

The TikZ-Extensions Package
Manual for version 0.2
<https://github.com/Qrrbrbirlbel/tikz-extensions>

Qrrbrbirlbel

August 17, 2022

Contents

I	Introduction	4
1	Usage	4
2	Why do we need it?	4
3	Should these libraries be part of TikZ?	4
II	TikZ Libraries	5
4	Calendar	6
4.1	Extensions	6
4.2	Week numbering (ISO 8601)	6
5	Arc to a point	8
6	More Horizontal and Vertical Lines	10
6.1	Zig-Zag	10
6.2	Zig-Zig	12

7	Extending the Path Timers	14
7.1	Rectangle	14
7.2	Parabola	14
7.3	Sine/Cosine	15
8	Using Images as a Pattern	16
9	Arcs through Three Points	17
10	Mirror, Mirror on the Wall	19
10.1	Using the “Spiegelungsmatrix”	19
10.2	Using built-in transformations	20
III	PGF Libraries	22
11	Transformations: Mirroring	23
11.1	Using the “Spiegelungsmatrix”	23
11.2	Using built-in transformations	23
IV	Miscellaneous	26
12	Calendar: Weeknumbers and more conditionals	27
12.1	Extensions	27
12.2	Week numbering (ISO 8601)	27
13	And a little bit more	28
13.1	PGFmath	28
13.1.1	Postfix operator R	28
13.1.2	Functions	28
13.1.3	Functions: using coordinates	29
13.2	PGFkeys	29
13.2.1	Conditionals	29
13.2.2	Handlers	30
13.3	PGFfor	31
V	Index	32

Part I

Introduction

1 Usage

This package is called `tikz-ext`, however, one can't load it via `\usepackage`. Instead, this package consists of multiple PGF and *TikZ* libraries which are loaded by either `\usepgflibrary` or `\usetikzlibrary`.

2 Why do we need it?

Since I have been answering questions on TeX.sx I've noticed that some questions come up again and again, every time with a slightly different approach on how to

solve them.

I don't like reinventing the wheel which is why I've gathered the code of my answers in this package.

And, yes, I am using them myself, too.

3 Should these libraries be part of *TikZ*?

I guess.

Part II

TikZ Libraries

These libraries only work with TikZ.



4 Calendar

TikZ Library `calendar-plus`

```
\usetikzlibrary{calendar-plus} % LATEX and plain TEX
\usetikzlibrary[calendar-plus] % ConTEXt
```

This library extends the TikZ library `calendar`.

4.1 Extensions

The values of following keys were stored in some macros that are not accessible by the user. These are now simple value-keys. The @-protected macros are still available, of course.

`/tikz/day xshift` (initially 3ex)

`/tikz/day yshift` (initially 3.5ex)

`/tikz/month xshift` (initially 9ex)

`/tikz/month yshift` (initially 9ex)

It is now also possible to nest `/tikz/if` occurrences.

`/tikz/if=(<conditions>)(<code or options>)else(<else code or options>)` (no default)

4.2 Week numbering (ISO 8601)

`/tikz/week code=<code>` (no default)

Works like `/tikz/day code` or `/tikz/month code`, only for weeks.

`/tikz/week text=<text>` (no default)

Works like `/tikz/day text` or `/tikz/month text`, only for weeks.

`/tikz/every week` (style, no value)

Works like `/tikz/every day` or `/tikz/every month`, only for weeks.

`/tikz/week label left` (style, no value)

Places the week label to the left of the first day of the month. (For `week list` and `month list` where a week does not start on a Monday, the position is chosen “as if” the week had started on a Monday – which is usually exactly what you want.)

July												
26	1 2 3											
27	4	5	6	7	8	9	10					
28	11	12	13	14	15	16	17					
29	18	19	20	21	22	23	24					
30	25	26	27	28	29	30	31					

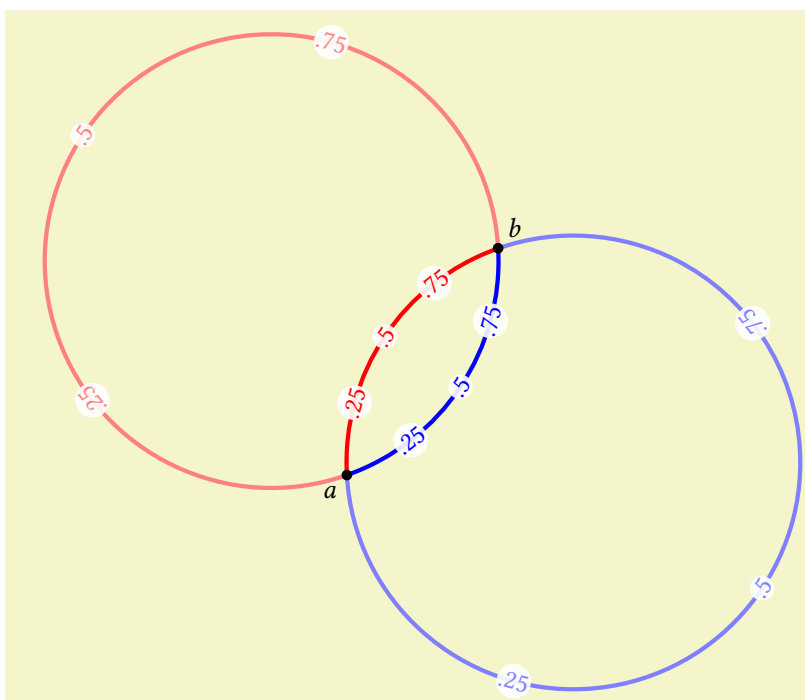
```
\usetikzlibrary {calendar-plus}
\tikz
\calendar [week list, month label above centered,
            dates=2022-07-01 to 2022-07-31,
            week label left,
            every week/.append style={gray!50!black,font=\sfamily}];
```

5 Arc to a point

TikZ Library `paths.arcto`

```
\usetikzlibrary{paths.arcto} % LATEX and plain TEX
\usetikzlibrary[paths.arcto] % ConTEXt
```

This library adds new path specifications `arc to` that specifies an arc *to* a point – without the user having to specify any angles.



```
\usetikzlibrary {paths.arcto}
\begin{tikzpicture}[ultra thick,dot/.style={label={#1}}]
\coordinate[dot=below left:$a$] (a) at (0,0);
\coordinate[dot=above right:$b$] (b) at (2,3);
\begin{scope}[
  radius=3,
  nodes={
    shape=circle,
    fill=white,
    fill opacity=.9,
    text opacity=1,
    inner sep=+0pt,
    sloped,
    allow upside down
  }]
\draw[blue] (a) arc to[]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\draw[red] (a) arc to[clockwise]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\draw[blue!50] (a) arc to[large]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\draw[red!50] (a) arc to[large, clockwise]
  node[near start] {.25} node {.5} node[near end] {.75} (b);
\end{scope}

\fill[radius=2pt] (a) circle[] (b) circle[];
\end{tikzpicture}
```

`\path ... arc to[options](coordinate or cycle) ...;`

When this operation is used, the path gets extended by an arc that goes through the current point and *(coordinate)*.

