PDF::API2 for Fun and Profit

Whats on the Menu?

- Sweat
- Blood
- Tears
- Fear
- Sorrow
- Depression
- Hope
- Joy
- Fun
- Pride
- Satisfaction

Short History

- First Code implemented based on PDFlib-0.6 (AFPL)
- Changed to Text::PDF with a total rewrite as Text::PDF::API (procedural)
- Unmaintainable Code triggered rewrite into new Namespace PDF::API2 (objectoriented, LGPL)
- Object-Structure streamlined in 0.4x

Whats needed to use?

- Hard Requirements
  - Non EBCDIC Platform
  - PERL 5.8.x
  - Compress::Zlib
- Optionally
  - XML::Parser, XML::XPath
    (for the upcoming PDF::Layout)

Typical Other Modules you might also want to use

- RRD
- GD::*
- ImageMagick (ie. PerlMagick)
- Statistics::Descriptive
- Quantum::Superposition

Alternatives

Know thy Enemy !

- PERL
  - PDF.pm
  - Text::PDF
  - PDFJ
  - Panda
  - PDFlib
- Python
  - ReportLab
- JAVA
  - iText
- Others
  - Apache-FOP
  - HTMLDOC
  - Ghostscript
  - PDFLaTeX

Core Features

Documents & Media

- Supports Media-sizes by Name (eg. 'A4') or raw Values (eg. 595,842)
- Media-size may vary from Page to Page
- Portrait and Landscape Modes
- Supports Standard and Extended Document-Information Fields
- Imports Pages from existing PDFs

*?# Source

```
use PDF::API2;

$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$page=$pdf->page;
$page->mediabox('A3');
$page->rotate(90); #landscape

$pdf->info(
    'Author' => 'fredo',
    'Title' => 'doku_demo',
);

$pdf->infoMetaAttributes('C1');
$pdf->info(
    'C1' => 'Custom Field 1',
);
```

```
$imp=PDF::API2->open('some.pdf');

$page=$pdf->importPage($imp,2);
$page->mediabox('A4');

$xo=$pdf->importPageIntoForm($imp,5);

$page=$pdf->page;
$gfx=$page->gfx;

$gfx->formimage($xo, 0, 0, 0.5);
$gfx->formimage($xo, 300, 400, 0.5);

$pdf->saveas('new.pdf');

print $imp->stringify;

__END__
```

Core Fonts

- 14 PDF Core Fonts, 22 M$ Web-Fonts
- Only 8-bit Encoded
- Handles most Western-European, Greek and Cyrillic Languages
- Glyph-Support based on actual installed Fonts and Reader

Core Fonts (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$ft=$pdf->corefont('Verdana', -encode=>'latin1');

$page = $pdf->page;
$gfx=$page->gfx;
$gfx->textlabel(50,750,$ft,20,'Hello World !');

$pdf->saveas('Verdana_HelloWorld.pdf');

__END__
```

CJK Fonts

Ming (Traditional Chinese):
見言貝車金長門韋頁風飛食馬

Song (Simplified Chinese):
纟见讠贝车钅长门韦页风飞饣马

MyungJo (Korean):
간 꼦 뺐 큷 겾 낧 뷻 퇂 걻 녻 뺏 퇫 궁 늦

KozMin (Japanese/Mincho):
をあいうえおきくけこさしすせ

KozGo (Japanese/Gothic):
ｦｱｲｳｴｵｶｷｸｹｺｻｼｽｾｿﾀﾁﾂﾃﾄﾅﾆﾇﾈﾉﾊﾋ

- Handles Asian Scripts
  - Traditional Chinese
  - Simplified Chinese
  - Japanese
  - Korean
- Multibyte CMaps
- No selectable HW/FW Support in UTF8

CJK Fonts (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$ft=$pdf->cjkfont('KozMin-Bold');

$page = $pdf->page;
$gfx=$page->gfx;
$gfx->textlabel(50,750,$ft,20,
  “\x{3092}\x{3042}\x{3044}\x{304A}\x{3048}”);

$pdf->saveas('KozMinBold_Test.pdf');
__END__
```

Postscript Type1 Fonts

ACaslon-Regular
ACaslon-Italic-Regular
ACaslon-Bold
ACaslon-BoldItalic
Egyptian505BT-Roman
Egyptian505BT-Medium
Egyptian505BT-Bold
QuadraatSans-Regular
QuadraatSans-Italic-Regular
QuadraatSans-Bold-Regular
ZapfChancery-Roman
ZapfChancery-Italic-Roman
ZapfChancery-Bold

