configuration 12
 - if not using bounding box 45
 - introduction to 5
- heading 55
- headings
 - adding code after 85, 85
 - adding code after outside bounding box 85
 - adding code before 85
 - determinants of configuration 12
 - ensuring required coordinates exist 57
 - introduction to 5
 - location 12
 - placement 44
 - placement relative to subheadings 45
 - purpose 9
 - style configuration 57
 - use of keys to create† 56
 - vertical lines corresponding to 98

without upper/lower subheadings	45	main connector always created	69
info	14	main title	
introduction to	5	as secondary†	14
johannes gutenberg†	83	introduction to	5
label		somewhat plain†	21
common style for upper and lower	75	style	75
labels	14	title lines	75
style	75	major steps	46, 49
layer		major year	
effect of placing elements on different	41	at era switch†	47
levels	54	marks	49
cf. chronos coordinates	55	name	48
cronoleg†	17	major years	
help with placement	54	common configuration	50
placement	44	dependent on modulo year	48
life	14	dividing with bare marks	48
connection	61	font	50
connectors	61	frequency of labelling	47
introduction to	5	labelled year modulo†	47
representation of temporal extension	74	labelled year non-modulo†	47
text tag	7, 8	labelling as prerequisite for minor year labels	47
line		millennium†	47
and colour rotation	58	non-modulo start date†	47
as component of life	63	recommended when using <code>step minor year</code>	47
as component of period	63	setting marks	48
as component of event	64	style differentiated from	48
as representation of time	14	style in common with	48
at aux	68	years modulo <i>vs.</i> non-modulo†	47
blues below†	19	marks	
cf. <code>every <tag></code>	82	adjusting chronos style defaults	17
configuring global defaults	79	effect of non-modulo year	48
default use of <code>chronos main colour</code> in	40	in <code>simple arrow†</code>	21
effect of drawing on different layers	41	in example <code>timeline†</code>	5
Fall of the Roman Empire†	64	placement	12
lines on line†	21	style for minor years	47
phantoms	71	using different styles for	47
plain arrow†	24	minor marks	
representation of time in life/period	14	placement	12
rotated 45†	21	minor steps	46
style	74	marks at	49
style, using directly	83	minor year	49
use of colour <code>leslie lampport in†</code>	63	half millennium†	47
lower subheadings		name	48
as only subheadings	45	minor years	
placement	45	common configuration	50
placement relative to upper subheadings	12	dividing with bare marks	48
main		font	50
frame	12, 14	frequency of labelling	47
main title	14	labelled year modulo†	47
main title as lacking connectors	9	labelled year non-modulo†	47
main connector		labelled only if labelling major years	47
anchor	68	non-modulo start date†	47
as component of life	63	placement	12
as component of period	63	setting minor marks	49
as component of event	64	style differentiated from	48
connection	74	style in common with	48

whether to label	47	connection points	68
years modulo <i>vs.</i> non-modulo†	47	custom style using <code>chronos keys†</code>	96
naming	67	date content in event	70
period	14	default use of <code>chronos main colour in</code>	40
introduction to	5	Donald Knuth†	7
representation of temporary extension	74	event dates split	78
text tag connector	64	font, date(s)	76
period/text tag	8	font, text	77
placement	67	holistic treatment of configuration	79
step divisions		introduction	14
common configuration	50	lines on <code>line†</code>	21
step minor year		main connector, identifying	75
attempted correction if specified without major		no style	79
years	47	plain arrow†	24
subheading	56	purpose	9
century subheadings	56	shifted right†	63
subheadings	33	sober judge†	17
adding code after	85, 85	stacking	54