For two points there exist two circles or four arcs that go through or connect these two points. Which one of these is constructed is determined by the following options that can be used inside of *(options)*.

`/tikz/arc to/clockwise`

(style, no value)

This constructs an arc that goes clockwise.

`/tikz/arc to/counter clockwise`

(style, no value)

This constructs an arc that goes counter clockwise.

This is the default.

`/tikz/arc to/large`

(style, no value)

This constructs an arc whose angle is larger than 180° .

`/tikz/arc to/small`

(style, no value)

This constructs an arc whose angle is smaller than 180° .

`/tikz/arc to/rotate= $\langle degree \rangle$`

(no default)

Rotates the arc by $\langle degree \rangle$. This only takes effect when x radius and y radius is different.

`/tikz/arc to/x radius= $\langle value \rangle$`

(no default)

This forwards the $\langle value \rangle$ to `/tikz/x radius`.

`/tikz/arc to/y radius= $\langle value \rangle$`

(no default)

This forwards the $\langle value \rangle$ to `/tikz/y radius`.

`/tikz/arc to/radius= $\langle value \rangle$`

(no default)

This forwards the $\langle value \rangle$ to both `/tikz/x radius` and `/tikz/y radius`.

`/tikz/every arc to`

(style, no value)

After `/tikz/every arc to` this will also be applied before any $\langle options \rangle$ are set.

It should be noted that this uses `\pgfpatharcto` where the TikZ manual warns of:

The internal computations necessary for this command are numerically very unstable. In particular, the arc will not always really end at the $\langle target coordinate \rangle$, but may be off by up to several points. A more precise positioning is currently infeasible due to \TeX 's numerical weaknesses. The only case it works quite nicely is when the resulting angle is a multiple of 90° .

The `arc to path` operation will also work only in the canvas coordinate system. The lengths of the vectors $(1, 0)$ and $(0, 1)$ will be used for the calculation of the radii but no further consideration is done.

6 More Horizontal and Vertical Lines

TikZ Library `paths.ortho`

```
\usetikzlibrary{paths.ortho} % LATEX and plain TEX  
\usetikzlibrary[paths.ortho] % ConTEXt
```

This library adds new path specifications `| - |`, `- | -` as well as `r-ud`, `r-du`, `r-lr` and `r-rl`.

6.1 Zig-Zag

Similar to the path operations `| -` and `- |` this library adds the path operations `| - |` and `- | -`.

```
\path ... | - | [<options>] <coordinate or cycle> ...;
```

This operation means “first vertical, then horizontal and then vertical again”.

```
\path ... - | - [<options>] <coordinate or cycle> ...;
```

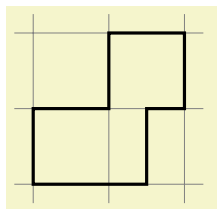
This operation means “first horizontal, then vertical and then horizontal again”.

`/tikz/hvvh/ratio=<ratio>`

(no default, initially 0.5)

This sets the ratio for the middle part of the Zig-Zag connection.

For values $\langle ratio \rangle < 0$ and $\langle ratio \rangle > 1$ the Zig-Zag lines will look more like Zig-Zig lines.



```
\usetikzlibrary {paths.ortho}  
\begin{tikzpicture}[very thick]  
\draw[help lines] (-.25, -1.25) grid (2.25, 1.25);  
\draw (0, 0) -| - (2, 1) --  
      (2, 0) -| -[ratio=.25] (0,-1) -- cycle;  
\end{tikzpicture}
```

`/tikz/hvvh/distance=<distance>`

(no default)

This sets the distance between the start point and the middle part of the Zig-Zag connection.

For values $\langle distance \rangle < 0$ the distance will be used for the target coordinate.

If $\langle number \rangle = 0$ then

- $pos = -1$ will be at the start,
- $pos = 2$ will be at the end,
- $pos = 0$ will be at the first kink,
- $pos = 1$ will be at the second kink and
- $pos = .5$ will still be in the middle of the middle part of the connection.

`/tikz/hvvh/middle 0 to 1`

(no value)

This is an alias for $spacing = 0$.

6.2 Zig-Zig

`\path ... r-ud[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “first up, then horizontal and then down”.

`/tikz/udlr/ud distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the horizontal line to $\langle length \rangle$.

`\path ... r-du[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “first down, then horizontal and then up”.

`/tikz/udlr/du distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the horizontal line to $\langle length \rangle$.

`\path ... r-lr[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “left down, then vertical and then right”.

`/tikz/udlr/lr distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the vertical line to $\langle length \rangle$.

`\path ... r-rl[$\langle options \rangle$] $\langle coordinate or cycle \rangle$...;`

This operation means “first right, then vertical and then down”.

`/tikz/udlr/rl distance= $\langle length \rangle$`

(no default, initially .5cm)

This sets the distance between the start and the vertical line to $\langle length \rangle$.

All distances can be set with on key.

`/tikz/udlr/distance=<length>`

(no default)

Sets all distances in the `/tikz/udlr` namespace.

`/tikz/udlr/from center=<true or false>`

(no default, initially false, default true)

This is an alias for `/tikz/hvvh/from center`.

7 Extending the Path Timers

TikZ Library `paths.timer`

```
\usetikzlibrary{paths.timer} % LATEX and plain TEX
\usetikzlibrary[paths.timer] % ConTEXt
```

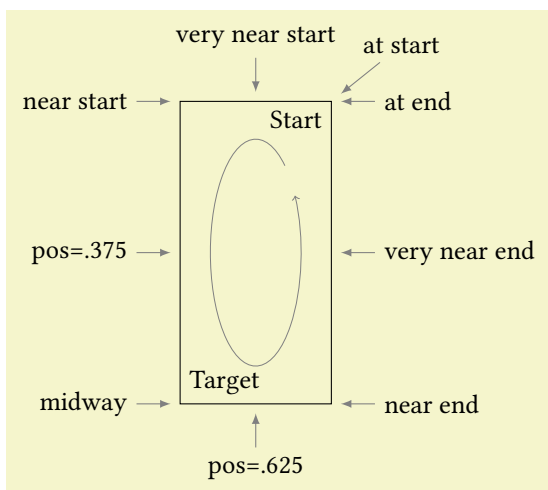
This library adds timers to the path specifications `rectangle`, `parabola`, `sin` and `cos`.

In TikZ, the path specification `rectangle`, `parabola`, `sin` and `cos` do not provide their own timer, i.e. a node placing algorithm that is dependent on the actual path. For `rectangle` the timer of the straight line between the rectangle's corners is used, for the other paths, nodes, coordinates, pics, etc. are placed on the last coordinate.

This library allows this.

