Reference Manual

Generated by Doxygen 1.5.6

Sun Nov 21 20:37:30 2010

Contents

Chapter 1

Chart::Base

Basic Class of Chart from which all the other classes are derived.

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

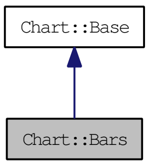
Chapter 5

Class Documentation

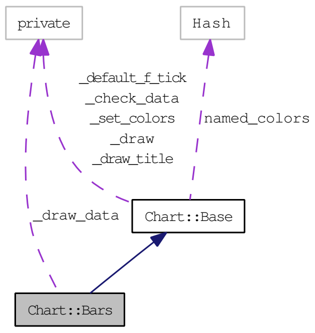
5.1 Chart::Bars Class Reference

Bars class derived from class Base.

Inheritance diagram for Chart::Bars:



Collaboration diagram for Chart::Bars:



Private Functions

- private _draw_data

  finally get around to plotting the data for (vertical) bars

5.1.1 Detailed Description

Bars class derived from class Base.

This class provides all functions which are specific to vertical bars

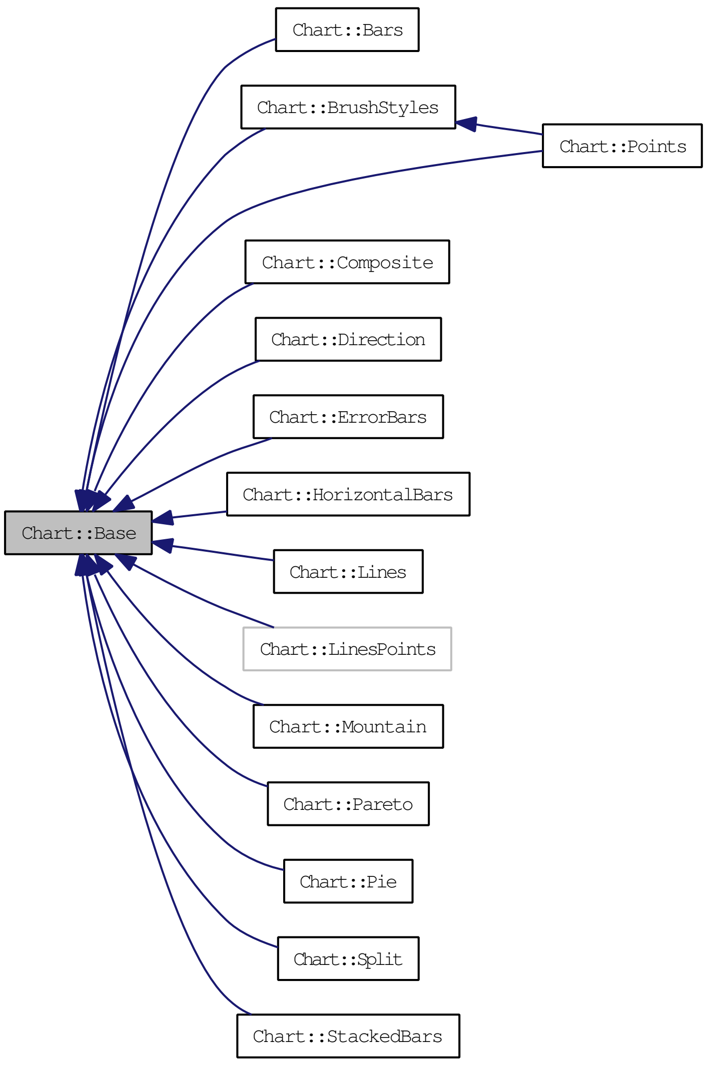
The documentation for this class was generated from the following file:

- Chart/Bars.pm

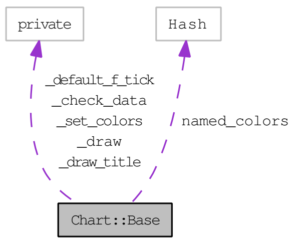
5.2 Chart::Base Class Reference

Base class for Chart; all other classes derived from here.

Inheritance diagram for Chart::Base:



Collaboration diagram for Chart::Base:



Private Functions

- private int _check_data

    Check the internal data to be displayed.

- private int _draw

    Plot the chart to the gd object
    Calls:.

- private int _set_colors

    specify my colors

- private int _draw_title

    draw the title for the chart

- private int _default_f_tick

    default tick conversion function This function is pointed to be $self->{f_x_tick} resp.

- private int _init (scalar x, scalar y)

    Initialize all default options here.

- private int _copy_data (scalar extern_ref)

    Copy external data via a reference to internal memory.

- private int _color_role_to_index (\list list_of_roles)
- private array _color_spec_to_rgb (scalar role, scalar spec)

    Return an array (list of) rgb values for spec.

- private int _draw_sub_title ()

    draw the sub-title for the chart

- private int _sort_data ()

    sort the data nicely (mostly for the pareto charts and xy-plots)

- private int _find_x_scale ()

For a xy-plot do the same for the x values, as '_find_y_scale' does for the y values!

- private int _find_y_scale ()

  find good values for the minimum and maximum y-value on the chart

- private _calcTickInterval (scalar dataset_min, scalar dataset_max, scalar flag_fixed_min, scalar flag_fixed_max, scalar minTicks, scalar maxTicks)

  Calculate the Interval between ticks in y direction.

- private int _calcXTickInterval (scalar min, scalar max, scalar minF, scalar maxF, scalar minTicks, scalar maxTicks)

  Calculate the Interval between ticks in x direction.

- private int _countTicks (scalar min, scalar max, scalar interval)

  Works out how many ticks would be displayed at that interval.

- private int _round2Tick (scalar input, scalar interval, scalar roundUP)

  Rounds up or down to the next tick of interval size.

- private array _sepFP (scalar num)

  Seperates a number into it's base 10 floating point exponent & mantisa.

- private array _find_y_range ()

  Find minimum and maximum value of y data sets.

- private array _find_x_range ()

  Find minimum and maximum value of x data sets.

- private int _plot ()

  main sub that controls all the plotting of the actual chart

- private int _draw_legend ()

  let the user know what all the pretty colors mean.

- private int _draw_bottom_legend ()

  put the legend on the bottom of the chart

- private int _draw_right_legend ()

  put the legend on the right of the chart

- private int _draw_top_legend ()

  put the legend on top of the chart

- private int _draw_left_legend ()

  put the legend on the left of the chart

- private int _draw_none_legend ()

  no legend to draw.

- private int _draw_x_label ()

  draw the label for the x-axis

- private int _draw_y_label ()

  draw the label for the y-axis

- private int _draw_ticks ()

  draw the ticks and tick labels

- private int _draw_x_number_ticks ()

  draw the ticks and tick labels

- private int _draw_x_ticks ()

  draw the x-ticks and their labels

- private int _draw_y_ticks ()

  draw the y-ticks and their labels

- private int _grey_background ()

  put a grey background on the plot of the data itself

- private int _draw_grid_lines ()

  draw grid_lines

- private int _draw_x_grid_lines ()

  draw grid_lines for x

- private int _draw_y_grid_lines ()

  draw grid_lines for y

- private int _draw_y2_grid_lines ()

  draw grid_lines for y

- private int _prepare_brush (scalar color, scalar type, scalar typeStyle)

  draw grid_lines for y

Public Class Methods

- object new ()

  Standard normal constructor.
  Calls.

Public Object Methods

- int set (hash opts)

  Set all options.

- hash getopts ()

  get all options

- int add_pt (list data)

  Graph API
  Add one dataset (as a list) to the dataref.

- add_pt (\list data)

  Graph API
  Add one dataset (as a reference to a list) to the dataref via.