adding code after outside bounding box	85	style, using directly	83
determinants of configuration	12	tag <code>gutenberg bible†</code>	83
ensuring required coordinates exist	57	tag <code>johannes gutenberg†</code>	83
introduction to	5	tag <code>movable type†</code>	83
placement	45	tag <code>printing press†</code>	83
placement relative to headings	12	text tag date formatting	37, 37
purpose	9	title lines	75
style configuration	57	use of <code>name in</code>	67
use of keys to create†	56	use of colour <code>leslie lamport in†</code>	63
without headings	45	text tag connector	14
<code><tag>/connection</code>	15	additional configuration for main	75
<code><tag>/connector</code>	15	as component of life	63
<code><tag>/line</code>	15	as component of period	63
<code><tag>/text tag</code>	15	as component of event	64
assigned colour passed to	58	configuration	75
<code>chronos connect</code>	83	configuring global defaults	80, 80
date content in event	70	creation for theories	64
date content in life/period	70	<code>johannes gutenberg†</code>	83
in <code>timeline†</code>	5	not feature of non-connectable elements	14
rotated	21	potential invisibility	65
rotated†	21	style, using directly	83
text		use of colour <code>leslie lamport in†</code>	63
date content in life/period	70	text tag connectors	63
text tag	15	theory	
absent in phantoms	71	connecting individual to multiple	9
addition of connectors in Donald Knuth†	8	introduction to	5
and colour rotation	58	theory	14
apply arbitrary <code>TikZ</code> to	74	theory circle	14
as component of info	66	create element of type	65
as component of life	63	introduction to	5
as component of period	63	lack of text tag	15
as component of theory	65	timeline	1
as component of event	64	<code><timeline additions specification></code>	12
at <code>aux</code>	68	BCE label	38
cf. <code>every <tag></code>	82	CE label	38
configuration specific to main connector	75	absence of borders in off-line	86
configuring global defaults	79	additional elements	61
connection	74	additional elements, connectable	64

- additional elements, non-connectable 65
 additional elements, timeline-connectable 61
 additional configuration 85
 anatomy 12
 as location of line 61
 as location of `leslie lampost†` 61
 bare marks 49
 borders 12
`chronos origin` dependant on era switch 48
`chronos year \chronosyeari` 92
`chronos` does not draw vertical 4
`chronos` draws horizontal 4
 colours 40, 52, 52
 colours for, derivation of 92
 colours, further processing changes 90
 colours, reason not to set in `chronos style` 92
 colours, reason to avoid hard-coding 94
 compatibility 102
 complementary `elements†` 8
 components of 12
 configuration keys 41
 configuration, `timeline line` 53
 configuration, further processing 90
 configuration, main key 41
 connections 14
 connections 15
 connections and lines 41
 connections, complex 63
 connectors 15, 63
 construction 4
 coordinates 55
 coordinates for unrepresented year 57
 coordinates, creating additional 55
 creation of complex 98
 customisation 17, 29, 35
 date, first 42
 date, last 42
 dates 41
 densely packed 54
 densely packed, non-standard paths 83
 densely packed, use of space in 69
 dimensions 42, 44, 45
 dimensions responsible for total size 12
 Donald Knuth† 7
 effect of borders on dimensions 12
 era labels 35, 38
 era margins 44
 era switch 40
 event year on line 50
 font 51
 height 43
 history of writing and printing† 5
 identifying explicit choices 100
 if no year zero 39
 independent of earlier 30
 introduction to 5
 key-value interface 105
 levels 54
 levels, creating 54
 levels, rendering visible 55
 limitations of `chronos` 4
 lines 14, 15
 major years 47
 margins 44
 marks 48
 marks and years, invisible 45
 marks, adding to style 49
 minor marks 49
 minor years 47