7.1 Rectangle

For the `rectangle` path operator, the timer starts with `pos = 0` (= at start) from the starting coordinate in a counter-clockwise direction along the rectangle. The corners will be at positions 0.0, 0.25, 0.5, 0.75 and 1.0.



```
\usetikzlibrary {paths.timer}
\begin{tikzpicture}[scale=2, every pin edge/.style={latex-, gray}]
\coordinate [label=above right:Target] (A) at (0,0);
\coordinate [label=below left:Start] (B) at (1,2);
\draw[->, help lines] ([shift=(50:.3 and .75)] .5,1)
  arc[start angle=50, delta angle=340, x radius=.3, y radius=.75];
\draw (B) rectangle (A)
  foreach \pos/\ang in {at start/60, very near start/90, near start/180, pos=.375/180,
    midway/180, pos=.625/270, near end/0, very near end/0, at end/0}{
    node[pin=\ang:\pos, style/.expanded=\pos]{};
  }
\end{tikzpicture}
```

7.2 Parabola

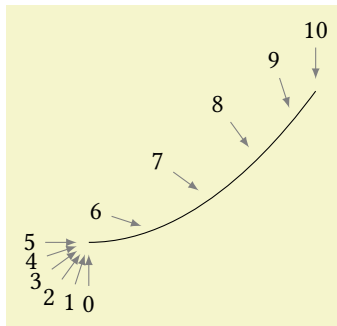
For the `parabola` path operator the timer is similar to the `.. controls ..` operator.

The position 0.5 will lie at the bend.



```
\usetikzlibrary {paths.timer}
\begin{tikzpicture}
\draw[help lines] (-2.25, -1.25) grid (2.25, 3.25);
\draw (2, -1) parabola bend (0, 0) (-1, 3);
\draw[ultra thick] (-2, -1) parabola bend (0, 0) (1, 3)
  foreach \pos in {1, ..., 4, 6, 7, ..., 9}{
    node[
      pos=. \pos, sloped, fill=white, font=\small, inner sep=+0pt
    ] {\pos}
  };
\end{tikzpicture}
```

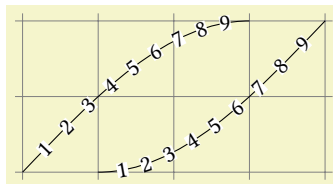
If no bend is specified half the positions will collapse into one end of the curve.



```
\usetikzlibrary {paths.timer}
\begin{tikzpicture}[every pin edge/.style={latex-, shorten <=1pt, gray}]
\draw (-2, -2) parabola (1, 0)
  foreach \pos in {0, 1, ..., 10} {
    node [pos=\pos/10, pin={\anchor=-18*\pos+90-18*\pos+270:\pos}]{}
  };
\end{tikzpicture}
```

7.3 Sine/Cosine

The sin and cos path operators also allow placing of nodes along their paths.



```
\usetikzlibrary {paths.timer}
\begin{tikzpicture}[mark nodes on line/.style={insert path={
  foreach \pos in {1, ..., 9} {node[
    sloped, fill=white, font=\small, inner sep=+0pt, pos=\pos/10] {\pos}}}}]
\draw[help lines] (-2.1, -2.1) grid (2.1, 0.1);
\draw (-2, -2) sin (1, 0) [mark nodes on line];
\draw[shift=(0:1)] (-2, -2) cos (1, 0) [mark nodes on line];
\end{tikzpicture}
```

8 Using Images as a Pattern

TikZ Library `patterns.images`

```
\usetikzlibrary{patterns.images} % LATEX and plain TEX
\usetikzlibrary[patterns.images] % ConTEXt
```

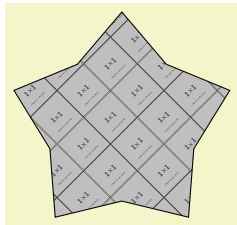
This library allows to use an image to be used as a repeating pattern for a path.

With this library arbitrary images (or indeed PDF documents) can be used as a repeating pattern for the background of a path. This is a two-step process:

1. Declaring an image as an “image-pattern”.
2. Using the “image-pattern”.

`\pgfsetupimageaspattern[⟨options⟩]{⟨name⟩}{⟨image⟩}`

`/tikz/image as pattern=⟨options⟩` (default `{}`)



```
\usetikzlibrary {patterns.images}
\pgfsetupimageaspattern[width=.5cm]{grid}{example-image-1x1}
\tikz \node[star, minimum size=3cm, draw,
  image as pattern={name=grid,options={left, bottom, y=-.5cm, rotate=45}}] {};
```

`/tikz/image as pattern/name=⟨name⟩` (no default)

Specifies the name of the “image-pattern” to be used.

`/tikz/image as pattern/option` (style, no value)

Options that’s be used by the internal `\pgftext`, only keys from `/pgf/text` should be used.

`/tikz/image as pattern/options=⟨style⟩` (style, no default)

Appends style `/tikz/image as pattern/option`.

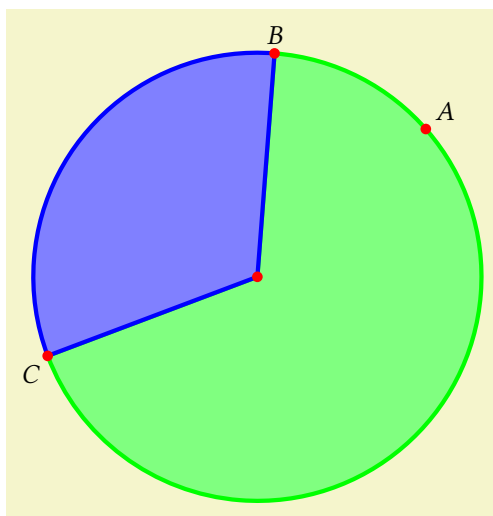
9 Arcs through Three Points

TikZ Library `topaths.arcthrough`

`\usetikzlibrary{topaths.arcthrough}` % \LaTeX and plain \TeX

`\usetikzlibrary[topaths.arcthrough]` % Con \TeX t

This library allows to use an arc defined by three points.



```
\usetikzlibrary {topaths.arcthrough}
\begin{tikzpicture}
\coordinate[label=above right:$A$] (A) at ( 3, 1);
\coordinate[label=above:$B$] (B) at ( 1, 2);
\coordinate[label=below left:$C$] (C) at (-2,-2);

\draw[ultra thick, draw=green, fill=green!50]
(B) to[arc through={clockwise,(A)}] (C)
-- (arc through center) -- cycle;
\draw[ultra thick, draw=blue, fill=blue!50]
(B) to[arc through=(A)] (C)
-- (arc through center) -- cycle;

\foreach \p in {A,B,C, arc through center} \fill[red] (\p) circle[radius=2pt];
\end{tikzpicture}
```

This can only be used for circles in the canvas coordinate system.

`/tikz/arc through/through=<coordinate>`

(no default, initially (0,0))

The coordinate on the circle that defines – together with the starting and target point – a circle.

`/tikz/arc through/center suffix=<suffix>`

(no default, initially)

The arc through will define a coordinate named `arc through center<suffix>` so that it can be referenced later.

`/tikz/arc through/clockwise`

(no value)

The resulting arc will go clockwise from the starting point to the target point. This will not necessarily go through the through point.

`/tikz/arc through/counter clockwise`

(no value)

The resulting arc will go counter clockwise from the starting point to the target point. This will not necessarily go through the through point.