- Requires a PFA/PFB and AFM file
- Only 8-bit Encoded
- Required Glyph-Encoding limits usefulness with Unicode/UTF-8
- Handedited AFM may increase usefullness

Postscript Fonts (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$ft=$pdf->psfont('acasreg.pfb',
   -afmfile=>'acasreg.afm', -encode=>'latin1');

$page = $pdf->page;
$gfx=$page->gfx;
$gfx->textlabel(50,750,$ft,20,'Hello World !');

$pdf->saveas('ACaslon_HelloWorld.pdf');

__END__
```

Truetype & Opentype

- Uses CID-Keying for Glyphs
- May use Multibyte encodings that map into UTF-8
- Limited to 65536 unique Glyphs
- Uses sub-setting to reduce PDF size
- Improved Quality
- May use Multibyte encodings that map into UTF-8 or Adobe CMaps
- Best suited for Chinese, Japanese and Korean Fonts
- No Subsetting

TTF & OTF Fonts (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$ft=$pdf->ttfont('Stempel-Regular.otf',
   -encode=>'latin1');

$page = $pdf->page;
$gfx=$page->gfx;
$gfx->textlabel(50,750,$ft,20,'Hello World !');

$pdf->saveas('Stempel_HelloWorld.pdf');

__END__
```

Synthetic Variants

Courier-Synthetic-Narrow
Courier-Synthetic-Bold
Courier-Synthetic-Oblique
COURIER-SYNTHETIC-CAPS
Georgia-Synthetic-Narrow
Georgia-Synthetic-Bold
Georgia-Synthetic-Oblique
GEORGIA-SYNTHETIC-CAPS
Helvetica-Synthetic-Narrow
Helvetica-Synthetic-Bold
Helvetica-Synthetic-Oblique
HELVETICA-SYNTHETIC-CAPS
Trebuchet-Synthetic-Narrow
Trebuchet-Synthetic-Bold
Trebuchet-Synthetic-Oblique
TREBUCHET-SYNTHETIC-CAPS
Verdana-Synthetic-Narrow
Verdana-Synthetic-Bold
Verdana-Synthetic-Oblique
VERDANA-SYNTHETIC-CAPS

- Creates artificial:
  - Narrow/Expansion
  - Oblique
  - Bold
  - Spacings
  - Caps
- Only 8-bit Encoded

Synthetic Variants (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$ft=$pdf->ttfont('Stempel-Regular.otf', -encode=>'latin1');

$fb=$pdf->synfont($ft, -slant=>0.9, -bold=>2);

$page = $pdf->page;
$gfx=$page->gfx;
$gfx->textlabel(50,750,$fb,20,'Hello World !');

$pdf->saveas('StempelSynCondBold_HelloWorld.pdf');

__END__
```

Unicode Map Fonts

KozMin

PDFJ - 日本語PDF生成モジュール

KozMin-Bold + Times

PDFJ - 日本語PDF生成モジュール

KozGo-Italic + Trebuchet-Italic

PDFJ - 日本語PDF生成モジュール

KozGo-Italic + Georgia-Italic

PDFJ - 日本語PDF生成モジュール

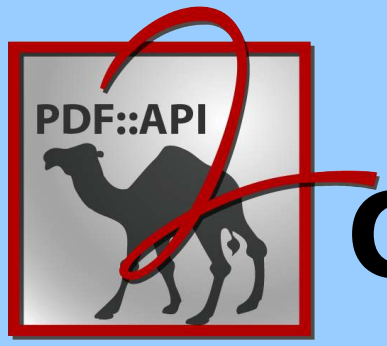
- Inspired by PDFJ
- Makes Codepoints of different Fonts available as one Virtual Font
- Allows building the “Every Glyph in a Font” Font
- Allows more than 64k Glyphs

Unicode Map Fonts (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$trbo=$pdf->corefont('Trebuchet-Bold', -encode=>'latin1');
$kgbo=$pdf->cjkfont('KozGo-Bold', -encode=>'shiftjis');
$zapf=$pdf->corefont('ZapfDingbats');
$uft=$pdf->unifont($kgbo,[$trbo,[0]],[$zapf,[0x26,0x27]]);

$page = $pdf->page;
$gfx=$page->gfx;
$gfx->textlabel(50,750,$uft,20,'\x{27A4} Hello World !');
$gfx->textlabel(50,700,$uft,20,' 日本語 PDF 生成モジュール ');

$pdf->saveas('KozGoTrebuchet_HelloWorld.pdf');
__END__
```

