Professional reports with SVG

Stefan Neufeind
SpeedPartner GmbH

About me

- Stefan Neufeind
- From Neuss (near Düsseldorf, Germany)

- Working for SpeedPartner GmbH (consulting, development, administration)

- PEAR-developer
- Loves PHP / FOSS :-)

Agenda

- About SVG
- Support for SVG
- SVG and PHP
  - PEAR::XML_SVG
  - PEAR::Image_Canvas
- Reporting with PEAR::Image_Graph
- SVG ready for use?
- Links

About SVG

- Standard defined by W3C
- XML = eXtensible Markup Language
- For 2D vector-graphics
- Static and animated
- DOM (Document Object Model)
  - Modifyable with ECMAScript, SMIL
  - Event-handlers, ...
- Editable using a simple text-editor

About SVG

Types of objects:

- Vector shapes
  - Lines, curves, ...
- Raster images
  - e.g. PNG used in SVG-document
- Text
  - Searchable, accessible, ...

About SVG

Features:

- Grouping, styling
- Transformations
- Clipping paths
- Alpha masks
- Filter effects
- Template objects
- Extensibility

About SVG

Profiles:

- SVG Tiny (SVGT)
  - For restricted devices like cellphones
- SVG Basic (SVGB)
  - For advances devices like PDAs
- Full SVG

About SVG

Namespaces:

- XML-namespace for SVG
- Usable together with other namespaces
- Allows mixing of XHTML, SVG, MathML, XUL/XBL, ... in one document

About SVG

Timeline:

- SVG 1.0:
  W3C Recommendation on 2001-09-04
- SVG 1.1:
  W3C Recommendation on 2003-01-14
- SVG 1.1 Tiny / Basic:
  W3C Recommendation on 2003-01-14
- SVG 1.2 Mobile (Tiny) / SVG 1.2 Full:
  Still drafts

About SVG

Outlook to SVG 1.2:

- Flowing text and graphics
- Xforms
- Multiple pages
- Painting/Multimedia enhancements

About SVG

Competition to SVG:

- Flash (binary format)
  - Not open (Format spec available for export)
  - Large market share
  - Poor accessibility
- VML (text format)
  - Open, but Internet Explorer only
- XAML (text source compiled to binary)
  - Published, but possibly not open

Support for SVG

- Content negotiation via HTTP
  - Deliver SVG or convert on server
- Rasterization on server (for older browser)
  - ImageMagick (quick but incomplete)
  - Batik (complete but slow [Java])
- Standalone viewer / browser-plugin

Support for SVG

Standalone viewers / browser-plugins:

- Adobe SVG Viewer
  - Free, MacOS/Windows Linux/Solaris
- Corel SVG viewer
  - Free, Windows, no longer supported
- Apache Squiggle (part of Batik toolkit)
  - Free, Java → Slow
- ...

Support for SVG

Native support:

- Opera browser
  - SVG 1.1 Tiny since 8.0 beta 3
- Mozilla
  - Firefox since 1.5 beta 1 (not on Linux yet)
  - Mozilla SeaMonkey suite
- KDE
  - Plugin for Konqueror (KSVG)
  - Since 3.4 support for SVG wallpapers :-)

Support for SVG

Native support (continued):

- Apple Safari
  - Porting of KSVG2 into WebCore in progress
- Amaya (W3C webbrowser/editor)
  - Partial SVG support
- Apache Batik SVG toolkit
  - For Java programs

Support for SVG

Mozilla support:

- included in latest builds
- native SVG-support:
  - handles compound documents mixed of SVG, MathML, XHTML, XUL (through namespaces)
  - supports SVG dom
  - allows using SVG in XBL (XUL binding language) etc.

Support for SVG

Mozilla support (continued):

- currently supports basic shapes:
  - beziers
  - stroking and filling with opacity
  - gradients
  - scripting
  - events
  - most of the DOM

Support for SVG

Mozilla support (continued):

- target:
  - full SVG 1.1 support
  - specification-conform
- big areas lacking features:
  - filters
  - svg defined fonts
  - declarative animations

SVG and PHP

PEAR-packages:

- XML_image2svg
  - Converts JPEG, GIF of PNG to base64-representation inside SVG-container
  - Not really helpful?
  - Old, seems unmaintained
- XML_svg2image
  - Converts svg to JPEG or PNG
  - Requires apache-batik via php-ext/java

SVG and PHP

PEAR-packages (continued):

- XML_SVG
  - Object-oriented API to build SVG documents
  - Based on SVG-library from Horde
  - Low level abstraction “wrapper“ for SVG
- Image_Canvas
  - High level of abstraction
  - Interface to GD, SVG, PDF, ImageMap (planned: Flash using MING / libSWF, PDF using File_PDF / cPdfWriter, ...)