`/tikz/arc through=⟨key-value⟩`

(no default)

This key should be used with `to` or `edge`. A parameter other than `center` suffix, `clockwise` or `counter clockwise` will be assumed to be the `through` coordinate.

10 Mirror, Mirror on the Wall

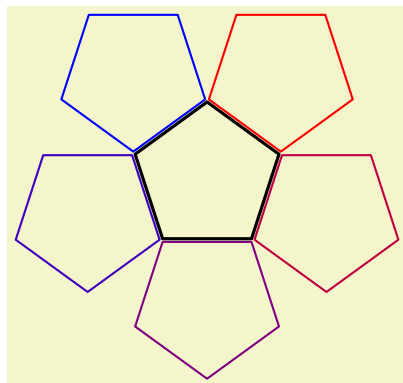
TikZ Library `transformations.mirror`

```
\usetikzlibrary{transformations.mirror} % LATEX and plain TEX
\usetikzlibrary[transformations.mirror] % ConTEXt
```

This library adds more transformations to TikZ.

As explained in section 11, there are two approaches to setting a mirror transformation. As with the commands in PGF, we'll be using lowercase `m` for the “Spiegelungsmatrix” and uppercase `M` for the built-in approach.

10.1 Using the “Spiegelungsmatrix”

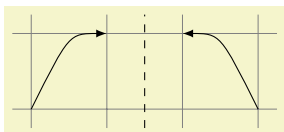


```
\usetikzlibrary {shapes.geometric,transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node [mirror=(a.corner \i)--(a.side \i), transform shape,
    reg poly=5, minimum size=+2cm, draw=red!\col!blue] {};
\end{tikzpicture}
```

`/tikz/xmirror=<value or coordinate>`

(no default)

Sets up a transformation that mirrors along a horizontal line that goes through point $(\langle value \rangle, 0)$ or $\langle coordinate \rangle$.



```
\usetikzlibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);
\draw[xmirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/ymirror=<value or coordinate>`

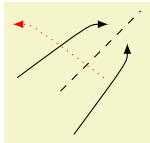
(no default)

Sets up a transformation that mirrors along a vertical line that goes through point $(0, \langle value \rangle)$ or $\langle coordinate \rangle$.

`/tikz/mirror x=<coordinate>`

(no default)

Similar to `/tikz/xmirror`, this however uses the xyz coordinate system instead of the canvas system.



```
\usetikzlibrary {transformations.mirror}
\begin{tikzpicture}[x=.5cm, y=(45:1cm)]

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);

\draw[ xmirror=(m), -latex, red, dotted] (0,0) .. controls (.5,1) .. (1,1);
\draw[mirror x=(m), -latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

`/tikz/mirror y=<coordinate>`

(no default)

Similar to `/tikz/ymirror`, this however uses the xyz coordinate system instead of the canvas system.

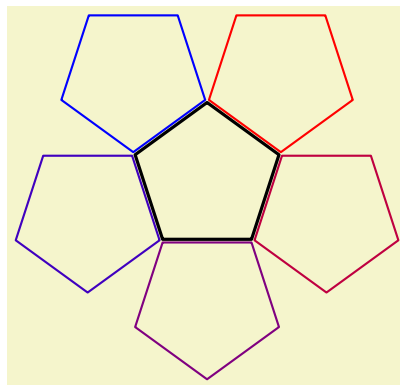
`/tikz/mirror=<point A>--<point B>`

(no default)

Sets up a transformation that mirrors along a line that goes through $\langle point A \rangle$ and $\langle point B \rangle$.

When only $\langle point A \rangle$ is given that line goes through $\langle point A \rangle$ and the origin.

10.2 Using built-in transformations

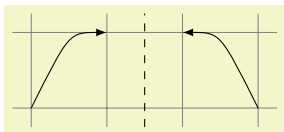


```
\usetikzlibrary {shapes.geometric,transformations.mirror}
\begin{tikzpicture}[line join=round, thick, reg poly/.style={
  shape=regular polygon, regular polygon sides={#1}}]
\node[reg poly=5, minimum size=+2cm, draw, very thick] (a) {};
\foreach \i[evaluate={\col=(\i-1)/.04}] in {1,...,5}
  \node [Mirror=(a.corner \i)--(a.side \i), transform shape,
    reg poly=5, minimum size=+2cm, draw=red!\col!blue] {};
\end{tikzpicture}
```

/tikz/xMirror= $\langle value \text{ or coordinate} \rangle$

(no default)

Sets up a transformation that mirrors along a horizontal line that goes through point $(\langle value \rangle, 0)$ or $\langle coordinate \rangle$.



```
\usetikzlibrary {transformations.mirror}  
\begin{tikzpicture}  
\draw[help lines] (-0.25, -.25) grid (3.25, 1.25);  
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);  
  
\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);  
\draw[xMirror=(m),-latex] (0,0) .. controls (.5,1) .. (1,1);  
\end{tikzpicture}
```

/tikz/yMirror= $\langle value \text{ or coordinate} \rangle$

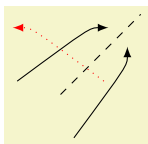
(no default)

Sets up a transformation that mirrors along a vertical line that goes through point $(0, \langle value \rangle)$ or $\langle coordinate \rangle$.

/tikz/Mirror x= $\langle coordinate \rangle$

(no default)

Similar to **/tikz/xMirror**, this however uses the xyz coordinate system instead of the canvas system.



```
\usetikzlibrary {transformations.mirror}  
\begin{tikzpicture}[x=.5cm, y=(45:1cm)]  
  
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);  
  
\draw[dashed] (1.5, -.25) coordinate (m) -- (1.5, 1.25);  
  
\draw[xMirror=(m), -latex, red, dotted] (0,0) .. controls (.5,1) .. (1,1);  
\draw[Mirror x=(m), -latex] (0,0) .. controls (.5,1) .. (1,1);  
\end{tikzpicture}
```

/tikz/Mirror y= $\langle coordinate \rangle$

(no default)

Similar to **/tikz/yMirror**, this however uses the xyz coordinate system instead of the canvas system.

/tikz/Mirror= $\langle point A \rangle$ - $\langle point B \rangle$

(no default)

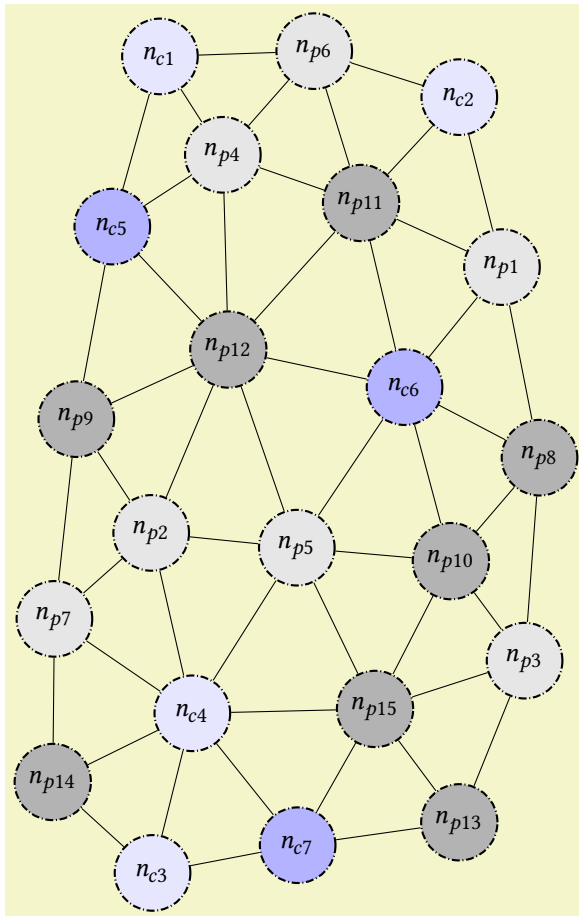
Sets up a transformation that mirrors along a line that goes through $\langle point A \rangle$ and $\langle point B \rangle$.

