# Prawn

## by example

Last Update: 2016-02-29
Prawn Version: 2.1.0
git commit: ff27f7e

# How to read this manual

This manual is a collection of examples categorized by theme and organized from the least to the most complex. While it covers most of the common use cases it is not a comprehensive guide.

The best way to read it depends on your previous knowledge of Prawn and what you need to accomplish.

If you are beginning with Prawn the first chapter will teach you the most basic concepts and how to create pdf documents. For an overview of the other features each chapter beyond the first either has a Basics section (which offer enough insight on the feature without showing all the advanced stuff you might never use) or is simple enough with only a few examples.

Once you understand the basics you might want to come back to this manual looking for examples that accomplish tasks you need.

Advanced users are encouraged to go beyond this manual and read the source code directly if any doubt is not directly covered on this manual.

# Reading the examples

The title of each example is the relative path from the Prawn source manual/ folder.

The first body of text is the introductory text for the example. Generaly it is a short description of the features illustrated by the example.

Next comes the example source code block in fixed width font.

Most of the example snippets illustrate features that alter the page in place. The effect of these snippets is shown right below a dashed line. If it doesn't make sense to evaluate the snippet inline, a box with the link for the example file is shown instead.

Note that the `stroke_axis` method used throughout the manual is part of standard Prawn. It is defined in this file:

https://github.com/prawnpdf/prawn/blob/master/lib/prawn/graphics.rb

# Basic concepts

This chapter covers the minimum amount of functionality you'll need to start using Prawn.

If you are new to Prawn this is the first chapter to read. Once you are comfortable with the concepts shown here you might want to check the Basics section of the Graphics, Bounding Box and Text sections.

The examples show:

- How to create new pdf documents in every possible way
- Where the origin for the document coordinates is. What are Bounding Boxes and how they interact with the origin
- How the cursor behaves
- How to start new pages
- What the base unit for measurement and coordinates is and how to use other convenient measures
- How to build custom view objects that use Prawn's DSL

# basic_concepts/creation.rb

There are three ways to create a PDF Document in Prawn: creating a new `Prawn::Document` instance, or using the `Prawn::Document.generate` method with and without block arguments.

The following snippet showcase each way by creating a simple document with some text drawn.

When we instantiate the `Prawn::Document` object the actual pdf document will only be created after we call `render_file`.

The generate method will render the actual pdf object after exiting the block. When we use it without a block argument the provided block is evaluated in the context of a newly created `Prawn::Document` instance. When we use it with a block argument a `Prawn::Document` instance is created and passed to the block.

The generate method without block arguments requires less typing and defines and renders the pdf document in one shot. Almost all of the examples are coded this way.

Assignment Implicit Block Explicit Block

```
# Assignment
pdf = Prawn::Document.new
pdf.text "Hello World"
pdf.render_file "assignment.pdf"

# Implicit Block
Prawn::Document.generate("implicit.pdf") do
  text "Hello World"
end

# Explicit Block
Prawn::Document.generate("explicit.pdf") do |pdf|
  pdf.text "Hello World"
end
```

This code snippet was not evaluated inline. You may see its output by running the example file located here:

http://github.com/prawnpdf/prawn/tree/master/manual/basic_concepts/creation.rb

# basic_concepts/origin.rb

This is the most important concept you need to learn about Prawn:

PDF documents have the origin `[0,0]` at the bottom-left corner of the page.

A bounding box is a structure which provides boundaries for inserting content. A bounding box also has the property of relocating the origin to its relative bottom-left corner. However, be aware that the location specified when creating a bounding box is its top-left corner, not bottom-left (hence the `[100, 300]` coordinates below).

Even if you never create a bounding box explictly, each document already comes with one called the margin box. This initial bounding box is the one responsible for the document margins.

