Barcode.Ruby

Ruby バーコード生成ライブラリ

マニュアル

バージョン 1.0

有限会社 パオ・アット・オフィス

https://www.pao.ac/

目次

サーバーサイドで、18種のバーコードを自在に生成。

ユーザーズマニュアル

バージョン 1.0 — 2026年2月

有限会社 パオ・アット・オフィス

https://www.pao.ac/

はじめに

Barcode.Rubyとは

物流倉庫のピッキングリスト、医療現場の検体ラベル、ECサイトの出荷伝票——。

バーコードはあらゆる業務システムの「最後の1ミリ」を担っています。

Barcode.Ruby は、そのバーコードを Pure Ruby で生成するライブラリです。

C拡張不要、ネイティブコンパイル不要。require 'barcode_pao' だけでインストールでき、1次元・2次元あわせて全18種 のバーコードを、PNG画像・JPEG画像・SVGベクター で出力できます。

Rubyの柔軟性とシンプルさをそのまま活かせるため、SinatraやRailsなどのWebフレームワークに組み込めば、手軽にバーコード生成APIを構築できます。

```
require 'barcode_pao'

qr = BarcodePao::QRCode.new(BarcodePao::FORMAT_PNG)
qr.draw("https://www.pao.ac/", 300)
base64 = qr.get_image_base64
```

たった3行で、QRコードが Base64 文字列になって返ってきます。

特長

特長	説明
Pure Ruby	外部C拡張不要。依存gemは chunky_png と ttfunk のみ
サーバーサイド特化	Sinatra / Rails と組み合わせるだけで REST API バーコードサーバーが完成
PNG / JPEG / SVG	画面表示にはPNG、印刷にはSVG、サムネイルにはJPEG。用途で使い分け可能
18種のバーコード	1D・2D・GS1・郵便まで、業務で必要なバーコードを網羅
attr_accessor パターン	Ruby らしい直感的なプロパティ設定。bc.show_text = true で簡潔に
豊富なカスタマイズ	色、テキスト、バー幅調整、均等割付まで細かく制御可能

対応バーコード一覧

1次元バーコード（14種類）

バーコード	コンストラクタ	用途
Code39	Code39.new	工場の部品ラベル、軍事規格
Code93	Code93.new	Code39の高密度版。郵便・物流
Code128	Code128.new	物流の標準。ASCII全文字対応
GS1-128	GS1128.new	医薬品・物流。ロット番号管理
NW-7	NW7.new	宅配便の送り状、図書館
ITF	ITF.new	段ボール箱の集合包装用
Matrix 2of5	Matrix2of5.new	工業用途
NEC 2of5	NEC2of5.new	日本の工業現場
JAN-8	JAN8.new	小さな商品用
JAN-13	JAN13.new	日本の商品バーコードの標準
UPC-A	UPCA.new	北米の商品コード
UPC-E	UPCE.new	UPC-Aの短縮版
GS1 DataBar 標準型	GS1DataBar14.new	青果・精肉売り場
GS1 DataBar 限定型	GS1DataBarLimited.new	小型商品向け
GS1 DataBar 拡張型	GS1DataBarExpanded.new	クーポン・特売情報

郵便バーコード（1種類）

バーコード	コンストラクタ	用途
郵便カスタマバーコード	YubinCustomer.new	郵便物の住所バーコード

2次元バーコード（3種類）

バーコード	コンストラクタ	用途
QRコード	QRCode.new	URL、決済、名刺交換
DataMatrix	DataMatrix.new	電子部品の超小型マーキング
PDF417	PDF417.new	運転免許証、搭乗券

できること

PNG画像出力 — Base64でそのまま返せる

draw でバーコードを生成し、get_image_base64 を呼ぶだけで Base64エンコードされたPNG画像が返ってきます。HTMLの <img> タグにそのまま埋め込めるので、REST APIの戻り値としてそのままクライアントに返せます。

```
qr = BarcodePao::QRCode.new(BarcodePao::FORMAT_PNG)
qr.draw("Hello World", 300)
base64 = qr.get_image_base64
# "..."
```