 placement of `event†` 6
 placement of `bi sheng†` 7
 placement of `jikji`'s connector relative to† 6
 problem of non-existent year 38
 representation of time on 46
 short (temporal duration) 47
`show nodes` 99
`skip event year on line` 50
 spanning eras 38
 split and unsplit events, combining in same
 `timeline unsupported` 78
 split and unsplit events, combining in same doc-
 ument 78
 step divisions 48
 style 52
 styles, event years on line 46
 styles, marks and years 46
 styles, marks and years, none 46
 styles, marks and years, on line 46
 styles, off line 45
 styles, on line 45
 styles, on line *vs.* off line 45
`timeline border` 53
`timeline line` 52
 total height as function of `timeline height`
 and `timeline border height` 42
 total width 44
 types drawn by `chronos` 3
 updating from T_EX SE code 103
 weird `\chronosyeari` in `chronos style` 92
 width 42
 years 47, 49
 years, modulo 47, 48
 years, not modulo 47
 years, style 48
 years, anchor 48
 years, major, format 38
`timeline border` 86, 92
 as location of line 63
 colour configuration 51, 51
 introduction to 12
`timeline line` 12

- timeline marks
 in off line colour† 19
 upper subheadings
 as only subheadings 45
 placement 45
 placement relative to headings 12
 use by chronos 67
 using assigned colour during creation 59
 whether to connect to timeline 68
 year 46, 92
 blues below† 19
 effects of configuration 100
 event year on line 50
 first *vs.* first labelled 47
 frequency of labelling 47
 labels, rotated† 19, 19
 marked at start 48
 non-modulo configuration 47
 rotate labels 48
 test for major 48
 unmarked 50
 years
 adjusting chronos style defaults 17
 at minor steps 49
 cf. bare marks 48
 chronos origin dependant on configuration
 modulo 48
 in simple arrow† 21
 in example timeline† 5
 marks for major 49
 modulo step major year and step minor year
 47
 placement 12
 set TikZ anchor 48
- ENVIRONMENTS:
- chronos
 as constructing timeline 12
 cannot be externalised with external 84
 cf. code posted on T_EX SE 4
 enable automemoization 84
 externalisation with memoize 84
 introduction to 5
 chronos 1, 11
 figure 59
 scope 83
 tikzpicture 11
 \chronosbaselineskip 55
 adding to 12
 bounding box 45
 content of (*timeline additions specification*) 12
- H**
- hash (#) 93
- K**
- KEY HANDLERS:
 .chronos key maker
 tag left† 97
 tag post† 97
 tag right† 97
 .chronos key maker 97
 for style creation 97
 .meaning to context
 use in chronos styles 96
 use in blues below† 89
- KEYS:
- <tag>/author 75
 <tag>/chronos connector 75
 <tag>/chronos connector+ 75
 <tag>/chronos connector' 75
 <tag>/connection 74
 <tag>/connection+ 74
 <tag>/connection' 74
 <tag>/date font 76
 <tag>/full dates 73
 <tag>/label 75
 <tag>/label+ 75
 <tag>/label' 75
 <tag>/line 74
 <tag>/line+ 74
 <tag>/line' 74
 <tag>/main text tag connector 75
 <tag>/main text tag connector+ 75
 <tag>/main text tag connector' 75
 <tag>/notice 76
 <tag>/only text 73
 <tag>/only years 73
 <tag>/rotate 76
 <tag>/show eras 73
 <tag>/text font 76
 <tag>/text tag 74
 <tag>/text tag connector 75
 <tag>/text tag connector+ 75
 <tag>/text tag connector' 75
 <tag>/text tag+ 74
 <tag>/text tag' 74
 <tag>/title 75
 <tag>/title+ 75
 <tag>/title' 75
 <tag>/without eras 73
 <tag>/year 76
 anchor 97
 for years 48
 used as TikZ 68
 will be overridden 48
 at
 as required for \chronosmaintitle 66
 if unset 81
 in custom style† 97
 trouble in custom styles 96

at	67	event years on line	46
at aux	68	event/chronos connector	
bce year label	37	set by every chronos connectors	80
before drawing frame	85	event/connection	
before drawing frame+	85	set by every connections	80
before drawing frame'	85	event/connectors	63
before headings	85	event/line	
before headings+	96	set by every lines	80