- retval **add_pt** ()
- int add_dataset (list data)

  Graph API
  Add many datasets (implemented as a list) to the dataref,.

- int add_dataset (\list data)

  Graph API
  Add many datasets (implemented as a references to alist) to the dataref,.

Public Functions

- int add_datafile (scalar filename, scalar format)

  Graph API
  it's also possible to add a complete datafile
  Uses.

- int clear_data ()

  Clear Graph API (by undefining 'dataref'.

- arrayref get_data ()

  Get array of data of the last graph.

- int png (scalar file, scalar dataref)

  Produce the graph of options set in png format.

- int cgi_png (scalar dataref)

  Produce the graph of options set in png format to be directly written for CGI.

- int scalar_png (scalar dataref)

  Produce the graph of options set in png format to be directly written for CGI.

- int jpeg (scalar file, scalar dataref)

  Produce the graph of options set in JPG format to be directly.
- int cgi_jpeg (scalar dataref)

  Produce the graph of options set in JPG format to be directly for CGI.
- int scalar_jpeg (scalar dataref)

  Produce the graph of options set in JPG format to be directly.
- int make_gd (scalar dataref)

  Produce the graph of options set in GD format to be directly.
- imagemap_dump ()

  get the information to turn the chart into an imagemap
- minimum (list array)

  determine minimum of an array of values
- maximum (list array)

  determine maximum of an array of values
- arccos (scalar a)

  Function arccos(a).
- arcsin (scalar a)

  Function arcsin(a).
- true (scalar b)

  determine true value of argument
- false (scalar b)

  determine false value of argument

Public Attributes

- Hash named_colors

  RGB values of named colors.

5.2.1 Detailed Description

Base class for Chart; all other classes derived from here.

Base class from which all other classes are derived. This class provides all functions which are common for all classes

5.2.2 Member Function Documentation

5.2.2.1 object Chart::Base::new ()

Standard normal constructor.

Calls.

Returns:

A new object.

See also:

_init

5.2.2.2 int Chart::Base::set (hash *opts*)

Set all options.

Parameters:

← ***%opts*** Hash of options to the Chart

Returns:

ok or croak

main method for customizing the chart, lets users specify values for different parameters

The options are saved locally to be able to output them via

See also:

getopts()

Reimplemented in Chart::Composite, and Chart::Direction.

5.2.2.3 hash Chart::Base::getopts ()

get all options

Returns:

hash of all set options so far

Return the set options as a hash

5.2.2.4 int Chart::Base::add_pt (list *data*)

Graph API

Add one dataset (as a list) to the dataref.

Parameters:

@data Dataset to add

5.2.2.5 Chart::Base::add_pt (\list *data*)

Graph API

Add one dataset (as a reference to a list) to the dataref via.

```
for ( 0 .. $data )
{
   push  $self->{'dataref'}->[$_] }, $data[$_];
}
```

Parameters:

\@data Dataset to add

5.2.2.6 int Chart::Base::add_dataset (list *data*)

Graph API

Add many datasets (implemented as a list) to the dataref,.

Parameters:

@data Dataset (list) to add

Reimplemented in Chart::Direction.

5.2.2.7 int Chart::Base::add_dataset (\list *data*)

Graph API

Add many datasets (implemented as a references to alist) to the dataref,.

Parameters:

\@data Dataset (reference to a list) to add

5.2.2.8 int Chart::Base::add_datafile (scalar *filename*, scalar *format*)

Graph API

it's also possible to add a complete datafile

Uses.

See also:

add_pt
add_dataset

Parameters:

← ***$filename*** Name of file which contents is to be added

← ***$format*** 'pt' or 'set' to distiguish between function add_pt() in case of 'pt' or function add_dataset() in case of 'set'

5.2.2.9 int Chart::Base::clear_data ()

Clear Graph API (by undefining 'dataref'.

Returns:

Status of function

5.2.2.10 arrayref Chart::Base::get_data ()

Get array of data of the last graph.

Returns:

Reference to data set of the last graph

5.2.2.11 int Chart::Base::png (scalar *file*, scalar *dataref*)

Produce the graph of options set in png format.

called after the options are set, this method invokes all my private methods to actually draw the chart and plot the data

See also:

_set_colors
_copy_data
_check_data
_draw

Parameters:

← ***$file*** Name of file to write graph to

← ***$dataref*** Reference to external data space

Returns:

Status of the plot

5.2.2.12 int Chart::Base::cgi_png (scalar *dataref*)

Produce the graph of options set in png format to be directly written for CGI.

called after the options are set, this method invokes all my private methods to actually draw the chart and plot the data

Parameters:

$dataref

Returns:

Status of the plot

5.2.2.13 int Chart::Base::scalar_png (scalar *dataref*)

Produce the graph of options set in png format to be directly written for CGI.

Called after the options are set,

this method invokes all my private methods to actually draw the chart and plot the data

Parameters:

$dataref Reference to the data to be plotted

Returns:

Status of the plot

5.2.2.14 int Chart::Base::jpeg (scalar *file*, scalar *dataref*)

Produce the graph of options set in JPG format to be directly.

called after the options are set, this method invokes all my private methods to actually draw the chart and plot the data

Parameters:

$file Name of file to write graph into

$dataref

Returns:

Status of the plot

5.2.2.15 int Chart::Base::cgi_jpeg (scalar *dataref*)

Produce the graph of options set in JPG format to be directly for CGI.

called after the options are set, this method invokes all my private methods to actually draw the chart and plot the data

Parameters:

$dataref

Returns:

Status of the plot

5.2.2.16 int Chart::Base::scalar_jpeg (scalar *dataref*)

Produce the graph of options set in JPG format to be directly.

called after the options are set, this method invokes all my private methods to actually draw the chart and plot the data

Parameters:

$dataref

Returns:

Status of the plot

5.2.2.17 int Chart::Base::make_gd (scalar *dataref*)

Produce the graph of options set in GD format to be directly.

called after the options are set, this method invokes all my private methods to actually draw the chart and plot the data

Parameters:

$dataref

Returns:

Status of the plot

5.2.2.18 Chart::Base::imagemap_dump ()

get the information to turn the chart into an imagemap

Returns:

Reference to an array of the image

Reimplemented in Chart::Composite.

5.2.2.19 Chart::Base::minimum (list *array*)

determine minimum of an array of values

Parameters:

@array List of numerical values

Returns:

Minimal value of list of values

5.2.2.20 Chart::Base::maximum (list *array*)

determine maximum of an array of values

Parameters:

@array List of numerical values

Returns:

Maximal value of list of values

5.2.2.21 Chart::Base::arccos (scalar *a*)

Function arccos(a).

Parameters:

$a Value

Returns:

arccos(a)

5.2.2.22 Chart::Base::arcsin (scalar *a*)

Function arcsin(a).

Parameters:

$a Value

Returns:

arcsin(a)

5.2.2.23 Chart::Base::true (scalar *b*)

determine true value of argument

Parameters:

← ***$b*** Bool value to check for true

Returns:

1 if argument is equal to TRUE, true, 1, t, T, and defined

5.2.2.24 Chart::Base::false (scalar *b*)

determine false value of argument

Parameters:

← ***$b*** Bool value to check for true

Returns:

1 if argument is equal to false, FALSE, 0, f, F or undefined

5.2.2.25 private int Chart::Base::_init (scalar *x*, scalar *y*)

Initialize all default options here.

Parameters:

← ***$x*** Width of the final image in pixels (Default: 400)

← ***$y*** Height of the final image in pixels (Default: 300)

5.2.2.26 private int Chart::Base::_copy_data (scalar *extern_ref*)

Copy external data via a reference to internal memory.