SVGベクター出力 — 拡大しても美しい

出力形式を FORMAT_SVG にするだけで、ベクター形式のSVG文字列が得られます。どれだけ拡大しても線がぼやけないため、印刷用途に最適です。

```
bc = BarcodePao::Code128.new(BarcodePao::FORMAT_SVG)
bc.draw("Hello-2026", 400, 100)
svg = bc.get_svg
# "<svg xmlns="...">...</svg>"
```

バイト列出力 — ファイル保存やHTTPレスポンスに

get_image_memory を使えば、PNG/JPEG のバイト列を直接取得できます。

```
bc = BarcodePao::Code39.new(BarcodePao::FORMAT_PNG)
bc.draw("HELLO", 400, 100)
image_bytes = bc.get_image_memory
File.binwrite("barcode.png", image_bytes)
```

カスタマイズ — 色もテキストも思いのままに

```
#
bc.set_foreground_color(0, 0, 128, 255)   #
bc.set_background_color(255, 255, 240, 255) #

#
bc.show_text = true
bc.text_font_scale = 1.2
bc.text_even_spacing = true

#
bc.px_adjust_black = -1
bc.px_adjust_white = 1

#
bc.fit_width = true
```

導入方法

ダウンロード

https://www.pao.ac/barcode.ruby/#download

パッケージ	内容	こんな方に
Easy 2 Steps	QRコード生成の最小RESTサーバー	まずは動かしてみたい方
All-in-One	全18種対応のフル機能RESTサーバー	本格的に評価したい方

ファイル構成

```
barcode_ruby_pure/
├── easy2steps/
│   ├── Gemfile
│   ├── app.rb
│   └── views/index.erb
├── allinone/
│   ├── Gemfile
│   ├── app.rb
│   ├── views/index.erb
│   └── public/
└── barcode_pao/
    ├── lib/barcode_pao.rb
    ├── lib/barcode_pao/...
    └── Roboto-Regular.ttf
```

起動方法

```
cd barcode_ruby_pure/easy2steps
bundle install
ruby app.rb
# → http://localhost:5740
```

クイックスタート — 最初の1本を生成しよう

QRコードをPNGで生成

```
require 'barcode_pao'

qr = BarcodePao::QRCode.new(BarcodePao::FORMAT_PNG)
qr.error_correction_level = BarcodePao::QR_ECC_M

qr.draw("https://www.pao.ac/", 300)

base64 = qr.get_image_base64
puts "Base64: #{base64[0..50]} ..."

image_bytes = qr.get_image_memory
File.binwrite("qr.png", image_bytes)
puts "Saved: qr.png"
```

1Dバーコードを SVGで生成

```
require 'barcode_pao'

bc = BarcodePao::Code128.new(BarcodePao::FORMAT_SVG)
bc.show_text = true
bc.text_even_spacing = true
bc.draw("Hello-2026", 400, 100)

svg = bc.get_svg
File.write("code128.svg", svg)
```

REST APIサーバーで提供

```
require 'sinatra'
require 'json'
require 'barcode_pao'

get '/api/qr' do
  content_type :json
  code = params[:code] || "https://www.pao.ac/"

  qr = BarcodePao::QRCode.new(BarcodePao::FORMAT_PNG)
  qr.draw(code, 300)
  base64 = qr.get_image_base64

  { base64: base64 }.to_json
end
```