SVG and PHP

PEAR::XML_SVG – source:

```
<?php
require_once('XML/SVG.php');

$svg = &new XML_SVG_Document(
  array('width' => 400, 'height' => 200));
$g = &new XML_SVG_Group(
  array('style' => 'stroke:black',
  'transform' => 'translate(200 100)'));
$g->addParent($svg);
// or: $svg->addChild($g);
```

SVG and PHP

PEAR::XML_SVG – source (continued):

```
$circle = &new XML_SVG_Circle(
  array('cx' => 0, 'cy' => 0, 'r' => 100,
  'style' => 'stroke-width:3'));
$circle->addChild(new XML_SVG_Animate(
  array('attributeName' => 'r',
        'attributeType' => 'XML',
        'from' =>   0, 'to' => 90,
        'dur' => '3s', 'fill'=> 'freeze')));
```

SVG and PHP

PEAR::XML_SVG – source (continued):

```
$circle->addChild(new XML_SVG_Animate(
  array('attributeName' => 'fill',
        'attributeType' => 'CSS',
        'from' => 'yellow', 'to' => 'red',
        'dur' => '3s', 'fill'=> 'freeze')));
$g->addChild($circle);
$text = &new XML_SVG_Text(array(
  'text' => 'SVG chart!', 'x' => 0,'y' => 0,
  'style'=> 'font-size:20;
text-anchor:middle;'));
```

SVG and PHP

PEAR::XML_SVG – source (continued):

```
$text->addChild(new XML_SVG_Animate(
  array('attributeName' => 'font-size',
        'attributeType' => 'auto',
        'from' =>  50, 'to' => 30,
        'dur' => '3s', 'fill'=> 'freeze')));
$g->addChild($text);
$svg->printElement();
?>
```

SVG and PHP

PEAR::XML_SVG – SVG:

```
<g style="stroke:black" transform="translate(200
100)">
  <circle cx="0" cy="0" r="100"
   style="stroke-width:3">
    <animate attributeName="r"
     attributeType="XML"
     from="0" to="90" dur="3s" fill="freeze" />
    <animate attributeName="fill"
     attributeType="CSS" from="yellow" to="red"
     dur="3s" fill="freeze" />
  </circle>
```