before headings+	85	event/main text tag connector	
before headings'	85	set by every main text tag connectors	81
caption	72	event/name	63
ce year label	37	event/text tag	
century subheading	56	set by every text tags	80
century subheading+	56	event/text tag connector	
century subheading'	56	set by every text tag connectors	80
chronos connectors		every chronos connectors	80
set by every chronos connectors	80	every chronos connectors+	80
chronos connectors	79	every chronos connectors'	80
chronos connectors+	79	every connections	80
chronos connectors'	79	every connections+	80
chronos tikz	85	every connections'	80
chronos tikz outside bb	85	every event	82
chronos tikz outside bb+	85	every event+	82
chronos tikz outside bb'	85	every event'	82
chronos tikz+	85	every info	82
chronos tikz'	85	every info+	82
circle texts	72	every info'	82
color scheme	29	every life	82
colour scheme	29	every life+	82
connections		every life'	82
set by every connections	80	every lines	80
connections	79	every lines+	80
connections+	79	every lines'	80
connections'	79	every main text tag connectors	80
connectors	68	every main text tag connectors+	80
connectors+	68	every main text tag connectors'	80
connectors'	68	every period	82
copyleft	78	every period+	82
copyleft+	78	every period'	82
copyleft/author	75	every text tag connectors	80
copyleft'	78	every text tag connectors+	80
copyright	78	every text tag connectors'	80
copyright+	78	every text tags	80
copyright/at	66	every text tags+	80
copyright/author	67	every text tags'	80
copyright'	78	every theory	82
dates content		every theory circle circle	81
effect of event dates split on use of	78	every theory circle circle+	81
dates content	71	every theory circle circle'	81
debug	99	every theory circle text	81
default color	69	every theory circle text+	81
default colour	69	every theory circle text'	81
documentation	31	every theory+	82
event	77	every theory'	82
event year on line skip	50, 70	font	
event years on line	46, 70	will be overridden	48

full dates	73	name	
heading	55	as required for \chronosmaintitle	66
heading+	55	as supporting chronos connect†	58
heading'	55	capitalisation	75
headings style	57	effect of event dates split on use of	78
headings style+	57	override for text tag content	71
headings style'	57	prevent capitalisation	67
info/at		required for phantoms	71
as required	66	use in assigned colour names	58
info/caption	66	whether to capitalise	67
info/name	66	name	67
as required	66	name content	75
info/text tag		effect of event dates split on use of	78
set by every text tags	80	if unset	75
key	31	problematic markup	67
labels	72	name content	71
levels		no color rotation	59
level 1†	61	no colour rotation	59
making visible	98	no simple color names	60
placement	44	no simple colour names	60
placement if frame not using bounding box	45	only text	73
u1†	61	only years	73
levels	54	period	77
levels at	54	period/chronos connector	
life	77	set by every chronos connectors	80
life/chronos connector		period/connection	
set by every chronos connectors	80	set by every connections	80
life/connection		period/connectors	64
set by every connections	80	period/dates content	8
life/connectors	61	period/line	
life/line		set by every lines	80
set by every lines	80	period/line+	64
life/main text tag connector		period/main text tag connector	
set by every main text tag connectors	81	set by every main text tag connectors	81
life/name	61	period/name	
life/text tag		as mandatory for ongoing	64
set by every text tags	80	period/text tag	
life/text tag connector		set by every text tags	80
set by every text tag connectors	80	period/text tag connector	
lines		set by every text tag connectors	80
set by every lines	80	place above	69
lines	79	redefinition in tag-specific contexts	97