実践サンプル集

1次元バーコード

```
# Code39 —
bc = BarcodePao::Code39.new(BarcodePao::FORMAT_PNG)
bc.show_text = true
bc.show_start_stop = true
bc.draw("HELLO123", 400, 100)

# Code128 —
bc = BarcodePao::Code128.new(BarcodePao::FORMAT_PNG)
bc.show_text = true
bc.code_mode = BarcodePao::CODE128_AUTO
bc.draw("Hello123", 400, 100)

# NW-7 —
bc = BarcodePao::NW7.new(BarcodePao::FORMAT_PNG)
bc.show_text = true
bc.draw("A1234567A", 400, 100)

# ITF —
bc = BarcodePao::ITF.new(BarcodePao::FORMAT_PNG)
bc.show_text = true
bc.draw("123456", 400, 100)
```

2次元バーコード

```
# QR
qr = BarcodePao::QRCode.new(BarcodePao::FORMAT_PNG)
qr.error_correction_level = BarcodePao::QR_ECC_M
qr.draw("https://www.pao.ac/", 300)

# DataMatrix
dm = BarcodePao::DataMatrix.new(BarcodePao::FORMAT_PNG)
dm.code_size = BarcodePao::DX_SZ_AUTO
dm.draw("Hello World", 200)

# PDF417
pdf = BarcodePao::PDF417.new(BarcodePao::FORMAT_PNG)
pdf.set_error_level(BarcodePao::PDF417_ERROR_LEVEL2)
pdf.set_columns(3)
pdf.draw("Hello World", 400, 100)
```

GS1系バーコード

```
# GS1-128
gs1 = BarcodePao::GS1128.new(BarcodePao::FORMAT_PNG)
gs1.show_text = true
gs1.draw("[01]04912345123459[10]ABC123", 500, 120)

# GS1 DataBar
db = BarcodePao::GS1DataBar14.new(BarcodePao::FORMAT_PNG, BarcodePao::OMNIDIRECTIONAL)
db.draw("1234567890128", 200, 80)
```

商品・郵便バーコード

```
# JAN-13
jan13 = BarcodePao::JAN13.new(BarcodePao::FORMAT_PNG)
jan13.show_text = true
jan13.extended_guard = true
jan13.text_even_spacing = false
jan13.draw("491234567890", 300, 100)

#
yubin = BarcodePao::YubinCustomer.new(BarcodePao::FORMAT_PNG)
yubin.draw("27500263-29-2-401", 25)
```

APIリファレンス

共通メソッド（全バーコード）

メソッド	説明	デフォルト
set_output_format(fmt)	出力形式（"png", "jpeg", "svg"）	"png"
set_foreground_color(r, g, b, a)	前景色（RGBA）	黒（0,0,0,255）
set_background_color(r, g, b, a)	背景色（RGBA）	白（255,255,255,255）
get_image_base64	Base64 データURI取得	—
get_svg	SVG文字列取得（SVGモード時）	—
get_image_memory	バイト列取得	—

プロパティ	説明	デフォルト
foreground_color	前景色 [R,G,B,A]	[0,0,0,255]
background_color	背景色 [R,G,B,A]	[255,255,255,255]
fit_width	幅ぴったり描画	false
px_adjust_black	黒バー幅調整	0
px_adjust_white	白スペース幅調整	0

1次元バーコード共通

プロパティ	説明	デフォルト
show_text	テキスト表示	true
text_even_spacing	テキスト均等割付	true
text_font_scale	フォントサイズ倍率	1.0
text_vertical_offset_scale	垂直オフセット倍率	1.0
min_line_width	最小線幅	1

draw メソッド: draw(code, width, height) — 1D バーコード生成

各バーコード固有設定

Code39 / NW-7

プロパティ	説明	デフォルト
show_start_stop	スタート/ストップコード表示	true

Code128

プロパティ	説明	デフォルト
code_mode	CODE128_AUTO / CODE128_CODE_A / CODE1	CODE128_AUTO

JAN-8 / JAN-13 / UPC-A / UPC-E

プロパティ	説明	デフォルト
extended_guard	ガードバー拡張	true

GS1 DataBar 14

コンストラクタ: GS1DataBar14.new(format, symbol_type)