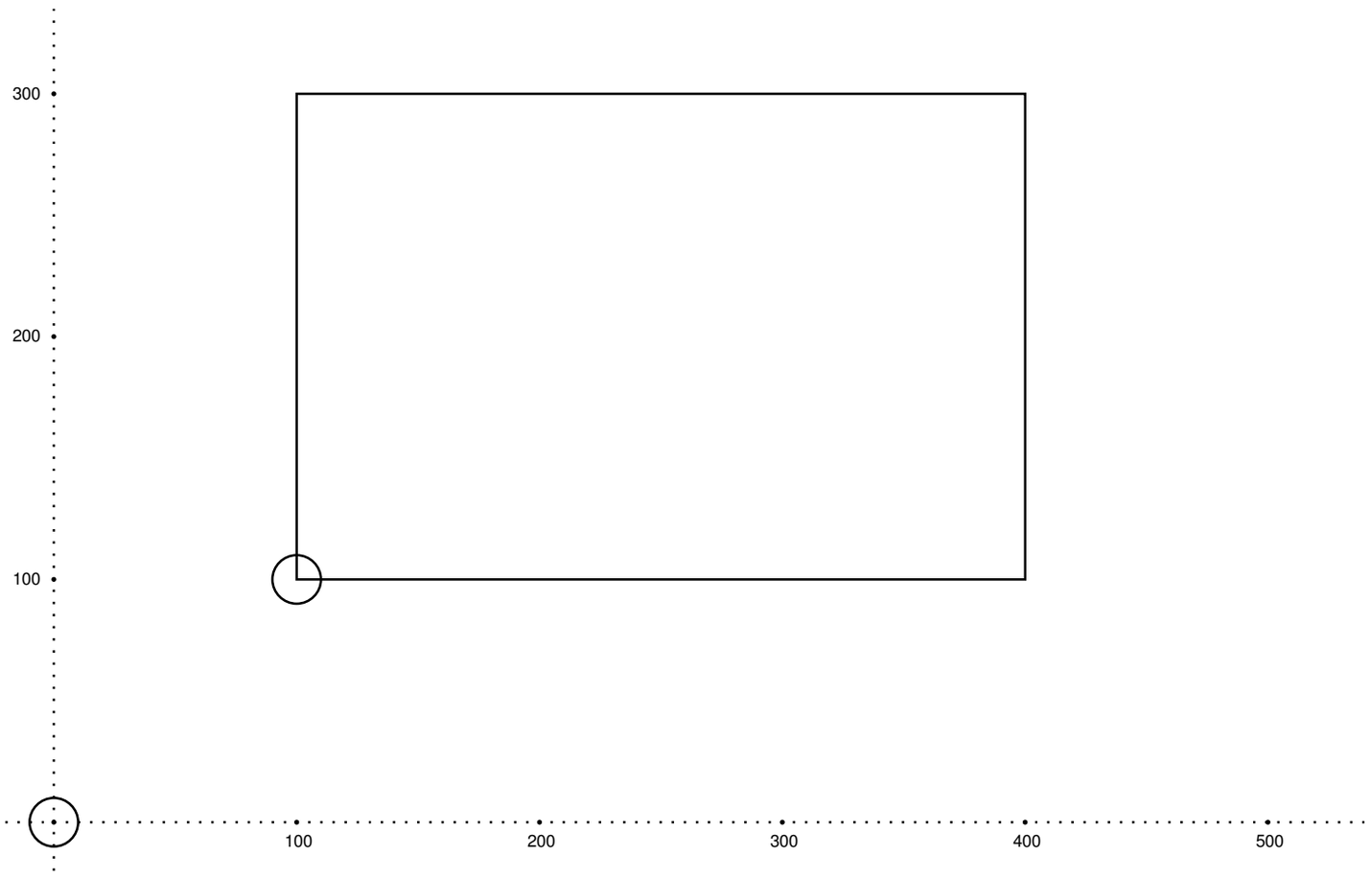
So practically speaking the origin of a page on a default generated document isn't the absolute bottom left corner but the bottom left corner of the margin box.