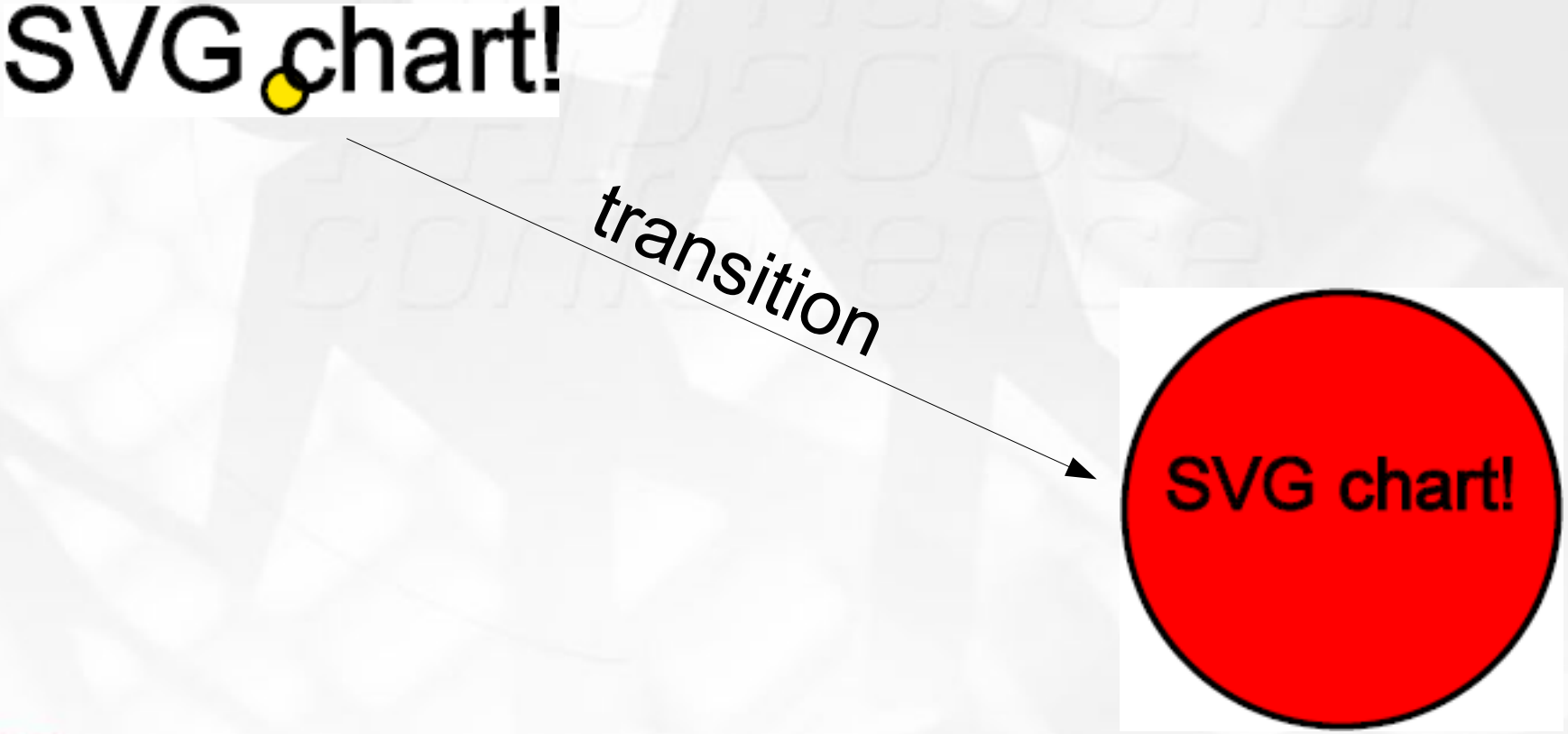
SVG and PHP

PEAR::XML_SVG – SVG (continued):

```
  <text x="0" y="0" style="font-size:20;
   text-anchor:middle;">SVG chart!
    <animate attributeName="font-size"
     attributeType="auto" from="50" to="30"
     dur="3s" fill="freeze" />
  </text>
</g>
```

SVG and PHP

PEAR::XML_SVG – animated image:



SVG and PHP

PEAR::Image_Canvas – source:

```
<?php
include_once 'Image/Canvas.php';

$Canvas =& Image_Canvas::factory('svg',
array('width' => 400, 'height' => 300));

$Canvas->setLineColor('black');
$Canvas->rectangle(array('x0' => 0,
  'y0' => 0, 'x1' => 399, 'y1' => 299));
```

SVG and PHP

PEAR::Image_Canvas – source (continued):

```
$Canvas->setGradientFill(
 array('direction' => 'horizontal',
   'start' => 'red', 'end' => 'blue'));
$Canvas->setLineColor('black');
$Canvas->ellipse(array('x' => 199,
  'y' => 149, 'rx' => 50, 'ry' => 50));
$Canvas->setFont(array('name' => 'Arial',
  'size' => 12));
$Canvas->addText(array('x' => 0, 'y' => 0,
  'text' => 'Demonstration of what
  Image_Canvas do!'));
```

SVG and PHP

PEAR::Image_Canvas – source (continued):

```
// ... some more text-generation skipped ...

$Canvas->addVertex(
  array('x' => 50, 'y' => 200));
$Canvas->addVertex(
  array('x' => 100, 'y' => 200));
$Canvas->addVertex(
  array('x' => 100, 'y' => 250));
$Canvas->setFillColor('red@0.2');
$Canvas->polygon(array('connect' => true));
```

SVG and PHP

PEAR::Image_Canvas – source (continued):

```
$Canvas->image(array('x' => 398, 'y' => 298,
  'filename' => './pear-icon.png',
  'alignment' => array(
    'horizontal' => 'right',
    'vertical' => 'bottom')));

$Canvas->show();
?>
```

SVG and PHP

PEAR::Image_Canvas – SVG:

```
<defs>
  <linearGradient id="gradient_1" x1="0%"
y1="0%" x2="100%" y2="0%">
    <stop offset="0%" style="stop-
color:rgb(255,0,0);"/>
    <stop offset="100%" style="stop-
color:rgb(0,0,255);"/>
  </linearGradient>
</defs>
<rect x="0" y="0" width="399" height="299"
style="stroke-width:1;stroke:rgb(0,0,0);
fill:none;"/>
```

SVG and PHP

PEAR::Image_Canvas – SVG (continued):