lines+	79	rotate all colors	59
lines'	79	rotate all colours	59
main text tag connectors		rotate no colors	59
set by every main text tag connectors	81	rotate no colours	59
main text tag connectors	80	show eras	73
main text tag connectors+	80	special date	71
main text tag connectors'	80	specification	31
main/frame	54	step major year	
main/frame+	54	years, modulo	47
main/frame'	54	step minor year	
main/title	75	years, modulo	47
main/title+	75	subheading	56
main/title'	75	subheading+	56
major step font	50	subheading'	56

- subheadings style 57
- subheadings style+ 57
- subheadings style' 57
- tag anchor
 - as anchor 68
 - in custom style† 97
 - trouble in custom styles 96
- tag anchor 68
- <tag>/chronos connector 75, 79
- <tag>/chronos connector+ 79
- <tag>/chronos connector' 79
- <tag>/connection 74, 79
- <tag>/connection+ 74, 79
- <tag>/connection' 74, 79
- <tag>/line 74, 79
- <tag>/line+ 74, 79
- <tag>/line' 74, 79
- <tag>/main text tag connector .. 75, 75, 80
- <tag>/main text tag connector+ ... 75, 80
- <tag>/main text tag connector' ... 75, 80
- <tag>/tag 74, 74
- <tag>/tag+ 74, 74
- <tag>/tag' 74, 74
- <tag>/text tag 74, 81
- <tag>/text tag connector 75, 80
- <tag>/text tag connector+ 80
- <tag>/text tag connector' 80
- <tag>/text tag+ 74
- <tag>/text tag' 74
- text content
 - problematic markup 67
- text content 71
- text tag connectors
 - set by every text tag connectors ... 80
- text tag connectors 80
- text tag connectors+ 80
- text tag connectors' 80
- text tags
 - set by every text tags 80
- text tags 79
- text tags+ 79
- text tags' 79
- theory 77
- theory circle/at 65
- theory circle/name
 - as mandatory 65
- theory/at 65
- theory/chronos connector
 - set by every chronos connectors ... 80
- theory/connection
 - set by every connections 80
- theory/connectors 65
- theory/main text tag connector
 - set by every main text tag connectors 81
- theory/name 65
- theory/tag anchor 65
- theory/text tag
 - set by every text tags 80
- theory/text tag connector
 - set by every text tag connectors ... 80
- KeyFont timeline
 - line caps 102
- timeline
 - arrow tips 102
- timeline 41
- timeline bce label 38
- timeline ce label 38
- timeline config 85
- timeline config 85
- timeline config+ 85
- timeline config+ 85
- timeline config' 85
- timeline/conditional timeline arrow . 96
 - use in blues below† 89
- timeline/conditional timeline arrow . 94
- timeline/do timeline arrow 94
- timeline/do timeline arrow 95
- timeline/eras font 50
- timeline/minor step font 50
- timeline/no timeline arrow
 - following chronos styles 94
- timeline/no timeline arrow 53
- timeline/step divisions 48
- timeline/step from year
 - and non-modulo years† 47
- timeline/step from year 47
- timeline/step major year 47, 47
- timeline/step major year 47
- timeline/step major years 47
- timeline/step minor year 47, 47
- timeline/step minor year 47
- timeline/step minor years 47
- timeline/step year 47
- timeline/step year 47
- timeline/step years 47
- timeline/timeline all marks 49
- timeline/timeline bare mark 49
- timeline/timeline border 53
- timeline/timeline border+ 53
- timeline/timeline border' 53
- timeline/timeline line 52
- timeline/timeline line+ 52
- timeline/timeline line' 52
- timeline/timeline mark 49
- timeline/timeline minor mark 49
- timeline/timeline year 89
- timeline/timeline year 48
- timeline/timeline years anchor 92
- timeline/timeline years anchor 48
- without eras 73

- xshift 97
yshift 67
- L**
- LAYERS:
- adding to appropriate 5
 - background 15, 84
 - vs.* chronos background 41
 - choosing appropriate 12
 - chronos background 15, 83
 - vs.* background 41
 - chronos foreground 15, 84
 - chronos middle ground 15, 83
 - chronos middle ground layer 19
 - chronos overlay 15, 84, 84