シンボルタイプ	定数
標準型	OMNIDIRECTIONAL
二層型	STACKED
標準二層型	STACKED_OMNIDIRECTIONAL

GS1 DataBar 拡張型

プロパティ	説明	デフォルト
columns	列数	0（自動）

郵便カスタマバーコード

draw メソッド: draw(code, height) — 幅は自動計算

QRコード

draw メソッド: draw(code, size) — 正方形

プロパティ	説明	デフォルト
string_encoding	"utf-8" / "shift-jis"	"utf-8"
error_correction_level	QR_ECC_L/M/Q/H	QR_ECC_M
version	0（自動）～ 40	0
encode_mode	QR_MODE_BINARY/NUMERIC/ALPHANUMERI	QR_MODE_BINARY

DataMatrix

draw メソッド: draw(code, size) — 正方形

プロパティ	説明	デフォルト
string_encoding	"utf-8" / "shift-jis"	"utf-8"
code_size	DX_SZ_AUTO ～ 各種サイズ定数	DX_SZ_AUTO
encode_scheme	DX_SCHEME_AUTO_BEST 等	DX_SCHEME_AUTO_BEST

PDF417

draw メソッド: draw(code, width, height)

メソッド	説明	デフォルト
set_error_level(level)	誤り訂正レベル（0～8）	PDF417_ERROR_LEVEL2
set_columns(cols)	列数	0（自動）
set_rows(rows)	行数	0（自動）
set_aspect_ratio(ratio)	縦横比	0.5
set_y_height(h)	Y高さ係数	3

動作環境

項目	要件
Ruby	3.0 以降
依存gem	chunky_png（PNG生成）、ttfunk（フォント描画）

Ruby がサポートするすべてのOS（Windows / macOS / Linux）で動作します。C拡張は不要です。

ライセンス・お問い合わせ

使用許諾

Barcode.Ruby
の使用について、利用者様と有限会社パオ・アット・オフィス（以下「弊社」）は、以下の各項目に同意するものとします。

1. 使用許諾書 — この使用許諾書は、利用者様がお使いのパソコンにおいて Barcode.Ruby を使用する場合に同意しなければならない契約書です。

2. 同意 — 利用者様が Barcode.Ruby を使用する時点で、本使用許諾書に同意されたものとします。

3. ライセンスの購入 — 製品版を使用して開発を行う場合、1 台の開発用コンピュータにつき 1 ライセンスの購入が必要です。お客様環境等、開発コンピュータでないマシンでの使用にはライセンスは不要です（ランタイムライセンスフリー）。

4. 著作権 — Barcode.Ruby の著作権は、いかなる場合においても弊社に帰属いたします。

5. 免責 — Barcode.Ruby の使用によって、直接的または間接的に生じたいかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

6. 禁止事項 — Barcode.Ruby およびその複製物を第三者に譲渡・貸与することはできません。開発ツールとしての再販・再配布を禁止します。ただし、モジュールとして組み込みを行い再販・再配布する場合は問題ございません。

7. 保証の範囲 — 弊社は Barcode.Ruby の仕様を予告なしに変更することがあります。

8. 適用期間 — 本使用許諾条件は利用者様が Barcode.Ruby を使用した日より有効です。

ライセンス

試用版: 生成されるバーコードに「SAMPLE」の透かしが表示されます。機能制限はありません。

製品版: 透かしなしでバーコードを生成できます。

お問い合わせ

Webサイト	https://www.pao.ac/
製品ページ	https://www.pao.ac/barcode.ruby/
メール	info@pao.ac

関連製品

製品	対応環境
Barcode.net	.NET（C#, VB.NET）
Barcode.jar	Java
Barcode.php	PHP
Barcode.wasm	JavaScript / TypeScript
Barcode.py	Python
Barcode.Flutter	Flutter / Dart
Barcode.Go	Go
Barcode.Rust	Rust
Barcode.Swift	Swift

Barcode.Ruby ユーザーズマニュアル バージョン 1.0 — 2026年2月