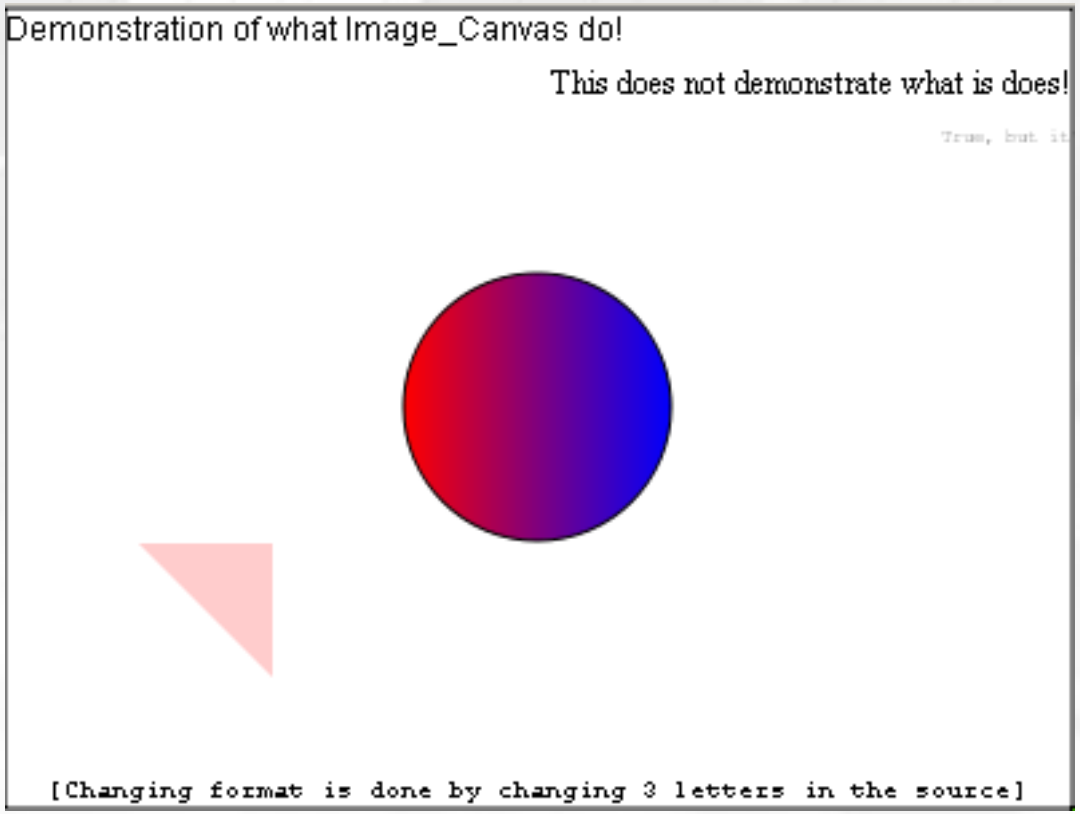
```
<ellipse cx="199" cy="149" rx="50" ry="50"
style="stroke-width:1;stroke:rgb(0,0,0);
fill:url(#gradient_1);"/>
<text x="0" y="12" style="font-
family:Arial;font-
size:12px;fill=rgb(0,0,0);">Demonstration of
what Image_Canvas do!</text>

...

<image xlink:href="
iVBORw0..." x="398" y="298" width="32"
height="32" preserveAspectRatio="none"/>
```

SVG and PHP

PEAR::Image_Canvas – image:



PEAR::Image_Graph

- Drawing graphs in various ways
- Object-oriented, flexible, extendable
- Since v0.5. uses PEAR::Image_Canvas as „driver model“ for output
  - GD (PNG, JPEG, GIF, WBMP)
  - Scalable Vector Graphics (SVG)
  - PDF (using PDFLib)
- Optional: Data-preprocessors packages
  - 'Numbers_Words' and 'Numbers_Roman'

PEAR::Image_Graph

- Use factory-methods
  - Not required – but allows “lazy includes“

```
require_once 'Image/Graph.php';
require_once 'Image/Canvas.php';
$Canvas =& Image_Canvas::factory('svg',
  array('width' => 600, 'height' => 400));
$Graph =& Image_Graph::factory('graph', $Canvas);

// instead of

$Graph =& new Image_Graph($Canvas);

// ...
```

PEAR::Image_Graph

- Mind the ampersand (call by reference)
  - Otherwise problems with modifying objects

```
// ...
$Plot =& $Graph->addNew
         ('line', &$Dataset);

// without ampersand the following
// would be impossible

$Plot->setLineColor('red');
// ...
```

PEAR::Image_Graph

- Adding elements to parents

```
// ...
$newElement =& Image_Graph::factory(
    'some_element',
    array($param1, $param2,  &$param3)
  );
$parentElement->add($newElement);

// better instead do:

$parentElement->addNew('some_element',
  array($param1, $param2, &$param3));
// ...
```