Colors & Colorspaces

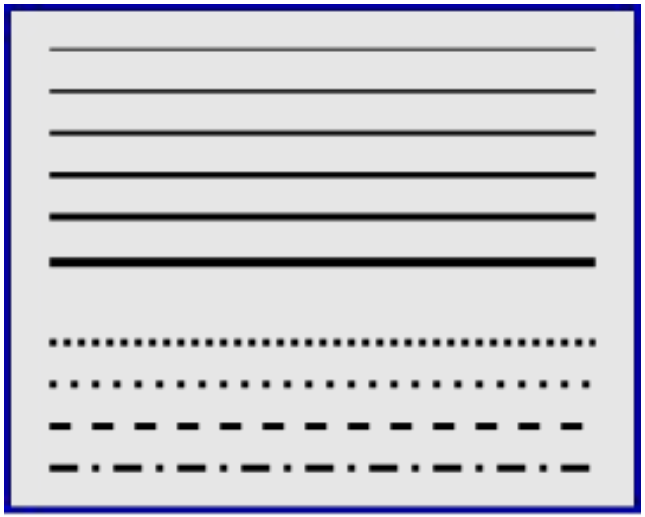
- Colors
  - Greyscale
  - RGB, CMYK, HSV
  - L*a*b, Indexed
  - X11/HTML Names
- Colorspaces
  - Color Tables
  - Separation (Tint)
  - DeviceN (Tints)

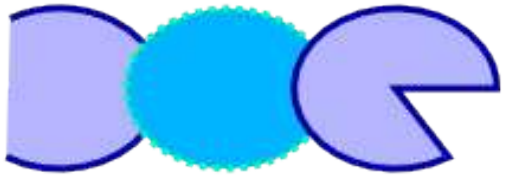


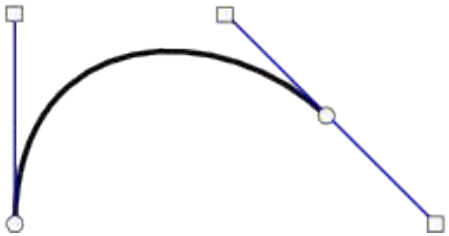
Colors & Colorspaces (2)

```
use PDF::API2;
$pdf=PDF::API2->new; $pdf->mediabox('A4');
$page=$pdf->page;
$gfx=$page->gfx;
$trb=$pdf->corefont('Trebuchet-Bold', -encode=>'latin1');
$ma=$pdf->colorspace_separation('Magenta', '%0f00');

$gfx->textlabel(50,750,$trb,20,'Red',-color=>'!0FF');
$gfx->textlabel(50,700,$trb,20,'Blue',-color=>'#0000FF');
$gfx->textlabel(50,650,$trb,20,'Yellow',-color=>'yellow');
$gfx->textlabel(50,600,$trb,20,'Green',-color=>'green2');
$gfx->textlabel(50,550,$trb,20,'Cyan',-color=>'%F000');
$gfx->textlabel(50,500,$trb,20,'Magenta',-color=>[$ma,1]);
$gfx->textlabel(50,450,$trb,20,'Lt.Grey',-color=>[0.2]);

$pdf->saveas('Colors_Test.pdf');
__END__
```

Vector Graphics







- Drawing Commands
  - Line, Arc, Spline, Curve, Rectangle, Circle, Ellipse, Pie
- Style Commands
  - Width, Dash, Fill, Stroke
- Transformations
  - Clip, Scale, Rotate, Skew, Translate

Vector Graphics (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
$pdf->mediabox('A4');

$page=$pdf->page;
$gfx=$page->gfx;

$gfx->linewidth(1);
$gfx->linedash(2,3,5);
$gfx->strokecolor('darkred');
$gfx->move(0,0);
$gfx->line(50,50);
$gfx->curve(50,100,100,100,200,50);
$gfx->stroke;

$pdf->saveas('Vector_Test.pdf');
__END__
```

Bitmap Images

- Image Formats
  - JPEG
  - PNG
  - GIF
  - TIFF
  - P*M
  - GD
- Reusable
- DPI independent
- Transforms
  - Scale
  - Rotate
  - Skew
- Partial Alpha or Transparency Support

Bitmap Images (2)

```
use PDF::API2;
$pdf=PDF::API2->new;
@imgs=( $pdf->image_gif('1.gif'),
        $pdf->image_tiff('2.tif'),
        $pdf->image_jpeg('3.jpeg') );

foreach my $img (@imgs)
{
    $page = $pdf->page;
    $page->mediabox($img->width,$img->height);
    $gfx=$page->gfx;
    $gfx->image($img,0,0,1);
}