 - connections on 41
 - control over 4
 - lines on 41
 - main 15, 41
 - timeline/border on 41
 - timeline/timeline on 41
- line caps 102
- M**
- MACROS:
- `\baselineskip` 55
 - `\bcelabel` 38, 102, 102
 - `\bceyearlabel` 37, 102, 102
 - `\celabel` 38, 102, 102
 - `\ceyearlabel` 37, 102, 102
 - `\chronosbaselineskip` 55
 - `\chronos@bce` 102, 102
 - `\chronos@borderheight` 102
 - `\chronos@ce` 102, 102
 - `\chronoscopyleft` 66, 76
 - configuration, local 67
 - name optional for 67
 - `\chronoscopyright` 66, 76
 - configuration, local 67
 - invoked by `\chronoscopyleft` 67
 - name optional for 67
 - `\chronosevent` 63
 - configuration, local 67
 - method allowing use of key-value interface in 105
 - method incompatible with key-value version of 104
 - renaming T_EX SE version 105
 - using assigned colour in 59
 - `\chronos@height` 102
 - `\chronosinfo` 66
 - configuration, local 67
 - `\chronoslegacyevent` 105
 - `\chronoslegacyperiod` 105
 - `\chronoslife` 14, 61
 - configuration, local 67
 - in example timeline† 5
 - using assigned colour in 59
 - `\chronos@llinell@yshift` 102
 - `\chronosmaintitle` 66
 - configuration, local 67
 - name optional for 67
 - `\chronosnewcolorscheme` 86
 - `\chronosnewcolourscheme` 86, 86
 - `\chronosperiod` 64
 - configuration, local 67
 - method allowing use of key-value interface in 105
 - method incompatible with key-value version of 104
 - renaming T_EX SE version 105
 - using assigned colour in 59
 - `\chronosset` 29
 - effect on `\chronoscopyright` 76
 - effect on timeline 12
 - not used† 100
 - purpose 29
 - setting normally local keys in 67
 - showing options 100
 - when (not) to use in document body ... 30
 - `\chronosset*` 30
 - `\chronosshowcolor` 99
 - `\chronosshowcolor*` 99
 - `\chronosshowcolour` 99
 - `\chronosshowcolour*` 99
 - `\chronosshowfeatures` 100, 100
 - `\chronosshowpreset` 100, 100
 - `\chronostheory` 64
 - configuration, local 67
 - using assigned colour in 59
 - `\chronostheorycircle` 65
 - configuration, local 67
 - `\chronos@width` 102
 - `\chronos@yearbce` 102, 102
 - `\chronos@yearce` 102, 102
 - `\chronosyeari` 92
 - use in blues below 89
- conditionally defined
- `\celabel†` 102
- conditionally used
- `\uishape†` 102
- `\CSFreeBoolean` 103
 - `\CSlet` 103
 - `\cslet` 103
 - `\CSletCS` 103
 - `\csletcs` 103
 - `\digwyddiad` 104
 - `\foreach` 48, 102
 - `\IfBooleanExprF` 103
 - `\IfBooleanExprT` 103
 - `\IfBooleanExprTF` 103
 - `\ifboolexpr` 103
 - `\ifcsdef` 103

- internal macro names 102, 102
 - internal macros conditionally defined . . 102
 - internal macros defined locally 102
 - memoize-ready 84
 - replacements for etoolbox macros 103
 - typesetting a timeline† 5
 - chronos-lib-colschemes 10
 - chronos-lib-styles 10, 92
 - chronosys 1
 - etoolbox 10, 103
 - as dependency 102
 - compatibility of chronos replacements . . 102
 - macros with chronos analogues 103
 - expl3 10, 29, 103
 - fp 10
 - memoize 84
 - compatibility 102
 - PGF 1
 - pgfcalendar 38
 - pgfcalendar 10
 - pgfkeys 31
 - pgfmath
 - $\langle value \rangle^+$ not parsed by 33
 - $\langle value \rangle^-$ not parsed by 34
 - $\langle value \rangle'$ not parsed by 33
 - pgfmath 33
 - $\langle value \rangle$ parsed by 33
 - $\langle value \rangle^+$ parsed by 33
 - $\langle value \rangle^-$ parsed by 34
 - svn-prov 10
 - TikZ 1
 - tikz 10
 - xcolor 10
 - xparse 10
- PGF/TIKZ LIBRARIES:
- arrows
 - compatibility 102
 - arrows.meta 10
 - compatibility 102
 - backgrounds 15
 - backgrounds 10
 - calc 10
 - decorations.text 10
 - external
 - cf. memoize 84
 - incompatibility 84
 - fit 10
 - fixedpointarithmetic 10
 - positioning 10
 - shadows 10
- PROGRAMMES:
- perl 84
 - python 84
- S**
- STYLES:
- $\langle tag \rangle$ /title lines 75