When only $\langle point A \rangle$ is given that line goes through $\langle point A \rangle$ and the origin.

Part III

PGF Libraries

These libraries (should) work with both PGF and TikZ.



```
\usetikzlibrary {graphs,graphdrawing} \usegdlibrary {force}
\tikzset{
  mynode/.style={
    circle, minimum size=10mm, draw, densely dashdotted, thick,
    decide color/.expand once=#1,
    decide color/.style 2 args={
      /utils/TeX/if=c#1
      {/utils/TeX/ifnum={#2<5}{blue!light}{blue!dark}}
      {/utils/TeX/ifnum={#2<8}{light}{dark}}},
    light/.style={fill=gray!20}, blue!light/.style={fill=blue!10},
    dark/.style={fill=gray!60}, blue!dark/.style={fill=blue!30}}
\tikz\graph[
  spring electrical layout, vertical=c2 to p13,
  node distance=1.5cm, typeset=$n_{\tikzgraphnodetext}$,
  nodes={mynode=\tikzgraphnodetext}] {
  % outer ring
  c2 -- {p1, p11, p6};
  p1 -- {p8, c6, p11};
  p8 -- {p3, p10, c6};
  p3 -- {p13, p15, p10};
  p13 -- {p15, c7};
  c7 -- {c3, c4, p15};
  c3 -- {p14, c4};
  p14 -- {p7, c4};
  p7 -- {p9, p2, c4};
  p9 -- {c5, p12, p2};
  c5 -- {c1, p4, p12};
  c1 -- {p6, p4};
  p6 -- {p11, p4};
  % inner ring
  p11 -- {c6, p12, p4};
  p5 -- {c6 -- {p10, p12}, p10 -- p15, p15 -- c4, c4 -- p2, p2 -- p12, p12 -- p4};
};
```

11 Transformations: Mirroring

TikZ Library `transformations.mirror`

```
\usepgflibrary{transformations.mirror} % LATEX and plain TEX and pure pgf
\usepgflibrary[transformations.mirror] % ConTEXt and pure pgf
\usetikzlibrary{transformations.mirror} % LATEX and plain TEX when using TikZ
\usetikzlibrary[transformations.mirror] % ConTEXt when using TikZ
```

This library adds mirror transformations to PGF.

Two approaches to mirror transformation exist:

1. Using the “Spiegelmatrix” (see left column).

This depends on `\pgfpointnormalised` which involves the sine and the cosine functions of PGFmath.

2. Using built-in transformations (see right column).

This depends on `\pgfmathanglebetween` which involves the arctangent (`atan2`) function of PGFmath.

Which one is better? I don’t know. Choose one you’re comfortable with.

11.1 Using the “Spiegelungsmatrix”

The following commands use the “Spiegelungsmatrix” that sets the transformation matrix following

$$A = \frac{1}{\|\vec{l}\|^2} \begin{bmatrix} l_x^2 - l_y^2 & 2l_x l_y \\ 2l_x l_y & l_y^2 - l_x^2 \end{bmatrix}.$$

`\pgftransformxmirror{⟨value⟩}`

Sets up a transformation that mirrors along a vertical line that goes through point $(⟨value⟩, 0)$.

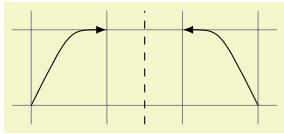
11.2 Using built-in transformations

The following commands use a combination of shifting, rotating, -1 scaling, rotating back and shifting back to reach the mirror transformation.

The commands are named the same as on the left side, only the `m` in `mirror` is capitalized.

`\pgftransformxMirror{⟨value⟩}`

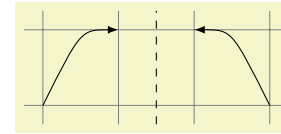
Sets up a transformation that mirrors along a vertical line that goes through point $(⟨value⟩, 0)$.



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -
.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) -- (1.5, 1.25);
\pgftransformxmirror{1.5}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-0.25, -
.25) grid (3.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (1.5, -.25) -- (1.5, 1.25);
\pgftransformxMirror{1.5}

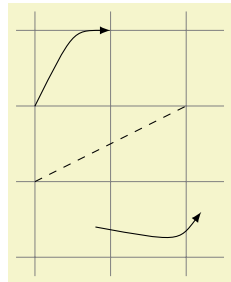
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

\pgftransformymirror{ $\langle value \rangle$ }

Sets up a transformation that mirrors along a horizontal line that goes through point $(0, \langle value \rangle)$.

\pgftransformmirror{ $\langle point A \rangle$ }{ $\langle point B \rangle$ }

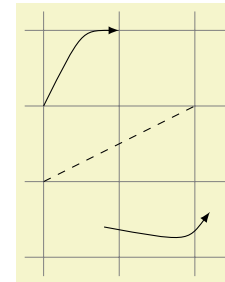
Sets up a transformation that mirrors along the line that goes through $\langle point A \rangle$ and $\langle point B \rangle$.



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -2.25) grid (2.5, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, -1) -- (2, 0);
\pgftransformmirror{\pgfpointxy{0}{-1}}
{\pgfpointxy{2}{0}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -2.25) grid (2.5, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, -1) -- (2, 0);
\pgftransformMirror{\pgfpointxy{0}{-1}}
{\pgfpointxy{2}{0}}

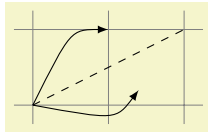
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

\pgfqtransformmirror{ $\langle point A \rangle$ }

Sets up a transformation that mirrors along the line that goes through the origin and $\langle point A \rangle$.