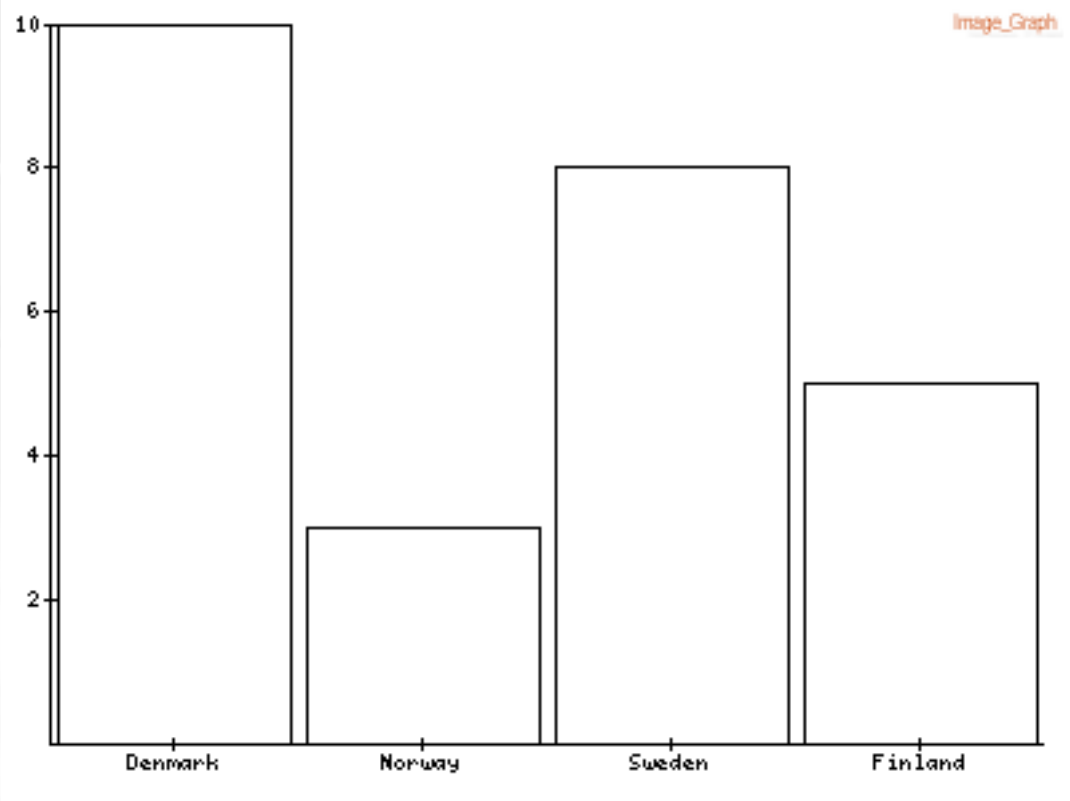
PEAR::Image_Graph

Components of a graph:

- graph
- plotarea (to create the plots on)
- plot
- dataset (data to plot)
- additional elements (e.g. legend)

PEAR::Image_Graph

Simple example:



PEAR::Image_Graph

Simple example:

```
require_once 'Image/Graph.php';
require_once 'Image/Canvas.php';
$Canvas =& Image_Canvas::factory('svg',
  array('width' => 600, 'height' => 400));
$Graph =& Image_Graph::factory('graph', $Canvas);
$Plotarea =& $Graph->addNew('plotarea');
$Dataset =& Image_Graph::factory('dataset');
$Dataset->addPoint('Denmark', 10);
$Dataset->addPoint('Norway', 3);
$Dataset->addPoint('Sweden', 8);
$Dataset->addPoint('Finland', 5);
$Plot =& $Plotarea->addNew('bar',&$Dataset);
$Graph->done();
```

PEAR::Image_Graph

Plotarea:

- Holds axis-layout
- Container for plots
- Standard-behaviour:
  - X-axis textual ('Denmark','Sweden', ...)
  - Y-axis linear
- Types:
  - 'axis' (linear), 'axis_log' (logarithmic),
  - 'Image_Graph_Axis_Category' (textual)

PEAR::Image_Graph

Graph types:

- 'line', 'area', 'bar', 'smooth_line', 'smooth_area', 'pie', 'step', 'impulse', 'dot' or 'scatter', 'radar', Image_Graph_Plot_CandleStick, Image_Graph_Plot_Band

Special modes:

- 'normal', 'stacked', 'stacked100pct'

PEAR::Image_Graph

1. Adding a plotarea:

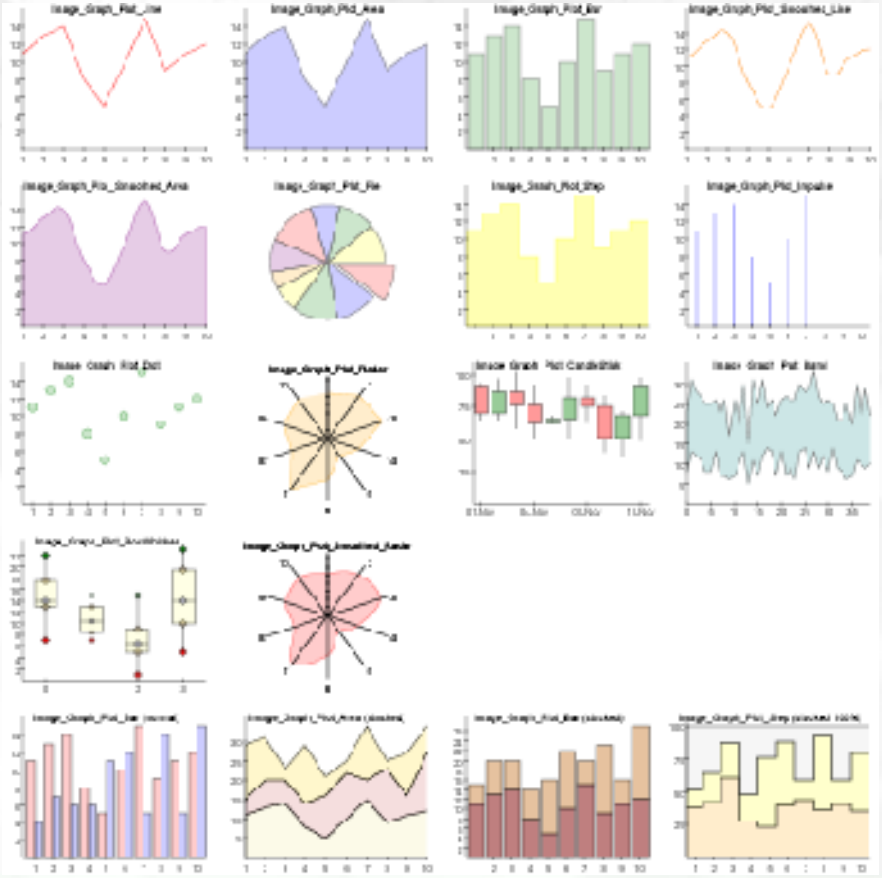
```
$Plotarea =& $Graph->addNew('plotarea',
                array('axis', 'axis_log'));
```