 - chronos connect 83
 - chronos create chronos connector 83
 - chronos create text tag connector 83
 - chronos mark line 83
 - chronos text tag 83
 - custom
 - tag left† 1, 96
 - tag post† 1, 96, 98
 - tag right† 1, 96
 - event date split 78
 - event year on line 50
 - matching connection 58
 - on chronos background layer 12
 - on chronos background layer 83
 - on chronos foreground layer 12
 - on chronos foreground layer 84
 - on chronos middle ground layer 83
 - on chronos overlay layer 98
 - on chronos overlay layer 84
 - placeholder lines 98
 - show coord 99
 - show coord 99
 - show coordinate 99
 - show node coord 99
 - show node coord 99
 - tag right 96
 - timeline/era switch off line 50
- T**
- TAGS:
- specific settings
 - activated by installation under /chronos . 97
 - copyleft 66, 66
 - copyright 66, 66
 - event 63
 - info 66
 - life 61
 - main 66
 - period 64
 - theory 64
 - theory circle 65
 - copyleft 78
 - as lacking connectors 9
 - at mandatory 67
 - availability of keys 66
 - components of 67
 - configuration, global 77
 - configuration, local 67
 - configuration, local/global 72
 - create element of tag type 67
 - default name 67
 - elements belonging to 14
 - options (summary) 62

- use of **name** in content of 67
- copyright 78
 - as lacking connectors 9
 - at mandatory 67
 - availability of keys 66
 - components of 67
 - configuration, global 77
 - configuration, local 67
 - configuration, local/global 72
 - create element of tag type 66
 - default name 67
 - elements belonging to 14
 - options (summary) 62
 - use of **name** in content of 67
- event
 - as connectable to other elements 82
 - as primary element 12
 - as supporting connectors 9
 - assignment of colours to elements of tag type 58
 - at optional 67
 - availability of keys 63
 - chronos connector 64
 - colour lists for colour rotation 58
 - colour rotation 58, 94
 - colour rotation (above) 60
 - colour rotation (below) 60
 - colours, using 82
 - components of 64
 - configuration, global 77, 82
 - configuration, local 67
 - configuration, local/global 72
 - connection 64
 - connectors 64
 - connectors, creating additional 68
 - create element of tag type 63
 - date 70
 - date formatting 36
 - default placement (**lines on line**) 21
 - Diamond Sutra*† 6
 - effect of colour scheme in **chronolog**† . . . 17
 - effect of **simple colour names** on 10
 - holistic treatment of configuration 79
 - Jikji*† 6
 - last position set globally 30
 - line 64
 - main connector 64
 - no style 79
 - options (summary) 62
 - plain arrow† 24
 - point connected to timeline 68
 - Publication of *Diamond Sutra*† 6
 - split text tags 78
 - split text tags, style 78
 - style for elements of type 21
 - styles, using 82
 - support for event years on line 46
- text tag 64
- text tag connector 64
 - use of **name** in content of 67
 - use of single date for placement 14
- info
 - as case of colour assignment without colour rotation 94
 - as lacking connectors 9
 - as primary element 12
 - as standalone 14
 - assignment of colours to elements of tag type 58
 - at mandatory 67
 - availability of keys 66
 - colours, using 82
 - components of 66
 - configuration, global 77, 82
 - configuration, local 67
 - configuration, local/global 72
 - create element of tag type 66
 - effect of **simple colour names** on 10
 - options (summary) 62
 - setting caption 72
 - style of caption 75
 - styles, using 82
 - use of **name** in content of 67
- life
 - as connectable to other elements 82
 - as basis for levels 5, 54
 - as example of tag context 59
 - as prefix† 32
 - as primary element 12
 - as supporting connectors 9
 - assignment of colours to elements of tag type 58
 - at optional 67
 - at aux 68
 - availability of keys 61
 - Bi Sheng† 6