\pgfqtransformMirror{ $\langle point A \rangle$ }

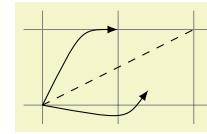
Sets up a transformation that mirrors along the line that goes through the origin and $\langle point A \rangle$.



```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -.25) grid (2.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, 0) -- (2, 1);
\pgfqtransformmirror{\pgfpointxy{2}{1}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```



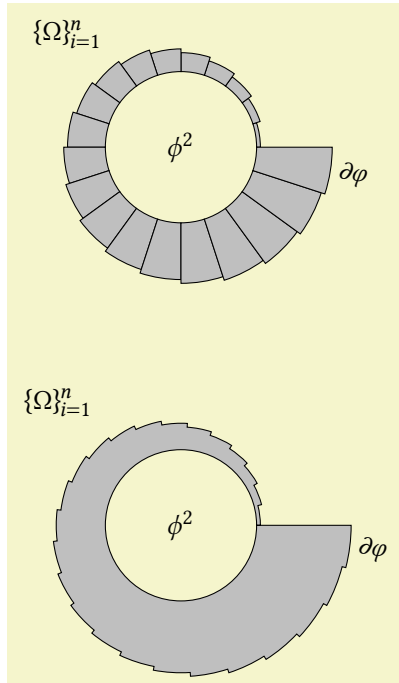
```
\usepgflibrary {transformations.mirror}
\begin{tikzpicture}
\draw[help lines] (-.25, -.25) grid (2.25, 1.25);
\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);

\draw[dashed] (0, 0) -- (2, 1);
\pgfqtransformMirror{\pgfpointxy{2}{1}}

\draw[-latex] (0,0) .. controls (.5,1) .. (1,1);
\end{tikzpicture}
```

Part IV

Miscellaneous



```
\usetikzlibrary {misc}
\begin{tikzpicture}[
  declare function={bigR(\n)=smallR+.05*\n;},
  declare constant={smallR=1; segments=20;},
  full arc=segments]
\foreach \iN[evaluate={\endRadius=bigR(\iN+1);}, use int=0 to segments-1]
\filldraw[fill=gray!50] (\iN R:\endRadius)
  arc [radius=\endRadius, start angle=\iN R, delta angle=+IR] -- (\iN R+1R:smallR)
  arc [radius=smallR, end angle=\iN R, delta angle=-IR] -- cycle;

\node                                {${\phi}^2$};
\node at (north west:{sqrt 2 * bigR(segments/2)}) {${\Omega}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR segments) {${\partial} \varphi$};

\tikzset{yshift=-.5cm, declare constant={segments=25;}, full arc=segments}
\filldraw[fill=gray!50] (right:smallR)
  \foreach \iN[evaluate={\endRadius=bigR(\iN+1);}, use int=0 to segments-1] {
    -- (\iN R:\endRadius) arc[radius=\endRadius, start angle=\iN R, delta angle=IR]}
    -- (right:smallR) arc[radius=smallR, start angle=0, delta angle=-360];

\node                                {${\phi}^2$};
\node at (north west:{sqrt 2 * bigR(segments/2)}) {${\Omega}_{i=1}^n$};
\node[rotate=-.5R, right] at (-.5R: bigR segments) {${\partial} \varphi$};
\end{tikzpicture}
```

12 Calendar: Weeknumbers and more conditionals

```
\usepackage{calendar-ext} % LATEX
\input calendar-ext.tex   % plain TEX
\usemodule[calendar-ext] % ConTEXt
```

This package adds week numbers and more conditionals to the PGF package pgfcalendar.

This package extends the pgfcalendar package.

12.1 Extensions

The following tests are added.

- **Jan** This test is passed by all dates that are in the month of January.
- **Feb** as above.
- **Mar** as above.
- **Apr** as above.
- **May** as above.
- **Jun** as above.
- **Jul** as above.
- **Aug** as above.
- **Sep** as above.
- **Oct** as above.
- **Nov** as above.
- **Dec** as above.
- **leap year**= $\langle year \rangle$ This test checks whether the given year is a leap year. If $\langle year \rangle$ is omitted, it checks the year of the current date.
- **and**= $\{\langle tests \rangle\}$ This test passes when all $\langle tests \rangle$ pass.
- **not**= $\{\langle tests \rangle\}$ This test passes when $\langle tests \rangle$ do not pass.
- **yesterday**= $\{\langle tests \rangle\}$ This test passes when the previous day passes $\langle tests \rangle$.

- **week**= $\langle num \rangle$ This test passes when the current week of the year equals $\{\langle num \rangle\}$.

The shorthands for d- and m- are slightly changed so that they are expandable. This makes it possible to use these shorthands inside of PGFmath. The shorthands for the week (see section 12.2) are added. These are

- n- (shortest numerical representation),
- n= (shortest but added horizontal space) and
- n0 (leading zero when below 10).

12.2 Week numbering (ISO 8601)

\pgfcalendarjulianyeartoweek $\{\langle julian\ day \rangle\}\{\langle year \rangle\}\{\langle week\ counter \rangle\}$

This command calculates the week for the $\langle julian\ day \rangle$ of $\langle year \rangle$. The $\langle week\ counter \rangle$ must be a T_EX counter.

The calculation follows the rule of ISO 8601 where the first week has that year's first Thursday in it.

Inside of \pgfcalendar the command \pgfcalendarcurrentweek will be available.

\pgfcalendarcurrentweek

This command returns the current week number (always two digits – use shorthand n. to strip the leading zero).

Inside of \ifdate the command \pgfcalendarifdateweek will be available.

\pgfcalendarifdateweek

This command returns the week number (always two digits).

13 And a little bit more

TikZ Library misc

```
\usetikzlibrary{misc} % LATEX and plain TEX
\usetikzlibrary[misc] % ConTEXt
```

This library adds miscellaneous utilities to PGFmath, PGF or TikZ.

13.1 PGFmath

13.1.1 Postfix operator R

Similar to `\segments[<num>]` in PSTricks, the postfix operator `R` allows the user to use an arbitrary number of segments of a circle to be used instead of an angle.

`/tikz/full arc=<num>` (default)

The number `<num>` of segments will be set up. Using `full arc` with an empty value disables the segmentation and `1R` equals 1° .

The given value `<num>` is evaluated when the key is used and doesn't change when `<num>` contains variables that change.

The `R` operator can then be used.

`xR` (postfix operator; uses the `fullarc` function)

Multiplies x with $\frac{360}{\langle num \rangle}$.

13.1.2 Functions

`strrepeat("Text", x)`

`\pgfmathstrrepeat{"Text"}{x}`

Returns a string with `Text` repeated x times.