Remember the external reference.

Therefore, this function can anly be called once!

Parameters:

$extern_ref Reference to external data space

5.2.2.27 private int Chart::Base::_color_role_to_index (\list *list_of_roles*)

Parameters:

\***@list_of_roles*** List of roles

Returns:

a (list of) color index(es) corresponding to the (list of) role(s) in @_.

5.2.2.28 private array Chart::Base::_color_spec_to_rgb (scalar *role*, scalar *spec*)

Return an array (list of) rgb values for spec.

Parameters:

← ***$role*** name of a role

← ***$spec*** [r,g,b] or name

Returns:

array of rgb values as a list (i.e., @rgb)

5.2.2.29 private int Chart::Base::_draw_sub_title ()

draw the sub-title for the chart

See also:

_draw_title
_draw_sub_title() is more or less obsolete as _draw_title() does the same by writing more than one line as the title. Both use decreased width and height of the font by one.

Returns:

status

5.2.2.30 private int Chart::Base::_sort_data ()

sort the data nicely (mostly for the pareto charts and xy-plots)

Returns:

status

5.2.2.31 private int Chart::Base::_find_x_scale ()

For a xy-plot do the same for the x values, as '_find_y_scale' does for the y values!

See also:

_find_y_scale

Returns:

status

5.2.2.32 private int Chart::Base::_find_y_scale ()

find good values for the minimum and maximum y-value on the chart

Returns:

status

New version, re-written by David Pottage of Tao Group.

This code is ∗AS IS∗ and comes with ∗NO WARRANTY∗

This Sub calculates correct values for the following class local variables, if they have not been set by the user.

max_val, min_val: The maximum and minimum values for the y axis.

y_ticks: The number of ticks to plot on the y scale, including the end points. e.g. If the scale runs from 0 to 50, with ticks every 10, y_ticks will have the value of 6.

y_tick_labels: An array of strings, each is a label for the y axis.

y_tick_labels_length: The length to allow for B tick labels. (How long is the longest?)

Reimplemented in Chart::Direction, and Chart::HorizontalBars.

5.2.2.33 private Chart::Base::_calcTickInterval (scalar *dataset_min*, scalar *dataset_max*, scalar *flag_fixed_min*, scalar *flag_fixed_max*, scalar *minTicks*, scalar *maxTicks*)

Calculate the Interval between ticks in y direction.

Calculate the Interval between ticks in y direction and compare the number of ticks to the user's given values min_y_ticks, max_y_ticks.

Parameters:

← ***$dataset_min*** Minimal value in y direction

← ***$dataset_max*** Maximal value in y direction

← ***$flag_fixed_min*** Indicator whether the dataset_min value is fixed

← ***$flag_fixed_max*** Indicator whether the dataset_max value is fixed

← ***$minTicks*** Minimal number of ticks wanted

← ***$maxTicks*** Maximal number of ticks wanted

Returns:

Array of ($tickInterval, $tickCount, $pMin, $pMax)

Reimplemented in Chart::Direction.

5.2.2.34 private int Chart::Base::_calcXTickInterval (scalar *min*, scalar *max*, scalar *minF*, scalar *maxF*, scalar *minTicks*, scalar *maxTicks*)

Calculate the Interval between ticks in x direction.

Calculate the Interval between ticks in x direction and compare the number of ticks to the user's given values minTicks, maxTicks.

Parameters:

← ***$min*** Minimal value of dataset in x direction

← ***$max*** Maximal value of dataset in x direction

← ***$minF*** Inddicator if those min value is fixed

← ***$maxF*** Inddicator if those max value is fixed

← ***$minTicks*** Minimal number of tick in x direction

← ***$maxTicks*** Maximal number of tick in x direction

Returns:

$tickInterval, $tickCount, $pMin, $pMax

5.2.2.35 private int Chart::Base::_countTicks (scalar *min*, scalar *max*, scalar *interval*)

Works out how many ticks would be displayed at that interval.

Parameters:

$min Minimal value

$max Maximal value

$interval value

Returns:

($tickCount, $minR, $maxR)

e.g min=2, max=5, interval=1, result is 4 ticks.

written by David Pottage of Tao Group.

$minR = $self->_round2Tick($min, $interval, -1);

$maxR = $self->_round2Tick($max, $interval, 1);

$tickCount = ($maxR/$interval) - ($minR/$interval) +1;

5.2.2.36 private int Chart::Base::_round2Tick (scalar *input*, scalar *interval*, scalar *roundUP*)

Rounds up or down to the next tick of interval size.

$roundUP can be +1 or -1 to indicate if rounding should be up or down.

written by David Pottage of Tao Group.

Parameters:

$input

$interval

$roundUP

Returns:

retN∗interval

5.2.2.37 private array Chart::Base::_sepFP (scalar *num*)

Seperates a number into it's base 10 floating point exponent & mantisa.

written by David Pottage of Tao Group.

Parameters:

$num Floating point number

Returns:

(exponent, mantissa)

5.2.2.38 private array Chart::Base::_find_y_range ()

Find minimum and maximum value of y data sets.

Returns:

(min, max, flag_all_integers)

Reimplemented in Chart::Direction, Chart::ErrorBars, and Chart::Mountain.

5.2.2.39 private array Chart::Base::_find_x_range ()

Find minimum and maximum value of x data sets.

Returns:

(min, max)

5.2.2.40 private int Chart::Base::_plot ()

main sub that controls all the plotting of the actual chart

Returns:

status

5.2.2.41 private int Chart::Base::_draw_legend ()

let the user know what all the pretty colors mean.

The user define the position of the legend by setting option 'legend' to 'top', 'bottom', 'left', 'right' or 'none'. The legend is positioned at the defined place, respectively.

Returns:

status

Reimplemented in Chart::Composite, Chart::Direction, and Chart::ErrorBars.

5.2.2.42 private int Chart::Base::_draw_bottom_legend ()

put the legend on the bottom of the chart

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.43 private int Chart::Base::_draw_right_legend ()

put the legend on the right of the chart

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.44 private int Chart::Base::_draw_top_legend ()

put the legend on top of the chart

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.45 private int Chart::Base::_draw_left_legend ()

put the legend on the left of the chart

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.46 private int Chart::Base::_draw_none_legend ()

no legend to draw.

Just return in this case. This routine may be overwritten by subclasses.

Returns:

1

Reimplemented in Chart::Composite.

5.2.2.47 private int Chart::Base::_draw_x_label ()

draw the label for the x-axis

Get font for labels

Get the color of x_label or text

Get size of font

and write x-Label

Returns:

status

5.2.2.48 private int Chart::Base::_draw_y_label ()

draw the label for the y-axis

Returns:

status

5.2.2.49 private int Chart::Base::_draw_ticks ()

draw the ticks and tick labels

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.50 private int Chart::Base::_draw_x_number_ticks ()

draw the ticks and tick labels

Returns:

status

5.2.2.51 private int Chart::Base::_draw_x_ticks ()

draw the x-ticks and their labels

Returns:

status

Reimplemented in Chart::Composite, Chart::Direction, Chart::HorizontalBars, and Chart::HorizontalBars.

5.2.2.52 private int Chart::Base::_draw_y_ticks ()

draw the y-ticks and their labels

Returns:

status

Reimplemented in Chart::Composite, Chart::Direction, and Chart::HorizontalBars.

5.2.2.53 private int Chart::Base::_grey_background ()

put a grey background on the plot of the data itself

Returns:

status

5.2.2.54 private int Chart::Base::_draw_grid_lines ()

draw grid_lines

Returns:

status

5.2.2.55 private int Chart::Base::_draw_x_grid_lines ()

draw grid_lines for x

Returns:

status

5.2.2.56 private int Chart::Base::_draw_y_grid_lines ()

draw grid_lines for y

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.57 private int Chart::Base::_draw_y2_grid_lines ()

draw grid_lines for y

Returns:

status

Reimplemented in Chart::Composite.