The following snippet strokes a circle on the margin box origin. Then strokes the boundaries of a bounding box and a circle on its origin.

```
stroke_axis

stroke_circle [0, 0], 10

bounding_box([100, 300], :width => 300, :height => 200) do
  stroke_bounds
  stroke_circle [0, 0], 10
end
```



# basic_concepts/cursor.rb

We normally write our documents from top to bottom and it is no different with Prawn. Even if the origin is on the bottom left corner we still fill the page from the top to the bottom. In other words the cursor for inserting content starts on the top of the page.

Most of the functions that insert content on the page will start at the current cursor position and proceed to the bottom of the page.

The following snippet shows how the cursor behaves when we add some text to the page and demonstrates some of the helpers to manage the cursor position. The **`cursor`** method returns the current cursor position.

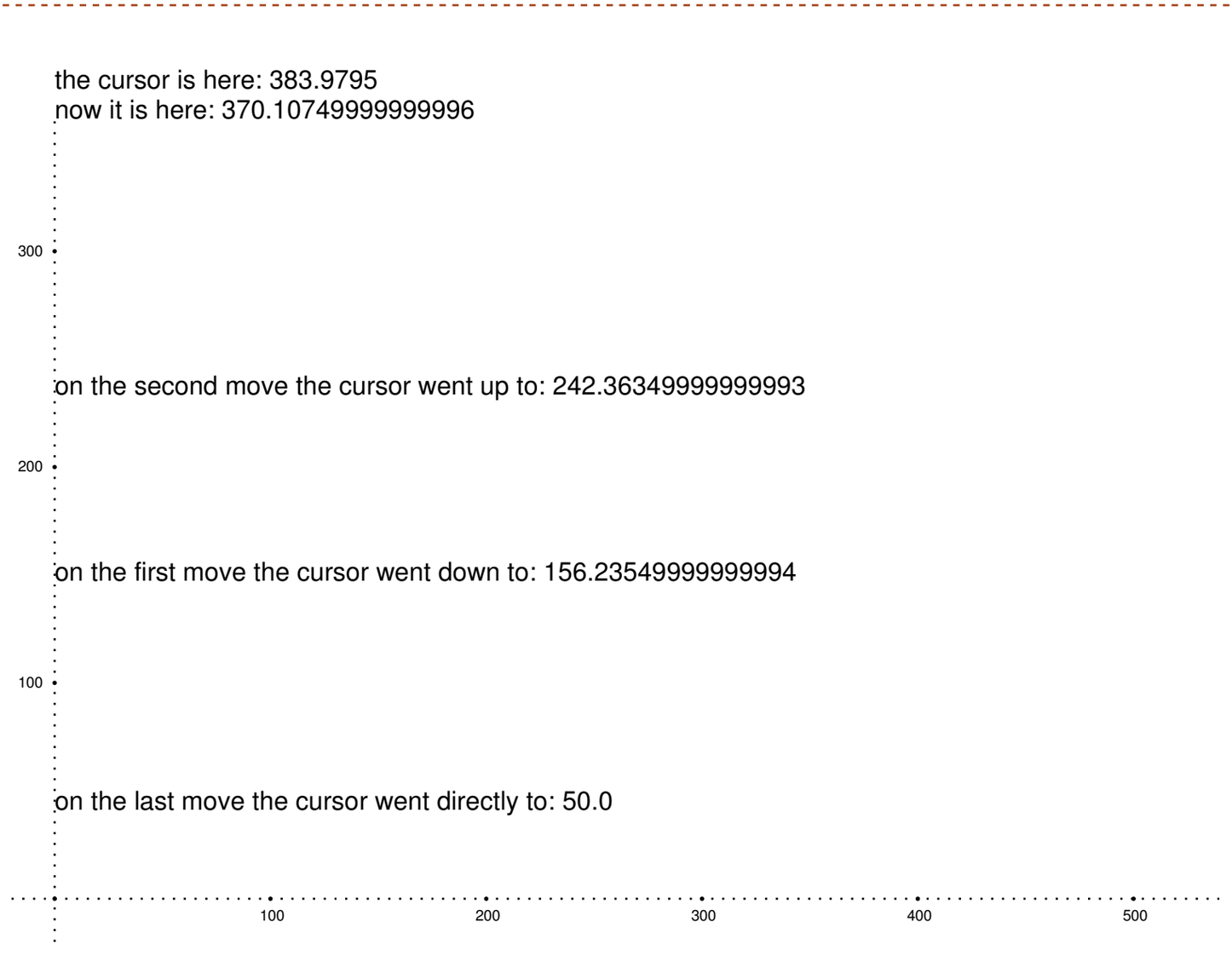
```
stroke_axis

text "the cursor is here: #{cursor}"
text "now it is here: #{cursor}"

move_down 200
text "on the first move the cursor went down to: #{cursor}"

move_up 100
text "on the second move the cursor went up to: #{cursor}"

move_cursor_to 50
text "on the last move the cursor went directly to: #{cursor}"
```