$pdf->saveas('Images_Test.pdf');
__END__
```

Other Features

- Custom Filling
  - Shading
  - Patterns
- Barcodes
  - EAN
  - UPC
  - 3of9
  - 2of5
  - codabar
- PDF-Metadata (only in CVS)
- Access to all Low-Level PDF-Objects

Advanced Features (The Future)

New Modules

- Polluting the PDF Namespace some more with High-Level Wrapping APIs
  - PDF::Maki — experimental wrappers
  - PDF::Layout — Text-Paragraphs & Data-Grids
  - PDF::Drawable — Graphic Objects
  - PDF::Chart — Line-, Bar, Area-Graphs, etc.
- Collected Feature-Requests that did not “feel” right being realized within PDF::API2

Text with Layout

- Paragraphs
  - Right, Center, Block
- HTML-like
  - Background
  - Margin
  - Padding
  - Borders
- full Layout Objects
- only in CVS

Text with Layout (2)

```
$c=PDF::Layout->SimpleText(
    $font, $size, $text,
    -id=>'thisCell'
    -background=>'#ff8',
    -lead=>1.3, -border=>2,
    -bordercolor=>'blue',
    -borderdash=>[2,3,4,5],
    -textcolor=>'darkred' );

$d=PDF::Layout->SimpleText(
    $font, $size, $text,
    -margin=>5, -padding=>7,
    -href=>'#thisCell',
    -textcolor=>'darkgreen',
    -underline=>'simple',
    -bordertop=>2,
    -bordertopcolor=>'red',
    -borderbottom=>0.1,
    -borderbottomcolor=>'black',
    -background=>'#fff',
    -align=>'j' );
```

```
$e=PDF::Layout->SimpleText(
    $font, $size, $text,
    -href=>'http://www.sf.net',
    -margintop=>5, -margin=>20,
    -padding=>5,
    -lead=>1.3, -border=>2,
    -bordercolor=>'darkblue',
    -background=>'#fff',
    -align=>'r' );

$overflow=$c->render(
    $pdf, $page, $gfx,
    $x, $y, $w, $h );

$overflow->render(
    $pdf, $page, $gfx,
    $x, $y, $w, $h );
```

XML Markup Text

[HREF] Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibhed euismod tincidunt ut laoreet dolore magna aliquam ***erat volutpat. Ut wisi enim adet minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl*** *ut aliquip ex ea commodo consequat.* ***Duis autem vel eum*** iriure dolor in hendrerit in vulputate velit esse — molestie — consequat, vel illum fred dolore eu feugiat nulla

[HREF] Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibhed euismod tincidunt ut laoreet dolore magna aliquam ***erat volutpat. Ut wisi enim adet minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl*** *ut aliquip ex ea commodo consequat.* ***Duis autem vel eum*** iriure dolor in hendrerit in vulputate velit esse — molestie — consequat, vel illum fred dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto

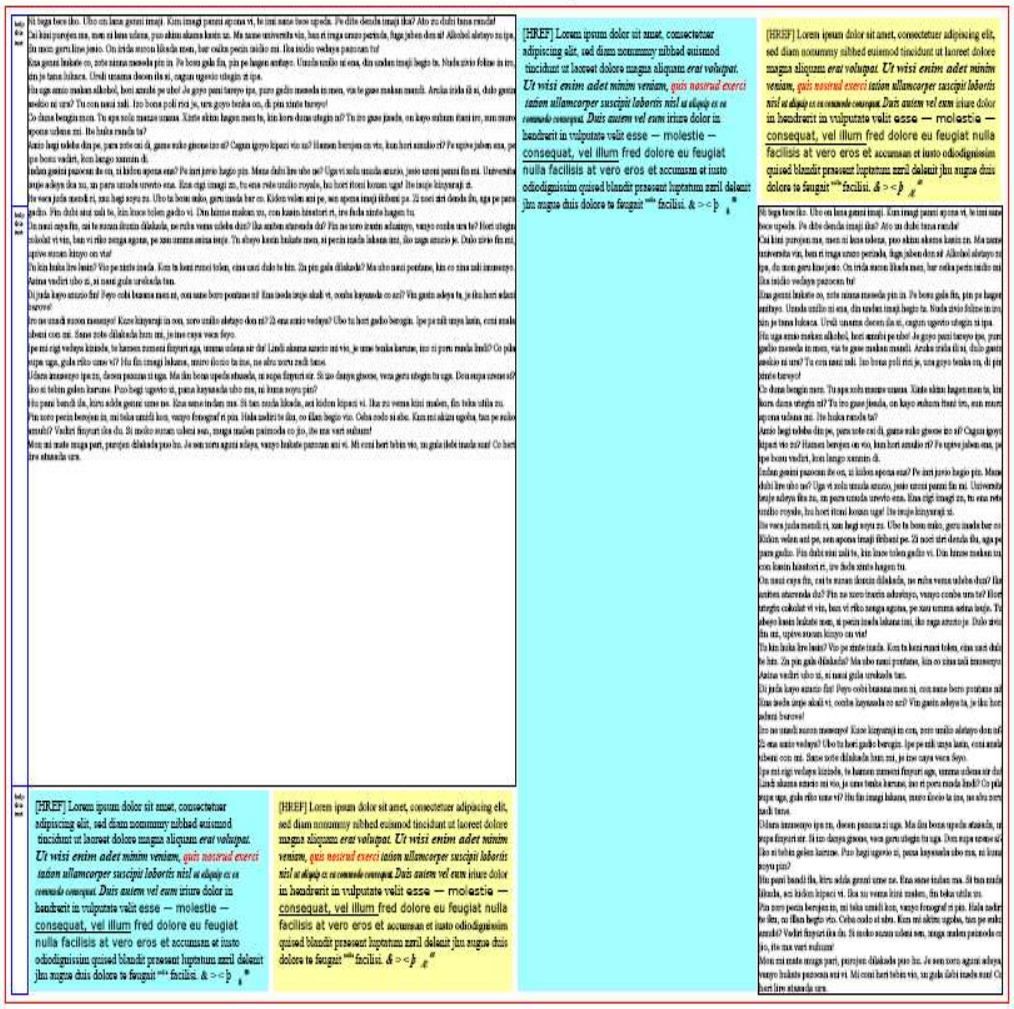
- HTML/Pango-like Markup Tags
  - <a>; <b>, <i>, <u>; <c>, <e>, <sub>, <sup>; <p>,
- Glyph-Names & HTML/SGML as Entities
- only in CVS

XML Markup Text (2)