5.2.2.58 private int Chart::Base::_prepare_brush (scalar *color*, scalar *type*, scalar *typeStyle*)

draw grid_lines for y

set the gdBrush object to trick GD into drawing fat lines & points of interesting shapes Needed by "Lines", "Points" and "LinesPoints" All hacked up by Richard Dice <`rdice@pobox.com`> Sunday 16 May 1999

Parameters:

$color

$type 'line','point'

$typeStyle one of 'circle', 'donut', 'triangle', 'upsidedownTriangle', 'square', 'hollowSquare', 'fatPlus'

Returns:

status

5.2.3 Member Data Documentation

5.2.3.1 Hash Chart::Base::named_colors

RGB values of named colors.

see URL `http://en.wikipedia.org/wiki/Web_colors#X11_color_-names`

5.2.3.2 private int Chart::Base::_check_data

Check the internal data to be displayed.

Make sure the data isn't really weird and collect some basic info about it

Not logical data is 'carp'ed.

Returns:

status of check

Reimplemented in Chart::Composite, and Chart::StackedBars.

5.2.3.3 private int Chart::Base::_draw

Plot the chart to the gd object

Calls:.

See also:

_draw_title
_draw_sub_title
_sort_data
_plot

Returns:

status

5.2.3.4 private int Chart::Base::_set_colors

specify my colors

Returns:

status

5.2.3.5 private int Chart::Base::_draw_title

draw the title for the chart

The title was defined by the user in set('title' => ....)

The user may define some title lines by separating them via character '\n';

The used font is taken from 'title_font';

The used color is calculated by function '_color_role_to_index' based on 'title' or 'text'

See also:

_color_role_to_index

Returns:

status

5.2.3.6 private int Chart::Base::_default_f_tick

default tick conversion function This function is pointed to be $self->{f_x_tick} resp. $self->{f_y_tick} if the user does not provide another function

Returns:

status

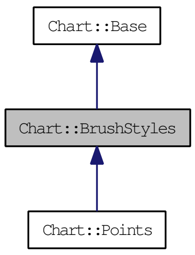
The documentation for this class was generated from the following file:

- Chart/Base.pm

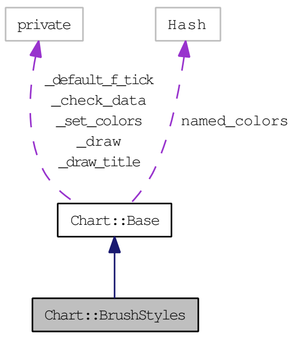
5.3 Chart::BrushStyles Class Reference

Define styles for Points and LinesPoints classes.

Inheritance diagram for Chart::BrushStyles:



Collaboration diagram for Chart::BrushStyles:



Public Functions

- OpenCircle

  Set the gdBrush object to have nice brushed object representing a circle of the size $radius.

- FilledCircle

  Set the gdBrush object to have nice brushed object representing a point of the size $radius.

- Star

  Set the gdBrush object to have nice brushed object representing a star of the size $radius.

- FilledDiamond

  Set the gdBrush object to have nice brushed object representing a filled diamond of the size $radius.

- OpenDiamond

  Set the gdBrush object to have nice brushed object representing a diamond of the size $radius-1.

- OpenRectangle

  Set the gdBrush object to have nice brushed object representing a rectangle of the height $radius-1 and width of $radius/2.

5.3.1 Detailed Description

Define styles for Points and LinesPoints classes.

This class provides functions which define different brush styles to extend the previous point as the only design for Points.pm or LinesPoints.pm

The different brush styles are:

See also:

OpenCircle
FilledCircle
Star
OpenDiamond
FilledDiamond
OpenRectangle
FilledRectangle

5.3.2 Member Data Documentation

5.3.2.1 Chart::BrushStyles::OpenCircle

Set the gdBrush object to have nice brushed object representing a circle of the size $radius.

Parameters:

← ****GD::Image*** $rbrush Reference to GD::Image
← ***int*** $radius Radius of the point in pixels
← ***int*** $color Color of the not filled point

Called by

use Chart::BrushStyles;

@Chart::Points::ISA = qw(Chart::BrushStyles);

$self->OpenCircle(\$rbrush,$radius, $newcolor);

to plot the GD::Image representing an open circle as the point

5.3.2.2 Chart::BrushStyles::FilledCircle

Set the gdBrush object to have nice brushed object representing a point of the size $radius.

Parameters:

← ∗***GD::Image*** $rbrush Reference to GD::Image
← ***int*** $radius Radius of the point in pixels
← ***int*** $color Color of the filled point

Returns:

nothing

Called by

use Chart::BrushStyles;

@Chart::Points::ISA = qw(Chart::BrushStyles);

$self->FilledCircle(\$rbrush,$radius, $color);

to plot the GD::Image representing a filled circle as the point

5.3.2.3 Chart::BrushStyles::Star

Set the gdBrush object to have nice brushed object representing a star of the size $radius.

Parameters:

← ∗***GD::Image*** $rbrush Reference to GD::Image
← ***int*** $radius Radius of the star in pixels
← ***int*** $color Color of the star

Returns:

nothing

Called by

use Chart::BrushStyles;

@Chart::Points::ISA = qw(Chart::BrushStyles);

$self->Star(\$rbrush,$radius, $color);

to get back an GD::Image representing a star as the point

5.3.2.4 Chart::BrushStyles::FilledDiamond

Set the gdBrush object to have nice brushed object representing a filled diamond of the size $radius.

Parameters:

← $*$***GD::Image*** $rbrush Reference to GD::Image

← ***int*** $radius Radius of the diamond in pixels

← ***int*** $color Color of the filled diamond

Returns:

nothing

Called by

use Chart::BrushStyles;

@Chart::Points::ISA = qw(Chart::BrushStyles);

$self->FilledDiamond(\$rbrush,$radius, $color);

to get back an GD::Image representing a filled diamond as the point

5.3.2.5 Chart::BrushStyles::OpenDiamond

Set the gdBrush object to have nice brushed object representing a diamond of the size $radius-1.

Parameters:

← $*$***GD::Image*** $rbrush Reference to GD::Image

← ***int*** $radius Radius of the diamond in pixels

← ***int*** $color Color of the diamond

Returns:

nothing

Called by

use Chart::BrushStyles;

@Chart::Points::ISA = qw(Chart::BrushStyles);

$self->OpenDiamond(\$rbrush,$radius, $color);

to get back an GD::Image representing a diamond as the point

5.3.2.6 Chart::BrushStyles::OpenRectangle

Set the gdBrush object to have nice brushed object representing a rectangle of the height $radius-1 and width of $radius/2.

Parameters:

← $*$***GD::Image*** $rbrush Reference to GD::Image

← ***int*** $radius Radius of the rectangle in pixels

← ***int*** $color Color of the rectangle

Returns:

nothing

Called by

use Chart::BrushStyles;

@Chart::Points::ISA = qw(Chart::BrushStyles);

$self->OpenDiamond(\$rbrush,$radius, $color);

to get back an GD::Image representing a rectangle as the point

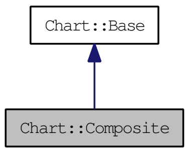
The documentation for this class was generated from the following file:

- Chart/BrushStyles.pm

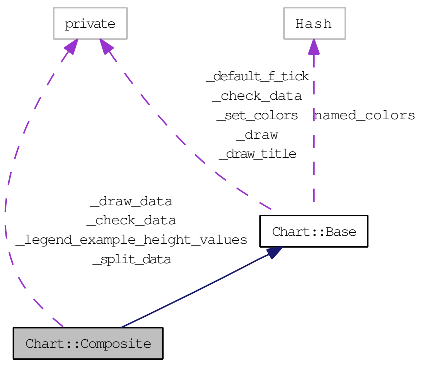
5.4 Chart::Composite Class Reference

Composite class derived from class Base.