# basic_concepts/other_cursor_helpers.rb

Another group of helpers for changing the cursor position are the pad methods. They accept a numeric value and a block. **`pad`** will use the numeric value to move the cursor down both before and after the block content. **`pad_top`** will only move the cursor before the block while **`pad_bottom`** will only move after.

**`float`** is a method for not changing the cursor. Pass it a block and the cursor will remain on the same place when the block returns.

```
stroke_horizontal_rule
pad(20) { text "Text padded both before and after." }

stroke_horizontal_rule
pad_top(20) { text "Text padded on the top." }

stroke_horizontal_rule
pad_bottom(20) { text "Text padded on the bottom." }

stroke_horizontal_rule
move_down 30

text "Text written before the float block."

float do
  move_down 30
  bounding_box([0, cursor], :width => 200) do
    text "Text written inside the float block."
    stroke_bounds
  end
end

text "Text written after the float block."
```

---

Text padded both before and after.

---

Text padded on the top.

Text padded on the bottom.

---

Text written before the float block.
Text written after the float block.

Text written inside the float block.

# basic_concepts/adding_pages.rb

A PDF document is a collection of pages. When we create a new document be it with **`Document.new`** or on a **`Document.generate`** block one initial page is created for us.

Some methods might create new pages automatically like **`text`** which will create a new page whenever the text string cannot fit on the current page.

But what if you want to go to the next page by yourself? That is easy.

Just use the **`start_new_page`** method and a shiny new page will be created for you just like in the following snippet.

```
text "We are still on the initial page for this example. Now I'll ask " +
     "Prawn to gently start a new page. Please follow me to the next page."

start_new_page

text "See. We've left the previous page behind."
```

---

We are still on the initial page for this example. Now I'll ask Prawn to gently start a new page. Please follow me to the next page.

See. We've left the previous page behind.

# basic_concepts/measurement.rb

The base unit in Prawn is the PDF Point. One PDF Point is equal to 1/72 of an inch.

There is no need to waste time converting this measure. Prawn provides helpers for converting from other measurements to PDF Points.

Just **`require "prawn/measurement_extensions"`** and it will mix some helpers onto **`Numeric`** for converting common measurement units to PDF Points.

```
require "prawn/measurement_extensions"

[:mm, :cm, :dm, :m, :in, :yd, :ft].each do |measurement|
  text "1 #{measurement} in PDF Points: #{1.send(measurement)} pt"
  move_down 5.mm
end
```

---

1 mm in PDF Points: 2.834645669291339 pt

1 cm in PDF Points: 28.34645669291339 pt

1 dm in PDF Points: 283.46456692913387 pt

1 m in PDF Points: 2834.645669291339 pt

1 in in PDF Points: 72 pt

1 yd in PDF Points: 2592 pt

1 ft in PDF Points: 864 pt