2. Adding a bar-plot:

```
// ...
$Plot =& $Plotarea->addNew('bar', $Dataset);
// ...
```

PEAR::Image_Graph

Plottype-examples:



PEAR::Image_Graph

Data for plots:

- Directly giving points

```
$Dataset =& Image_Graph::factory('dataset');
$Dataset->addPoint('Denmark', 10);
$Dataset->addPoint('Norway', 3);
```

- Function callback

```
function foo($bar) {
  return 2 * $bar + 10;
}
// 100 values between -3 and 10
$Dataset =& Image_Graph::factory
    ('function', array(-3, 10, 'foo', 100));
```

PEAR::Image_Graph

Using colors:

- Color names (e.g. 'green') or RGB ('#00ff00', '75%,20%,19%', array(0, 255, 0)
- Optionally opacity ('red@0.2', '#00ff00@0.9', array(0, 255, 0, 0.2))

```
$element->setLineColor('red');
$element->setFillColor('#0000ff@0.1');
$element->setBackgroundColor('green@0.1');
$element->setBorderColor(array(0, 0, 0));
```

PEAR::Image_Graph

Fillstyles:

- Simple fill
- Gradient fill
  - Vertical, horizontal, diagonally, radial

```
$fill =& Image_Graph::factory('gradient',
  array(IMAGE_GRAPH_VERTICAL, 'white','red'));
$element->setFillStyle($fill);
```

- Image

```
$fill =& Image_Graph::factory(
  'Image_Graph_Fill_Image', './image.png');
$element->setFillStyle($fill);
```

PEAR::Image_Graph

Layouts:

- Horizontal, vertical, ...

```
// split 40% from left; A and B are plotareas
$Graph->add(Image_Graph::horizontal(
            $A,
            $B,
            40) );

// directly create plotareas as well
$Graph->add(Image_Graph::vertical(
   $part1 = Image_Graph::factory('plotarea'),
   $part3 = Image_Graph::factory('plotarea'),
   40 ) );
```

PEAR::Image_Graph

Layouts:

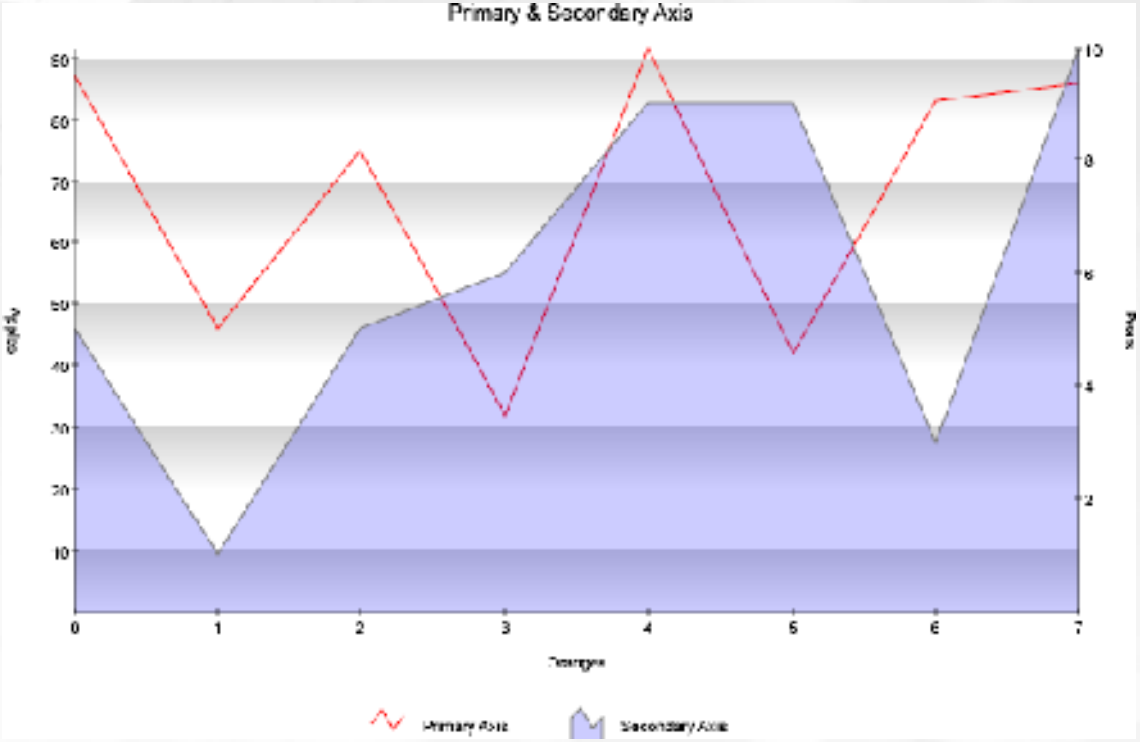
- ... and matrix

```
$Matrix =& $Graph->addNew
  ('Image_Graph_Layout_Matrix', array(2, 2));
$part1 =& $Matrix->getEntry(0, 0);
$part2 =& $Matrix->getEntry(0, 1);
$part3 =& $Matrix->getEntry(1, 0);
$part4 =& $Matrix->getEntry(1, 1);
```