Inheritance diagram for Chart::Composite:



Collaboration diagram for Chart::Composite:



Private Functions

- private int _check_data

  Overwrite _check_data of Chart::Base and check the internal data to be displayed.

- private _split_data

  split data to the composited classes

- private _draw_data

  finally get around to plotting the data for composite chart

- private _legend_example_height_values

  init the legend_example_height_values

- private int _draw_legend ()

  let the user know what all the pretty colors mean

- private int _draw_top_legend ()

  put the legend on the top of the data plot

- private int _draw_right_legend ()

  put the legend on the right of the chart

- private int _draw_left_legend ()

  draw the legend at the left of the data plot

- private int _draw_bottom_legend ()

  put the legend on the bottom of the chart

- private int _draw_none_legend ()

  no legend to draw.

- private int _draw_ticks ()

  draw the ticks and tick labels

- private int _draw_x_ticks ()

  draw the x-ticks and their labels

- private int _draw_y_ticks ()

  draw the y-ticks and their labels

- private _sub_update ()

  update all the necessary information in the sub-objects

- private _boundary_update ()

  copy the current gd_obj boundaries from one object to another

- private int _draw_y_grid_lines ()

  draw grid_lines for y

- private int _draw_y2_grid_lines ()

  draw grid_lines for y

Public Object Methods

- int set (hash opts)

  Set all options.

Public Functions

- imagemap_dump ()

  Overwrite function imagemap_dump of base class.

Protected Functions

- protected retval **__print_array** ()

5.4.1 Detailed Description

Composite class derived from class Base.

This class provides all functions which are specific to composite charts

5.4.2 Member Function Documentation

5.4.2.1 int Chart::Composite::set (hash *opts*)

Set all options.

Parameters:

$\leftarrow$ ***%opts*** Hash of options to the Chart

Returns:

ok or croak

Overwrite the set function of class Base to pass options to the sub-objects later

Reimplemented from Chart::Base.

5.4.2.2 Chart::Composite::imagemap_dump ()

Overwrite function imagemap_dump of base class.

Get the information to turn the chart into an imagemap had to override it to reassemble the @data array correctly

Returns:

Reference to an array of the image

Reimplemented from Chart::Base.

5.4.2.3 private int Chart::Composite::_draw_legend ()

let the user know what all the pretty colors mean

Returns:

status

Reimplemented from Chart::Base.

5.4.2.4 private int Chart::Composite::_draw_top_legend ()

put the legend on the top of the data plot

Overwrite the base class _draw_top_legend

Returns:

status

Reimplemented from Chart::Base.

5.4.2.5 private int Chart::Composite::_draw_right_legend ()

put the legend on the right of the chart

Overwrite the base class _draw_right_legend

Returns:

status

Reimplemented from Chart::Base.

5.4.2.6 private int Chart::Composite::_draw_left_legend ()

draw the legend at the left of the data plot

Overwrite the base class _draw_left_legend

Returns:

status

Reimplemented from Chart::Base.

5.4.2.7 private int Chart::Composite::_draw_bottom_legend ()

put the legend on the bottom of the chart

Overwrite the base class _draw_bottom_legend

Returns:

status

Reimplemented from Chart::Base.

5.4.2.8 private int Chart::Composite::_draw_none_legend ()

no legend to draw.

. just update the color tables for subs

This routine overwrites this function of the Base class

Returns:

status

Reimplemented from Chart::Base.

5.4.2.9 private int Chart::Composite::_draw_ticks ()

draw the ticks and tick labels

Overwrites function _draw_ticks() of base class

Returns:

status

Reimplemented from Chart::Base.

5.4.2.10 private int Chart::Composite::_draw_x_ticks ()

draw the x-ticks and their labels

Overwrites function _draw_x_ticks() of base class

Returns:

status

Reimplemented from Chart::Base.

5.4.2.11 private int Chart::Composite::_draw_y_ticks ()

draw the y-ticks and their labels

Overwrites function _draw_y_ticks() of base class

Returns:

status

Reimplemented from Chart::Base.

5.4.2.12 private Chart::Composite::_sub_update ()

update all the necessary information in the sub-objects

Only for Chart::Composite

5.4.2.13 private Chart::Composite::_boundary_update ()

copy the current gd_obj boundaries from one object to another

Only for Chart::Composite

5.4.2.14 private int Chart::Composite::_draw_y_grid_lines ()

draw grid_lines for y

Overwrites this function of Base

Reimplemented from Chart::Base.

5.4.2.15 private int Chart::Composite::_draw_y2_grid_lines ()

draw grid_lines for y

Overwrites this function of Base

Reimplemented from Chart::Base.

5.4.3 Member Data Documentation

5.4.3.1 private int Chart::Composite::_check_data

Overwrite _check_data of Chart::Base and check the internal data to be displayed.

Make sure the data isn't really weird and collect some basic info about it

Returns:

status of check

Reimplemented from Chart::Base.

5.4.3.2 private Chart::Composite::_split_data

split data to the composited classes

create sub-objects for each type, store the appropriate data sets in each one, and stick the correct values into them (ie. 'gd_obj');

5.4.3.3 private Chart::Composite::_legend_example_height_values

init the legend_example_height_values

The documentation for this class was generated from the following file:

- Chart/Composite.pm

5.5 Chart::Constants Class Reference

Constants class defines all necessary constants for Class Chart.

5.5.1 Detailed Description

Constants class defines all necessary constants for Class Chart.

Defined are

PI = 3.141...

Usage:

use Chart:Constants; ...

My $pi = Chart::Constants::PI;

...

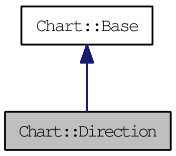
The documentation for this class was generated from the following file:

- Chart/Constants.pm

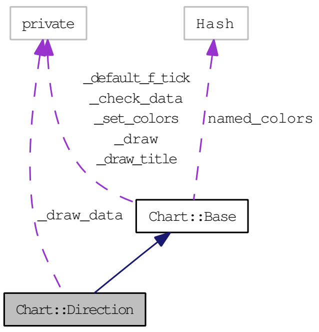
5.6 Chart::Direction Class Reference

Direction class derived class for Chart to implement direction charts.

Inheritance diagram for Chart::Direction:



Collaboration diagram for Chart::Direction:



Private Functions

- private _draw_data

  finally get around to plotting the data for direction charts

- private int _find_y_scale ()

  we use the find_y_scale methode to determine the labels of the circles and the amount of them

- private _calcTickInterval (scalar dataset_min, scalar dataset_max, scalar flag_-fixed_min, scalar flag_fixed_max, scalar minTicks, scalar maxTicks)

  Calculates the ticks for direction in normalised units.

- private int _draw_y_ticks ()

  draw the circles and the axes

- private int _draw_x_ticks ()

  We don't need x ticks, it's all done in _draw_y_ticks.

- private int _prepare_brush (scalar color, scalar type) *set the gdBrush object to trick GD into drawing fat lines*

- private int _draw_legend ()

  let them know what all the pretty colors mean

- private array _find_y_range ()

  Find minimum and maximum value of y data sets.

Public Object Methods

Todo

calculate the width of the labels

- int set (hash opts)

  Set all options.