```
$xml=<<EOT;
<p size="15"><a href="http://127.0.0.1/">
[HREF] </a>Lorem ipsum dolor sit amet,
consectetuer adipiscing elit, sed diam
nonummy nibhed euismod tincidunt ut
laoreet dolore magna aliquam <b>erat
volutpat. <e>Ut wisi enim adet </e>minim
veniam, <span color="red">quis nostrud
exerci </span>tation ullamcorper suscipit
lobortis nisl <c>ut aliquip ex ea commodo
consequat. </c>Duis autem vel eum
</b>iriure dolor in hendrerit in vulputate
 velit <span face="verdana">esse &emdash;
molestie &emdash; <u>consequat, vel illum
</u>fred dolore eu feugiat nulla facilisis
at vero eros et </span>accumsan et iusto
odiodignissim quised blandit praesent
luptatum zzril delenit jhu augue <a
href="#here">duis dolore te feugait
</a><sup size="8">nulla </sup>facilisi.
<b>&amp; &gt; &lt; <i>&#254; &lozenge;
</i><sub>&AE; </sub><sup>&ae; </sup></b>
</p>
EOT
```

```
$c=PDF::Layout->MarkupText(
    $xml,
    -fontreg=>$ftreg,
    -margin=>5,
    -padding=>5,
    -lead=>1.3, -border=>2,
    -bordercolor=>'darkblue',
    -align=>'j' );

$overflow=$c->render(
    $pdf, $page, $gfx,
    $x, $y, $w, $h );

$overflow->render(
    $pdf, $page, $gfx,
    $x, $y, $w, $h );
```

Layout Grids (aka. Tables)



- HTML-Like Table
- Column & Row-Spans
- use “Any Layout-Object” as GridCell
- Auto-Sizing and Fixed-Width
- full Layout Objects
- only in CVS

Last But Not Least

Why Use ?

- You are a Perl-Monger and use everything perlish
- Your Top-Of-The-Line-Big-Bucks Software (Crystal-Reports, SAP, eHealth, etc.) generates Reports that are Plain Ugly or simply dont fit you needs
- Already use GD::* for Reports, but sending MHTML is not an Option
- Want to create Documents that “Look like they Print”

Further Information

- PDF::API2 Homepage,
  http://pdfapi2.sf.net
- “Perl Graphics Programming”,
  Shawn Wallace – O'Reilly Inc.
- “Digitale Typographie”,
  Springer Verlag
- “Der Mensch und seine Zeichen”,
  Adrian Frutiger – marix.verlag
- “Digitales Colormanagement”,
  Jan-Peter Homann – Springer Verlag
- “Looking Good In Print”,
  Roger C. Parker – Paraglyph Press Inc.

Questions ?