```
foofoofoofoofoo \pgfmathparse{strrepeat("foo", 5)}
\pgfmathresult
```

`isInString("String", "Text")`

`\pgfmathisInString{"String"}{"Text"}`

Returns 1 (true) if `Text` contains `String`, otherwise 0 (false).

0 and 1

```
\pgfmathparse{isInString("foo", "bar")}
\pgfmathresult \ and\
\pgfmathparse{isInString("foo", "foobar")}
\pgfmathresult
```

`strcat("Text A", "Text B", ...)`

`\pgfmathstrcat{"Text A"}{"Text B"}{...}`

Returns the concatenation of all given parameters.

blue!21!green

```
\pgfmathparse{strcat("blue!", int(7*3), "!green")}
\pgfmathresult
```

`isEmpty("Text")`

`\pgfmathisEmpty{"Text"}`

Returns 1 (true) if `Text` is empty, otherwise 0 (false).

0 and 1 and 1

```
\pgfmathparse{isEmpty("foo")} \pgfmathresult\ and\
\pgfmathparse{isEmpty("")} \pgfmathresult\ and\
\def\emptyText{}
\pgfmathparse{isEmpty("\emptyText")} \pgfmathresult
```

`atanXY(x,y)`

`\pgfmathatanXY{x}{y}`

Arctangent of $y \div x$ in degrees. This also takes into account the quadrant. This is just a argument-swapped version of `atan2` which makes it easier to use the `\p` commands of the `calc` library.

53.13011

```
\pgfmathparse{atanXY(3,4)} \pgfmathresult
```

atanYX(y, x)

`\pgfmathatanYX{y}{x}`

Arctangent of $y \div x$ in degrees. This also takes into account the quadrant.

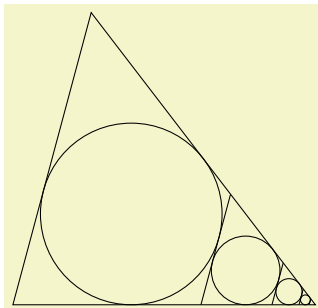
53.13011 `\pgfmathparse{atanYX(4,3)} \pgfmathresult`

13.1.3 Functions: using coordinates

The following functions can only be used with PGF and/or TikZ. Since the arguments are usually plain text (and not numbers) one has to wrap them in `"`.

anglebetween(`"p1"`, `"p2"`)

`\pgfmathanglebetween{"p1"}{"p2"}`



```
\usetikzlibrary {calc,misc,through}
\begin{tikzpicture}
\path (0,0) coordinate (A) + (0:4) coordinate (B) +(75:4) coordinate (C);
\draw (A) -- (B) -- (C) -- cycle;
\foreach \cnt in {1,...,4}{
  \pgfmathsetmacro\triA{distancebetween("B","C")}
  \pgfmathsetmacro\triB{distancebetween("C","A")}
  \pgfmathsetmacro\triC{distancebetween("A","B")}
  \path (barycentric cs:A=\triA,B=\triB,C=\triC) coordinate (M)
    node [draw, circle through=($(A)!(M)!(C)$)] (M) {};
  \draw ($(C)-(A)$) coordinate (vecB)
    (M.75-90) coordinate (@)
    (intersection of @--[shift=(vecB)]@ and B--C) coordinate (C) --
    (intersection of @--[shift=(vecB)]@ and B--A) coordinate (A);}
\end{tikzpicture}
```

Return the angle between the centers of the nodes `p1` and `p2`.

qanglebetween(`"p"`)

`\pgfmathqanglebetween{"p"}`

Return the angle between the origin and the center of the node `p`.

distancebetween(`"p1"`, `"p2"`)

`\pgfmathdistancebetween{"p1"}{"p2"}`

Return the distance (in pt) between the centers of the nodes `p1` and `p2`.

qdistancebetween(`"p"`)

`\pgfmathqdistancebetween{"p"}`

Return the distance (in pt) between the origin and the center of the node `p`.

13.2 PGFkeys

13.2.1 Conditionals

`/utils/if=<cond><true><false>`

(no default)

This key checks the conditional `<cond>` and applies the styles `<true>` if `<cond>` is true, otherwise `<false>`. `<cond>` can be anything that PGFmath understands.

As a side effect on how PGFkeys parses argument, the `<false>` argument is actu-

ally optional.

The following keys use TeX' macros `\if`, `\ifx`, `\ifnum` and `\ifdim` for faster executions.

`/utils/TeX/if=<token A><token B><true><false>`

(no default)

This key checks via `\if` if $\langle token A \rangle$ matches $\langle token B \rangle$ and applies the styles $\langle true \rangle$ if it does, otherwise $\langle false \rangle$.

As a side effect on how PGFkeys parses argument, the $\langle false \rangle$ argument is actually optional.

`/utils/TeX/iffx= $\langle token A \rangle \langle token B \rangle \langle true \rangle \langle false \rangle$` (no default)

As above.

`/utils/TeX/ifnum= $\langle num cond \rangle \langle true \rangle$
opt $\langle false \rangle$` (no default)

This key checks `\ifnum $\langle num cond \rangle$` and applies the styles $\langle true \rangle$ if true, otherwise $\langle false \rangle$. A delimiting `\relax` will be inserted after $\langle num cond \rangle$.

As a side effect on how PGFkeys parses argument, the $\langle false \rangle$ argument is actually optional.

`/utils/TeX/ifdim= $\langle dim cond \rangle \langle true \rangle \langle false \rangle$` (no default)

As above.

`/utils/TeX/isempty= $\langle Text \rangle \langle true \rangle \langle false \rangle$` (no default)

This checks whether $\langle Text \rangle$ is empty and applies styles $\langle true \rangle$ if true, otherwise $\langle false \rangle$.

13.2.2 Handlers

While already a lot of values given to keys are evaluated by PGFmath at some point, not all of them are.

Key handler $\langle key \rangle /.pgfmath= $\langle eval \rangle$$

This handler evaluates $\langle eval \rangle$ before it is handed to the key.

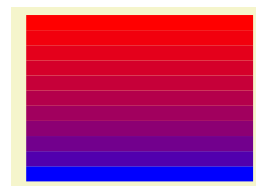
Key handler $\langle key \rangle /.pgfmath int= $\langle eval \rangle$$

As above but truncates the result.

Key handler $\langle key \rangle /.pgfmath strcat= $\langle eval \rangle$$

As above but uses the `strcat` function.

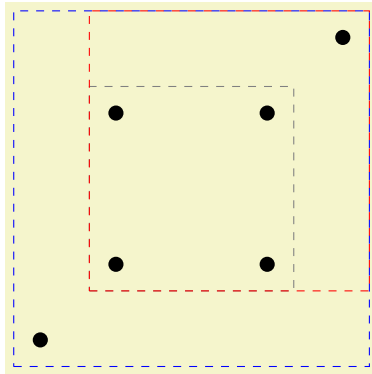
In the example below, one could have used the `/pgf/foreach/evaluate` key from `\foreach`.



```
\usetikzlibrary {misc}
\tikz\foreach \i in {0,10,...,100}
\draw[
  line width=+.2cm,
  color/.pgfmath strcat={"red!",sqrt(\i)*10,"!blue"}
]
(0,\i/50) -- +(right:3);
```

Key handler $\langle key \rangle /.List= $\langle e1 \rangle, \langle e2 \rangle, ..., \langle en \rangle$$

This handler evaluates the given list with `\foreach` and concatenates the element and the result is then given to the used key.