- int add_dataset (list data)

  Add many datasets to the dataref.

Protected Object Methods

- protected retval **_calcTickInterval** ()

5.6.1 Detailed Description

Direction class derived class for Chart to implement direction charts.

5.6.2 Member Function Documentation

5.6.2.1 int Chart::Direction::set (hash *opts*)

Set all options.

Parameters:

← ***%opts*** Hash of options to the Chart

Returns:

ok or croak

main method for customizing the chart, lets users specify values for different parameters

dont check the number of points in the added datasets in a polarplot overwrite Base method

Reimplemented from Chart::Base.

5.6.2.2 int Chart::Direction::add_dataset (list *data*)

Add many datasets to the dataref.

Graph API

Overwrite Base method

Parameters:

@data Dataset to add

Reimplemented from Chart::Base.

5.6.2.3 private int Chart::Direction::_find_y_scale ()

we use the find_y_scale methode to determine the labels of the circles and the amount of them

Returns:

status

This function is an overwrite to the same function found in the base class Chart::Base

Reimplemented from Chart::Base.

5.6.2.4 private Chart::Direction::_calcTickInterval (scalar *dataset_min*, scalar *dataset_max*, scalar *flag_fixed_min*, scalar *flag_fixed_max*, scalar *minTicks*, scalar *maxTicks*)

Calculates the ticks for direction in normalised units.

Calculate the Interval between ticks in y direction and compare the number of ticks to the user given values min_y_ticks, max_y_ticks

Parameters:

← ***$dataset_min*** Minimal value in y direction

← ***$dataset_max*** Maximal value in y direction

← ***$flag_fixed_min*** Indicator whether the dataset_min value is fixed

← ***$flag_fixed_max*** Indicator whether the dataset_max value is fixed

← ***$minTicks*** Minimal number of ticks wanted

← ***$maxTicks*** Maximal number of ticks wanted

Returns:

$tickInterval, $tickCount, $pMin, $pMax

Reimplemented from Chart::Base.

5.6.2.5 private int Chart::Direction::_draw_y_ticks ()

draw the circles and the axes

Overwrites _draw_y_ticks() of Base class

Returns:

status

Reimplemented from Chart::Base.

5.6.2.6 private int Chart::Direction::_draw_x_ticks ()

We don't need x ticks, it's all done in _draw_y_ticks.

Returns:

status

Overwrites the corresponding function in Base

Reimplemented from Chart::Base.

5.6.2.7 private int Chart::Direction::_prepare_brush (scalar *color*, scalar *type*)

set the gdBrush object to trick GD into drawing fat lines

Parameters:

$color

$type

Returns:

status

5.6.2.8 private int Chart::Direction::_draw_legend ()

let them know what all the pretty colors mean

Returns:

status

Overwrite corresponding function of Base

Reimplemented from Chart::Base.

5.6.2.9 private array Chart::Direction::_find_y_range ()

Find minimum and maximum value of y data sets.

Returns:

(min, max, flag_all_integers)

Overwrites corresponding Base function

Reimplemented from Chart::Base.

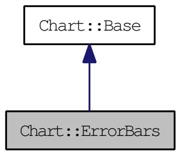
The documentation for this class was generated from the following file:

- Chart/Direction.pm

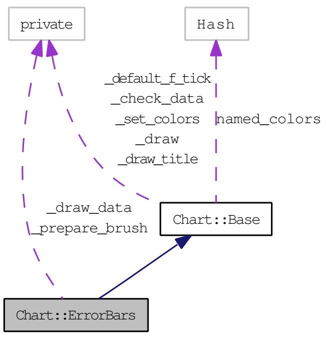
5.7 Chart::ErrorBars Class Reference

ErrorBars class derived from class Base.

Inheritance diagram for Chart::ErrorBars:



Collaboration diagram for Chart::ErrorBars:



Private Functions

- private _draw_data

  finally get around to plotting the data

- private _prepare_brush

  set the gdBrush object to trick GD into drawing fat lines

- private int _draw_legend ()

  let them know what all the pretty colors mean

Protected Object Methods

- protected retval _find_y_range ()

  Find minimum and maximum value of y data sets.

5.7.1 Detailed Description

ErrorBars class derived from class Base.

This class provides all functions which are specific to pointes having carrying vertical bars which represent errors or standard deviations

5.7.2 Member Function Documentation

5.7.2.1 protected retval Chart::ErrorBars::_find_y_range ()

Find minimum and maximum value of y data sets.

Returns:

(min, max, flag_all_integers)

Reimplemented from Chart::Base.

5.7.2.2 private int Chart::ErrorBars::_draw_legend ()

let them know what all the pretty colors mean

Returns:

status # let them know what all the pretty colors mean

Reimplemented from Chart::Base.

5.7.3 Member Data Documentation

5.7.3.1 private Chart::ErrorBars::_draw_data

finally get around to plotting the data

Overwrites Base function

5.7.3.2 private Chart::ErrorBars::_prepare_brush

set the gdBrush object to trick GD into drawing fat lines

Overwrite Base function

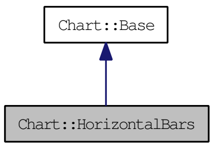
The documentation for this class was generated from the following file:

- Chart/ErrorBars.pm

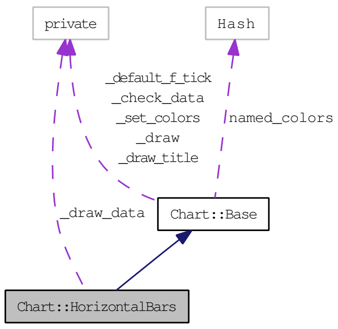
5.8 Chart::HorizontalBars Class Reference

Bars class derived from class Base.

Inheritance diagram for Chart::HorizontalBars:



Collaboration diagram for Chart::HorizontalBars:



Private Functions

- private _draw_data

  finally get around to plotting the data for (horizontal) bars

- private int _draw_x_ticks ()

  draw the x-ticks and their labels

- private int _draw_y_ticks ()

  draw the y-ticks and their labels Overwrites this function of Chart::Base

- private int _find_y_scale ()

  find good values for the minimum and maximum y-value on the chart overwrite the find_y_scale function, only to get the right f_x_ticks !!!!!

Protected Object Methods

- protected retval _draw_x_ticks ()

  draw the x-ticks and their labels Overwrites this function of Chart::Base

5.8.1 Detailed Description

Bars class derived from class Base.

This class provides all functions which are specific to horizontal bars

5.8.2 Member Function Documentation

5.8.2.1 private int Chart::HorizontalBars::_draw_x_ticks ()

draw the x-ticks and their labels Overwrites this function of Chart::Base

Returns:

status

Reimplemented from Chart::Base.

5.8.2.2 private int Chart::HorizontalBars::_draw_x_ticks ()

draw the x-ticks and their labels

Returns:

status

Reimplemented from Chart::Base.

5.8.2.3 private int Chart::HorizontalBars::_draw_y_ticks ()

draw the y-ticks and their labels Overwrites this function of Chart::Base

Returns:

status

Reimplemented from Chart::Base.

5.8.2.4 private int Chart::HorizontalBars::_find_y_scale ()

find good values for the minimum and maximum y-value on the chart overwrite the find_y_scale function, only to get the right f_x_ticks !!!!!

Returns:

status

Reimplemented from Chart::Base.

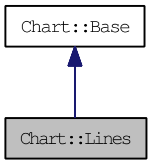
The documentation for this class was generated from the following file:

- Chart/HorizontalBars.pm

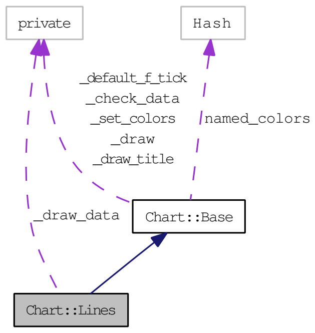
5.9 Chart::Lines Class Reference

Bars class derived from class Base.