 - chronos connector 63
 - colour lists for colour rotation 58
 - colour names assigned to **donald knuth**† . . 58
 - colour rotation 58
 - colour rotation (above) 60
 - colour rotation (below) 60
 - colours of the rainbow† 58
 - colours, using 82
 - components of 63
 - configuration, global 77, 82
 - configuration, local 67
 - configuration, local/global 72
 - connection 63
 - connectors 63
 - connectors, creating additional 68
 - create element of tag type 61
 - date formatting 36
 - date ranges 36
 - date specifications, equivalent 70

- dates 70
- default placement (lines on line) 21
- Donald Knuth† 7
- effect of simple colour names on 10
- highlighted by colour scheme in `chronolog†` 17
- line 63
- main connector 63
- options (summary) 62
- `plain arrow†` 24
- point connected to timeline 68
- split text tags unsupported 78
- styles, using 82
- text tag 63
- text tag connector 63
- use of name in content of 67
- use of two dates for placement 14
- main
 - `at` mandatory 67
 - availability of keys 66
 - components of main title 66
 - configuration, global 77
 - configuration, local 67
 - configuration, local/global 72
 - default name for main title 67
 - elements belonging to 14
 - no associated list of properties 100
 - options (summary) 62
 - style for main title 75
 - use of name in content of 67
- period
 - as connectable to other elements 82
 - as primary element 12
 - as supporting connectors 9
 - assignment of colours to elements of tag type 58
 - `at` optional 67
 - `at aux` 68
 - availability of keys 64
 - `chronos` connector 63
 - colour lists for colour rotation 58
 - colour rotation 58
 - colour rotation (above) 60
 - colour rotation (below) 60
 - colours, using 82
 - components of 63
 - configuration, global 77, 82
 - configuration, local 67
 - configuration, local/global 72
 - connection 63
 - connectors 63
 - connectors, creating additional 68
 - create element of tag type 64
 - date formatting 36
 - date ranges 36
 - date specifications, equivalent 70
 - dates 70
 - default placement (lines on line) 21
 - effect of colour scheme in `chronolog†` 17
 - effect of simple colour names on 10
 - last position set globally 30
 - line 63
 - main connector 63
 - mandatory keys for completed 64
 - mandatory keys for ongoing 64
 - options (summary) 62
 - `plain arrow†` 24
 - point connected to timeline 68
 - representation on timeline 64
 - split text tags unsupported 78
 - styles, using 82
 - text tag 63
 - text tag connector 63
 - use of name in content of 67
 - use of two dates for placement 14
 - `WoOdBlOcK pRiNtInG†` 8
 - `Woodblock Printing†` 8
- theory
 - `TEX†` 8
 - as connectable 64
 - as connectable to other elements 82
 - as primary element 12
 - as supporting connectors 9
 - assignment of colours to elements of tag type 58
 - `at` optional 67
 - availability of keys 65
 - cf. non-connectable elements 14
 - colour rotation 58
 - colours, using 82
 - components of 65
 - configuration, global 77, 82
 - configuration, local 67
 - configuration, local/global 72
 - connecting multiple people to 9
 - connectors, creating additional 68
 - create element of tag type 65
 - `cronolog` 7
 - default placement 65
 - effect of simple colour names on 10
 - `metafont†` 58
 - options (summary) 62
 - styles, using 82
 - text tags dateless 67
 - use of name in content of 67
 - using default colour lists as tag-specific .. 59
- theory circle
 - as lacking connectors 9
 - as primary element 12
 - as standalone 14
 - `at` mandatory 67
 - availability of keys 65
 - common style for labels 75
 - components of 65
 - configuration, global 77

configuration, local	67
configuration, local/global	72
configuring base ring	81
non-use of name in	67
options (summary)	62
slowness	14