PEAR::Image_Graph

<u>More complex example:</u>

- TrueType-font, title, plotarea, legend
- Vertical layout
- Gradient fill
- Two y-axes
- Axes-titles



PEAR::Image_Graph

More complex example (continued):

```
require_once 'Image/Graph.php';
require_once 'Image/Canvas.php';
$Canvas =& Image_Canvas::factory('svg',
  array('width' => 600, 'height' => 400));
$Graph =& Image_Graph::factory('graph', $Canvas);
$Font =& $Graph->addNew('ttf_font', 'Gothic');
$Font->setSize(8);
$Graph->setFont($Font);
```

PEAR::Image_Graph

More complex example (continued):

```
$Graph->add(Image_Graph::vertical(
 Image_Graph::factory('title',
   array('Primary & Secondary Axis', 11)),
 Image_Graph::vertical(
  $Plotarea = Image_Graph::factory('plotarea'),
  $Legend =Image_Graph::factory('legend'),90),5));
$Legend->setPlotarea($Plotarea);

$GridY2 =& $Plotarea->addNew('bar_grid',
  IMAGE_GRAPH_AXIS_Y_SECONDARY);
$GridY2->setFillStyle(Image_Graph::factory(
  'gradient', array(IMAGE_GRAPH_GRAD_VERTICAL,
  'white', 'lightgrey') ) );
```

PEAR::Image_Graph

More complex example (continued):

```
$Dataset1 =& Image_Graph::factory('random',
  array(8, 10, 100, true));
$Plot1 =& $Plotarea->addNew('line', &$Dataset1);
$Plot1->setLineColor('red');
$Dataset2 =& Image_Graph::factory('random',
  array(8, 1, 10, true));
$Plot2 =& $Plotarea->addNew
  ('Image_Graph_Plot_Area', $Dataset2,
   IMAGE_GRAPH_AXIS_Y_SECONDARY);
$Plot2->setLineColor('gray');
$Plot2->setFillColor('blue@0.2');
$Plot1->setTitle('Primary Axis');
$Plot2->setTitle('Secondary Axis');
```

PEAR::Image_Graph

More complex example (continued):

```
$AxisX =& $Plotarea->getAxis(IMAGE_GRAPH_AXIS_X);
$AxisX->setTitle('Oranges');
$AxisY =& $Plotarea->getAxis(IMAGE_GRAPH_AXIS_Y);
$AxisY->setTitle('Apples', 'vertical');
$AxisYsecondary =& $Plotarea->getAxis(
  IMAGE_GRAPH_AXIS_Y_SECONDARY);
$AxisYsecondary->setTitle('Pears', 'vertical2');

// output the Graph
$Graph->done();
```

PEAR::Image_Graph

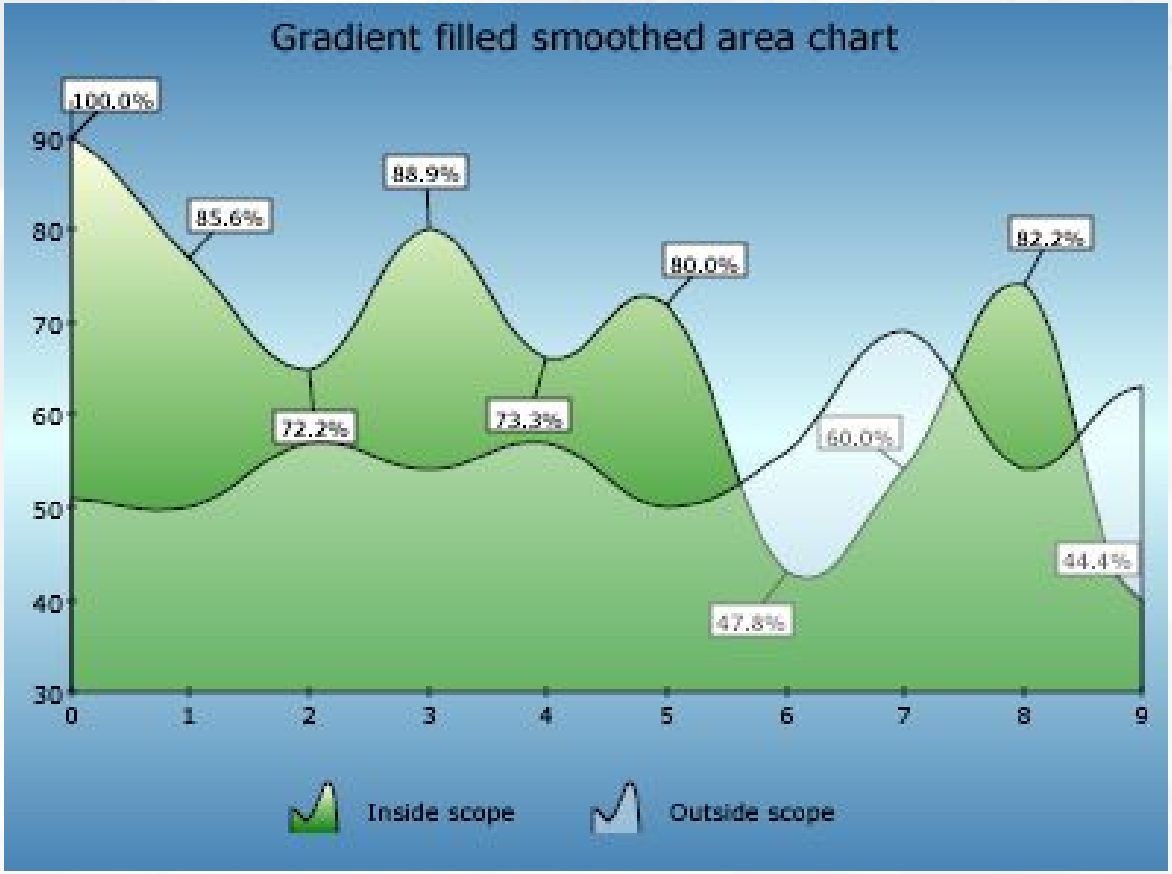
Data preprocessors:

- Adjustment / translation of data (e.g. Axis labels)

```
$AxisX =& $Plotarea->getAxis(IMAGE_GRAPH_AXIS_X);
$AxisX->setDataPreprocessor(Image_Graph::factory
  ('Image_Graph_DataPreprocessor_Array',
   array(array(1 => '30 Jul',
               2 => '31 Jul',
               3 => '1 Aug',
               4 => '2 Aug'))));
$AxisY =& $Plotarea->getAxis(IMAGE_GRAPH_AXIS_Y);
$AxisY->setDataPreprocessor(Image_Graph::factory
  ('Image_Graph_DataPreprocessor_Formatted',
   '+ %0.1f%%'));
```

PEAR::Image_Graph

Another impression:



PEAR::Image_Graph

- Flexible graph-creation
- Multiple outputs, including SVG

However please note:

- Image_Graph still a moving target
- SVG-support might still need work
- Looking forward to your feedback!

SVG ready for use?

Yes ...

- Future for small, but high quality image
  - Zoomable, accessible, ...
- Text-based transformations of data
- Tools in place for using SVG (editors, ...)
- Ready for use in clear enviroments (browser / plugins installed)
- Migration possible (server-based rasterisation, ...)

SVG ready for use?

But ...

- Advanced features still not
  - supported everywhere
  - giving same result everywhere
- Some “mainstream” tools lacking support
  - especially OpenOffice.org (but under discussion)

SVG ready for use?

Your impression?

Links

- W3C Spec:
  http://www.w3.org/TR/SVG11/
- W3C about accessiblity:
  http://www.w3.org/TR/SVG11/access.html
- Viewers:
  http://www.w3.org/Graphics/SVG/SVG-Implementations
- Adobe Indepth FAQ:
  http://www.adobe.com/svg/indepth/faq.html
- SVG Foundation:
  http://www.svgfoundation.org/
- http://www.zvon.org/xxl/svgReference/Output/index.html
  http://www.zvon.org/xxl/SVGReferenceSVG/Output/intro.svg

Links

- Adobe SVG Viewer (Windows, MacOS, Linux, Solaris)
  http://www.adobe.com/svg/viewer/install/
- Corel SVG viewer (Windows)
  No longer supported!
  http://www.corel.com/servlet/Satellite?pagename=Corel/Downloads/Details&id=1047022177437
- Apache Squiggle (part of Batik toolkit)
  http://xml.apache.org/batik/svgviewer.html
  Java Webstart for Batik-demo:
  http://nagoya.apache.org/batik_1.1/batikNagoya.jnlp
- Mozilla SVG
  http://www.mozilla.org/projects/svg/
- Amaya (W3C's editor/browser)
  http://www.w3.org/Amaya/

Links

- Free editors:
  - Skencil (Linux)
    http://www.nongnu.org/skencil/
  - Inkscape (Windows, MacOS, Linux)
    http://www.inkscape.org/
- Content:
  - Samples:
    http://www.croczilla.com/svg/samples/
  - Cliparts:
    http://www.openclipart.org/

Links

- PHP:
  - PEAR-packages
    http://pear.php.net/
    Image_GIS, Image_Graph,
    XML_SVG, Image_Canvas,
    XML_image2svg, XML_svg2image
  - Lots of Image_Graph examples (visual and code):
    http://pear.veggerby.dk
  - SVG class from PHP Classes
    http://www.phpclasses.org/browse/package/457.html

Thank you!

Up-to-date slides available at:
http://talks.speedpartner.de/

Questions?
neufeind (at) speedpartner.de