Inheritance diagram for Chart::Lines:



Collaboration diagram for Chart::Lines:



Private Functions

- private _draw_data

  finally get around to plotting the data for lines

- private int _prepare_brush (scalar color)

  set the gdBrush object to trick GD into drawing fat lines

5.9.1 Detailed Description

Bars class derived from class Base.

This class provides all functions which are specific to lines

5.9.2 Member Function Documentation

5.9.2.1 private int Chart::Lines::_prepare_brush (scalar *color*)

set the gdBrush object to trick GD into drawing fat lines

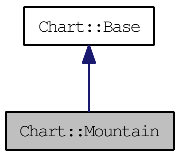
The documentation for this class was generated from the following file:

- Chart/Lines.pm

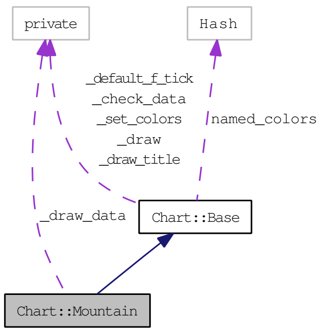
5.10 Chart::Mountain Class Reference

Mountain class derived class for Chart to implement mountain type of plots.

Inheritance diagram for Chart::Mountain:



Collaboration diagram for Chart::Mountain:



Private Functions

- private _draw_data

  draw the data

- private array _find_y_range ()

  Find minimum and maximum value of y data sets.

5.10.1 Detailed Description

Mountain class derived class for Chart to implement mountain type of plots.

Some Mountain chart details:

The effective y data value for a given x point and dataset is the sum of the actual y data values of that dataset and all datasets "below" it (i.e., with higher dataset indexes).

If the y data value in any dataset is undef or negative for a given x, then all datasets are treated as missing for that x.

The y minimum is always forced to zero.

To avoid a dataset area "cutting into" the area of the dataset below it, the y pixel for each dataset point will never be below the y pixel for the same point in the dataset below the dataset.

5.10.2 Member Function Documentation

5.10.2.1 private array Chart::Mountain::_find_y_range ()

Find minimum and maximum value of y data sets.

Returns:

(min, max, flag_all_integers)

Reimplemented from Chart::Base.

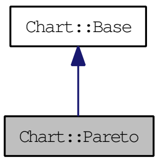
The documentation for this class was generated from the following file:

- Chart/Mountain.pm

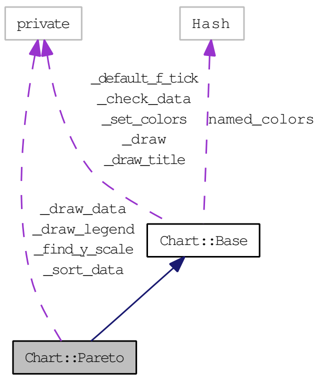
5.11 Chart::Pareto Class Reference

Pareto class derived class for Chart to implement.

Inheritance diagram for Chart::Pareto:



Collaboration diagram for Chart::Pareto:



Private Functions

- private _find_y_scale

  calculate the range with the sum dataset1.

- private _sort_data

  sort the data

- private _draw_legend

  let them know what all the pretty colors mean

- private _draw_data

  finally get around to plotting the data

5.11.1 Detailed Description

Pareto class derived class for Chart to implement.

5.11.2 Member Data Documentation

5.11.2.1 private Chart::Pareto::_find_y_scale

calculate the range with the sum dataset1.

all datas has to be positiv

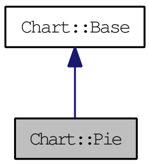
The documentation for this class was generated from the following file:

- Chart/Pareto.pm

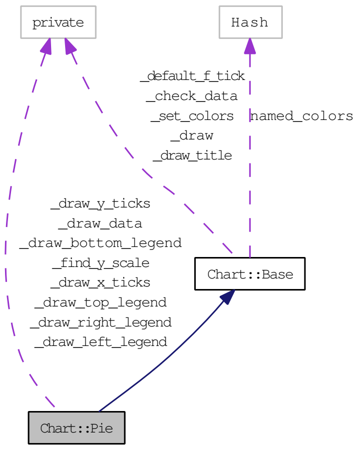
5.12 Chart::Pie Class Reference

Pie class derived class for Chart to implement pies.

Inheritance diagram for Chart::Pie:



Collaboration diagram for Chart::Pie:



Private Functions

- private _draw_data

  finally get around to plotting the data

- private _draw_right_legend

  Overwrite the legend methods to get the right legend.

- private _draw_left_legend

  put the legend on the left of the chart

- private _draw_bottom_legend

  put the legend on the bottom of the chart

- private _draw_top_legend

  put the legend on top of the chart

- private _draw_x_ticks

  Override the ticks methods for the pie charts.

- private _draw_y_ticks

  Override the ticks methods for the pie charts.

- private _find_y_scale

  Override the find_y_scale methods for the pie charts.

5.12.1 Detailed Description

Pie class derived class for Chart to implement pies.

5.12.2 Member Data Documentation

5.12.2.1 private Chart::Pie::_draw_data

finally get around to plotting the data

The user may define the kind of labelling the data by setting

'label_values' to 'percent' if she wants to have the percentages

'label_values' to 'values' if she wants to have the absolut values

'label_values' to 'both' if she wants to have absolut values and percentages

'label_values' to 'none' if she wants to have neither absolute values nor percentages

'ring' to a number less then 1 to define a ring as output; if 'ring' is 1 ore greater a full pie is plotted

5.12.2.2 private Chart::Pie::_draw_x_ticks

Override the ticks methods for the pie charts.

Here: do nothing

5.12.2.3 private Chart::Pie::_draw_y_ticks

Override the ticks methods for the pie charts.

5.12.2.4 private Chart::Pie::_find_y_scale

Override the find_y_scale methods for the pie charts.

Here: do nothing

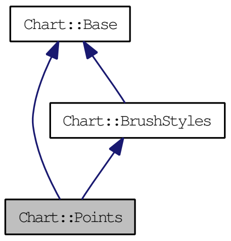
The documentation for this class was generated from the following file:

- Chart/Pie.pm

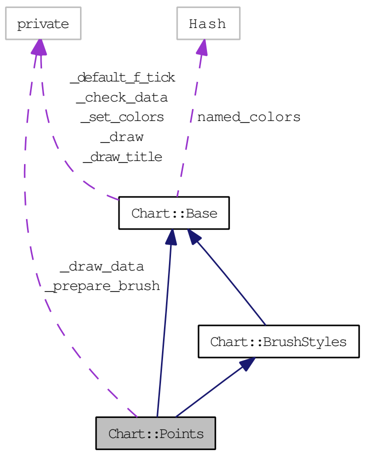
5.13 Chart::Points Class Reference

Points class derived from class Base.

Inheritance diagram for Chart::Points:



Collaboration diagram for Chart::Points:



Private Functions

- private _draw_data

  finally get around to plotting the data

- private _prepare_brush

  set the gdBrush object to have nice brushed Objects.

5.13.1 Detailed Description

Points class derived from class Base.

This class provides all functions which are specific to points

5.13.2 Member Data Documentation

5.13.2.1 private Chart::Points::_prepare_brush

set the gdBrush object to have nice brushed Objects.

These objectes are define by the option brushStyle. The size of the objects are defined by option 'pt_size', i.e., the smaller 'pt_size' is defined, the smaller these objects are.