```
\usetikzlibrary {fit,misc}
\begin{tikzpicture}[nodes={draw, dashed, inner sep=+10pt}]
\foreach \point [count=\cnt] in {(0,0), (0,2), (2,0), (2,2), (3,3), (-1,-1)}
\fill \point circle[radius=.1] coordinate (point-\cnt);
\node[gray, fit/.List={(point-1),(point-...),(point-4)}] {};
\node[red, fit/.List={(point-1),(point-...),(point-5)}] {};
\node[blue, fit/.List={(point-1),(point-...),(point-6)}] {};
\end{tikzpicture}
```

13.3 PGFfor

Instead of `\foreach \var in {start, start + delta, ..., end}` one can use `\foreach \var[use int=start to end step delta]`.

`/pgf/foreach/use int=<start>to<end>step<delta>`

(no default)

The values $\langle start \rangle$, $\langle end \rangle$ and $\langle delta \rangle$ are evaluated by PGFmath at initialization. The part step $\langle delta \rangle$ is optional ($\langle delta \rangle = 1$).

`/pgf/foreach/use float=<start>to<end>step<delta>`

(no default)

Same as above, however the results are not truncated.

Part V
Index

Index

This index only contains automatically generated entries. A good index should also contain carefully selected keywords. This index is not a good index.

- | - | path operation, 10
- | - path operation, 10
- cos path operation, 15
- parabola path operation, 14
- sin path operation, 15

- and date test, 27
- anglebetween math function, 29
- Apr date test, 27
- arc through key, 18
- arc to path operation, 8
- atan2 math function, 28
- atanXY math function, 28
- atanYX math function, 29
- Aug date test, 27

- calendar-ext package, 27
- calendar-plus library, 6
- center suffix key, 17
- clockwise key, 8, 17
- counter clockwise key, 9, 17

- Date tests
 - and, 27
 - Apr, 27
 - Aug, 27
 - Dec, 27
 - Feb, 27
 - Jan, 27
 - Jul, 27
 - Jun, 27
 - leap year, 27
 - Mar, 27
 - May, 27
 - not, 27
 - Nov, 27

- Oct, 27
- Sep, 27
- week, 27
- yesterday, 27
- day xshift key, 6
- day yshift key, 6
- Dec date test, 27
- distance key, 10, 13
- distancebetween math function, 29
- du distance key, 12

- every arc to key, 9
- every week key, 6

- Feb date test, 27
- from center key, 11, 13
- full arc key, 28

- if key, 6, 29
- ifdim key, 30
- ifempty key, 30
- ifnum key, 30
- ifx key, 30
- image as pattern key, 16
- isEmpty math function, 28
- isInString math function, 28

- Jan date test, 27
- Jul date test, 27
- Jun date test, 27

- Key handlers
 - .List, 30
 - .pgfmath, 30
 - .pgfmath int, 30
 - .pgfmath strcat, 30

- large key, 9
- leap year date test, 27
- Libraries
 - calendar-plus, 6
 - misc, 28
 - paths.arcto, 8
 - paths.ortho, 10
 - paths.timer, 14
 - patterns.images, 16
 - topaths.arctthrough, 17
 - transformations.mirror, 19, 23
- .List handler, 30
- lr distance key, 12
- Mar date test, 27
- Math functions
 - anglebetween, 29
 - atan2, 28
 - atanXY, 28
 - atanYX, 29
 - distancebetween, 29
 - isEmpty, 28
 - isInString, 28
 - qanglebetween, 29
 - qdistancebetween, 29
 - strcat, 28
 - strrepeat, 28
- Math operators
 - R, 28
- May date test, 27
- middle 0 to 1 key, 12
- Mirror key, 21
- mirror key, 20
- Mirror x key, 21
- mirror x key, 20
- Mirror y key, 21
- mirror y key, 20
- misc library, 28
- month xshift key, 6
- month yshift key, 6

- name key, 16
- not date test, 27
- Nov date test, 27
- Oct date test, 27
- option key, 16
- options key, 16
- Packages and files
 - calendar-ext, 27
- Path operations
 - | - |, 10
 - | -, 10
 - cos, 15
 - parabola, 14
 - sin, 15
 - arc to, 8
 - r-du, 12
 - r-lr, 12
 - r-rl, 12
 - r-ud, 12
 - rectangle, 14
- paths.arcto library, 8
- paths.ortho library, 10
- paths.timer library, 14
- patterns.images library, 16
- /pgf/
 - foreach/
 - use float, 31
 - use int, 31
- \pgfcalendarcurrentweek, 27
- \pgfcalendarifdateweek, 27
- \pgfcalendarjulianyeartoweek, 27
- .pgfmath handler, 30
- .pgfmath int handler, 30
- .pgfmath strcat handler, 30
- \pgfmathanglebetween, 29
- \pgfmathatanXY, 28
- \pgfmathatanYX, 29
- \pgfmathdistancebetween, 29
- \pgfmathisEmpty, 28

- \pgfmathisInString, 28
- \pgfmathqanglebetween, 29
- \pgfmathqdistancebetween, 29
- \pgfmathstrcat, 28
- \pgfmathstrrepeat, 28
- \pgfqtransformMirror, 24
- \pgfqtransformmirror, 24
- \pgfsetupimageaspattern, 16
- \pgftransformMirror, 24
- \pgftransformmirror, 24
- \pgftransformxMirror, 23
- \pgftransformxmirror, 23
- \pgftransformyMirror, 24
- \pgftransformymirror, 24
- qanglebetween math function, 29
- qdistancebetween math function, 29
- R postfix math operator, 28
- r-du path operation, 12
- r-lr path operation, 12
- r-rl path operation, 12
- r-ud path operation, 12
- radius key, 9
- ratio key, 10
- rectangle path operation, 14
- rl distance key, 12
- rotate key, 9
- Sep date test, 27
- small key, 9
- spacing key, 11
- strcat math function, 28
- strrepeat math function, 28
- through key, 17
- /tikz/
 - arc through/
 - center suffix, 17
 - clockwise, 17
 - counter clockwise, 17
 - through, 17

- arc through, 18
- arc to/
 - clockwise, 8
 - counter clockwise, 9
 - large, 9
 - radius, 9
 - rotate, 9
 - small, 9
 - x radius, 9
 - y radius, 9
- day xshift, 6
- day yshift, 6
- every arc to, 9
- every week, 6
- full arc, 28
- hvvh/
 - distance, 10
 - from center, 11
 - middle 0 to 1, 12
 - ratio, 10
 - spacing, 11
- if, 6
- image as pattern/
 - name, 16
 - option, 16
 - options, 16
- image as pattern, 16
- Mirror, 21
- mirror, 20
- Mirror x, 21
- mirror x, 20
- Mirror y, 21
- mirror y, 20
- month xshift, 6
- month yshift, 6
- udlr/
 - distance, 13
 - du distance, 12
 - from center, 13
 - lr distance, 12
 - rl distance, 12

- ud distance, 12
- week code, 6
- week label left, 6
- week text, 6
- xMirror, 21
- xmirror, 19
- yMirror, 21
- ymirror, 20
- topaths.arctthrough library, 17
- transformations.mirror library, 19, 23

- ud distance key, 12
- use float key, 31
- use int key, 31
- /utils/
 - if, 29
 - TeX/
 - if, 29
 - ifdim, 30
 - ifempty, 30
 - ifnum, 30
 - ifx, 30

- week date test, 27
- week code key, 6
- week label left key, 6
- week text key, 6

- x radius key, 9
- xMirror key, 21
- xmirror key, 19

- y radius key, 9
- yesterday date test, 27
- yMirror key, 21
- ymirror key, 20