Returns:

(GD::Image,offset)

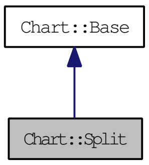
The documentation for this class was generated from the following file:

- Chart/Points.pm

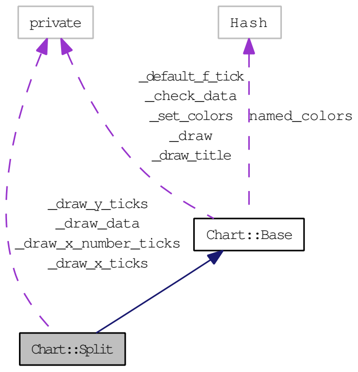
5.14 Chart::Split Class Reference

Split class derived from class Base.

Inheritance diagram for Chart::Split:



Collaboration diagram for Chart::Split:



Private Functions

- private _draw_x_number_ticks

  draw the ticks

- private _draw_x_ticks

  override the function implemented in base

- private _draw_y_ticks

  override the function implemented in base

- private _draw_data

  plot the data

5.14.1 Detailed Description

Split class derived from class Base.

This class provides all functions which are specific to splitted plots

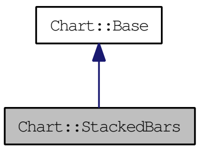
The documentation for this class was generated from the following file:

- Chart/Split.pm

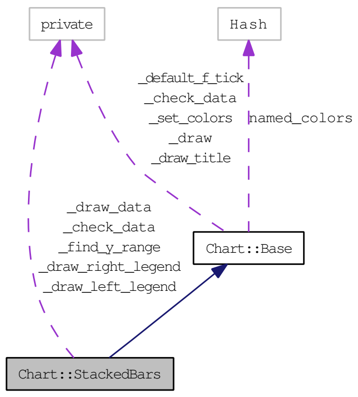
5.15 Chart::StackedBars Class Reference

StackedBars class derived from class Base.

Inheritance diagram for Chart::StackedBars:



Collaboration diagram for Chart::StackedBars:



Private Functions

- private _check_data

    override check_data to make sure we don't get datasets with positive and negative values mixed

- private **_find_y_range**
- private _draw_data

    finally get around to plotting the data

- private **_draw_left_legend**
- private **_draw_right_legend**

5.15.1 Detailed Description

StackedBars class derived from class Base.

This class provides all functions which are specific to stacked bars

The documentation for this class was generated from the following file:

- Chart/StackedBars.pm

Chapter 6

File Documentation

6.1 Chart/Bars.pm File Reference

Implementation of Chart::Bars.

Classes

- class Chart::Bars
  Bars class derived from class Base.

6.1.1 Detailed Description

Implementation of Chart::Bars.

written by

Author:

david bonner (`dbonner@cs.bu.edu`)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.2 Chart/Base.pm File Reference

Implementation of Chart::Base.

Classes

- class Chart::Base

  Base class for Chart; all other classes derived from here.

6.2.1 Detailed Description

Implementation of Chart::Base.

written by

Author:

david bonner (`dbonner@cs.bu.edu`)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.3 Chart/BrushStyles.pm File Reference

Chart::BrushStyles.

Classes

- class Chart::BrushStyles

  Define styles for Points and LinesPoints classes.

6.3.1 Detailed Description

Chart::BrushStyles.

written and maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.4 Chart/Composite.pm File Reference

Implementation of Chart::Composite.

Classes

- class Chart::Composite

  Composite class derived from class Base.

- class Chart::Composite

  Composite class derived from class Base.

6.4.1 Detailed Description

Implementation of Chart::Composite.

written by

Author:

david bonner (`dbonner@cs.bu.edu`)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

———————————————————— History: ———-

6.5 Chart/Constants.pm File Reference

Constants used in Chart:

PI.

Classes

- class Chart::Constants

  Constants class defines all necessary constants for Class Chart.

6.5.1 Detailed Description

Constants used in Chart:

PI.

written and maintained by

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.6 Chart/Direction.pm File Reference

Implementation of Chart::Direction.

Classes

- class Chart::Direction

  Direction class derived class for Chart to implement direction charts.

- class Chart::Direction

  Direction class derived class for Chart to implement direction charts.

6.6.1 Detailed Description

Implementation of Chart::Direction.

written by

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.7 Chart/ErrorBars.pm File Reference

Implementation of Chart::ErrorBars.

Classes

- class Chart::ErrorBars

    ErrorBars class derived from class Base.

- class Chart::ErrorBars

    ErrorBars class derived from class Base.

6.7.1 Detailed Description

Implementation of Chart::ErrorBars.

written by

Author:

david bonner (dbonner@cs.bu.edu)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (Chart@fs.wettzell.de)

Date:

2010-11-01

Version:

2.4.3

6.8 Chart/HorizontalBars.pm File Reference

Implementation of Chart::HorizontalBars.

Classes

- class Chart::HorizontalBars

  Bars class derived from class Base.

6.8.1 Detailed Description

Implementation of Chart::HorizontalBars.

maintained and written by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.9 Chart/Lines.pm File Reference

Implementation of Chart::Lines.

Classes

- class Chart::Lines

  Bars class derived from class Base.

6.9.1 Detailed Description

Implementation of Chart::Lines.

written by david bonner `dbonner@cs.bu.edu`

maintained by the Chart Group at Geodetic Fundamental Station Wettzell `Chart@fs.wettzell.de`

Author:

Chart Group (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.10 Chart/LinesPoints.pm File Reference

Implementation of Chart::LinesPoints.

Classes

- class **Chart::LinesPoints**

6.10.1 Detailed Description

Implementation of Chart::LinesPoints.

written by

Author:

david bonner (`dbonner@cs.bu.edu`)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.11 Chart/Mountain.pm File Reference

Implementation of Chart::Mountain.

Classes

- class Chart::Mountain

  Mountain class derived class for Chart to implement mountain type of plots.

- class Chart::Mountain

  Mountain class derived class for Chart to implement mountain type of plots.

6.11.1 Detailed Description

Implementation of Chart::Mountain.

written by david bonner `dbonner@cs.bu.edu`

maintained by

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

Updated for compatibility with changes to Chart::Base by peter clark `ninjaz@webexpress.com`

6.12 Chart/Pareto.pm File Reference

Implementation of Chart::Pareto.

Classes

- class Chart::Pareto

  Pareto class derived class for Chart to implement.

6.12.1 Detailed Description

Implementation of Chart::Pareto.

written and maintained by

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.13 Chart/Pie.pm File Reference

Implementation of Chart::Pie.

Classes

- class Chart::Pie

    Pie class derived class for Chart to implement pies.

6.13.1 Detailed Description

Implementation of Chart::Pie.

written and maintained by

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.14 Chart/Points.pm File Reference

Implementation of Chart::Points.

Classes

- class Chart::Points

    Points class derived from class Base.

6.14.1 Detailed Description

Implementation of Chart::Points.

written by

Author:

david bonner (`dbonner@cs.bu.edu`)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.15 Chart/Split.pm File Reference

Implementation of Chart::Split.

Classes

- class Chart::Split

  Split class derived from class Base.

6.15.1 Detailed Description

Implementation of Chart::Split.

written and maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3

6.16 Chart/StackedBars.pm File Reference

Implementation of Chart::StackedBars.

Classes

- class Chart::StackedBars

  StackedBars class derived from class Base.

6.16.1 Detailed Description

Implementation of Chart::StackedBars.

written by

Author:

david bonner (`dbonner@cs.bu.edu`)

maintained by the

Author:

Chart Group at Geodetic Fundamental Station Wettzell (`Chart@fs.wettzell.de`)

Date:

2010-11-01

Version:

2.4.3