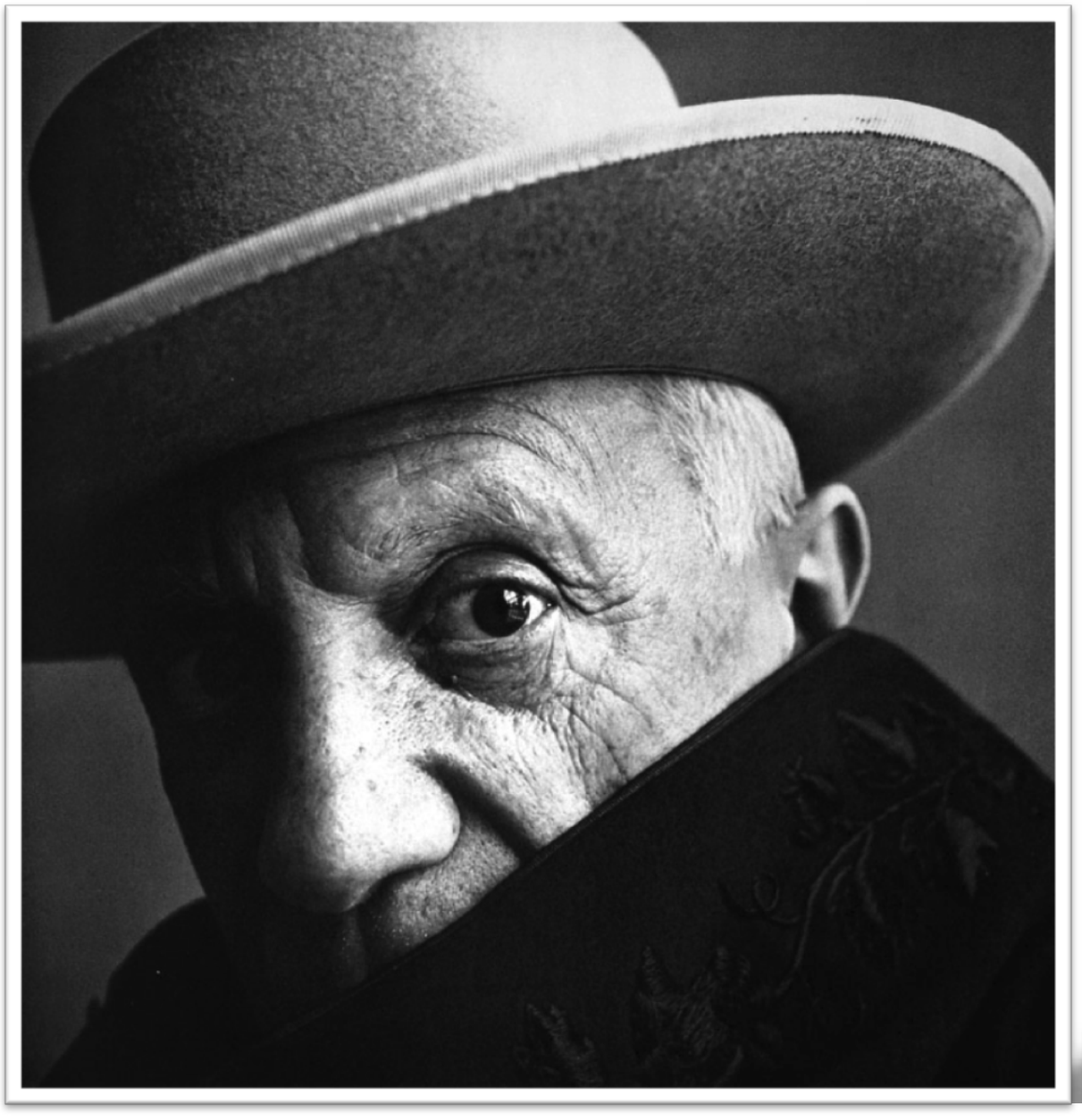
The other side of Go

Programming Pictures

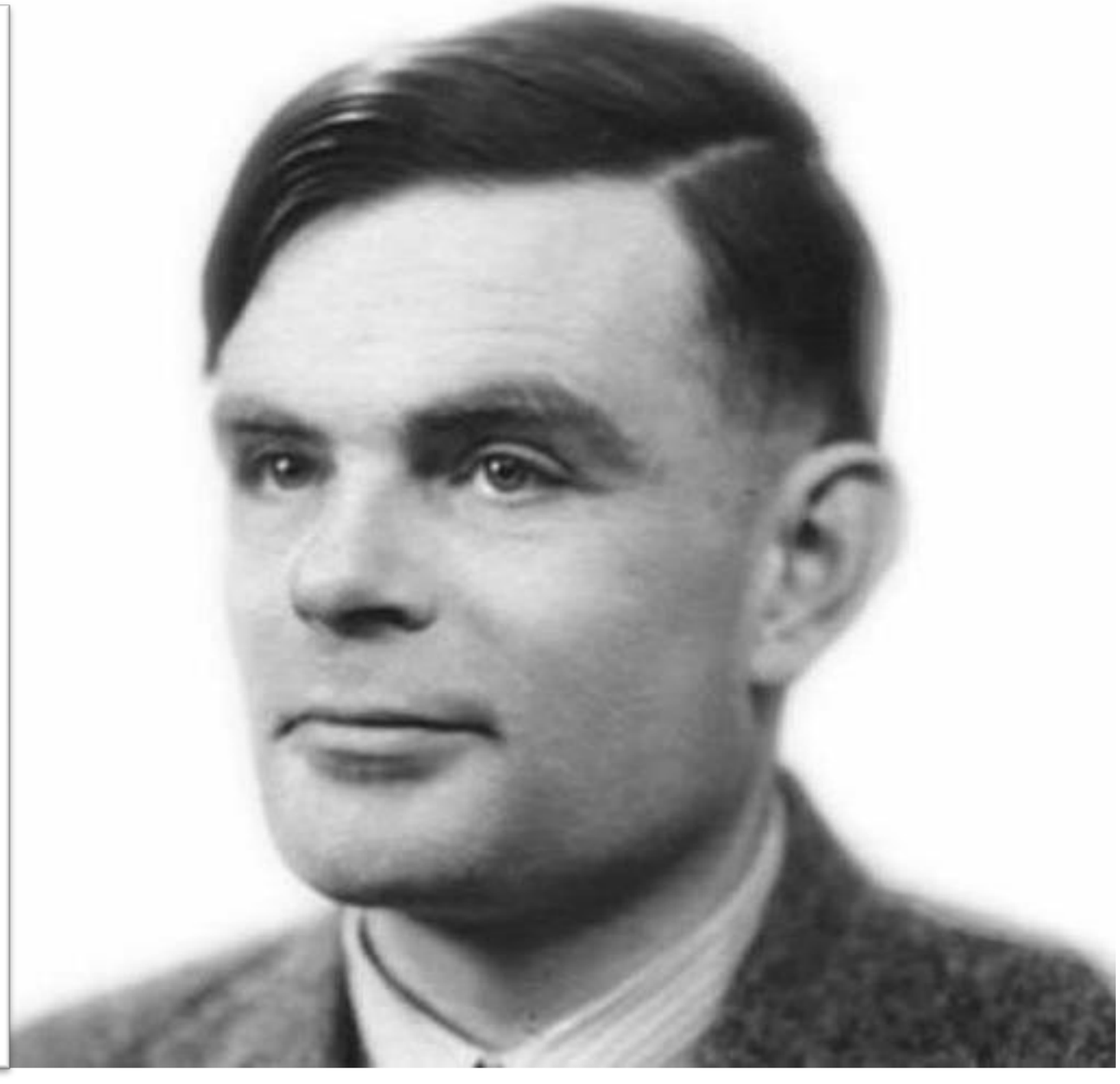
Anthony Starks

Go is great in the back end


But sometimes it's about the picture



Picasso



Turing

API Design

Client Program Design

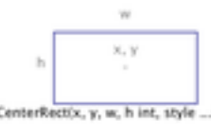
Visual Design and Relationships


SVG

SVG Go Library

github.com/ajstarks/svgo

Rect(x, y, w, h int, style ...string)



CenterRect(x, y, w, h int, style ...string)

Roundrect(x, y, w, h, rx, ry int, style ...string)

Square(x, y, w int, style ...string)

Line(x1, y1, x2, y2 int, style ...string)

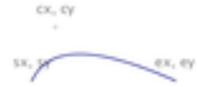
Polyline(x, y []int, style ...string)

Polygon(x, y []int, style ...string)

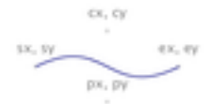
Circle(x, y, r int, style ...string)

Ellipse(x, y, rx, ry int, style ...string)

Arc(sx, sy, ax, ay, r int, lflag, sflag bool, ex, ey int, style ...string)



Qbez(sx, sy, cx, cy, ex, ey int, style ...string)

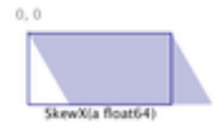


Bezier(sx, sy, cx, cy, px, py, ex, ey int, style ...string)

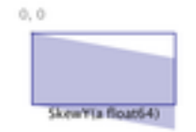
Translate(x, y int)

Scale(n float64)

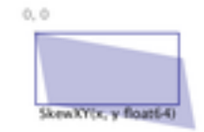
ScaleXY(x, y float64)



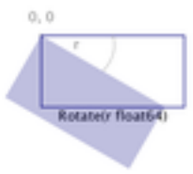
SkewX(a float64)



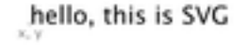
SkewY(a float64)



SkewXY(x, y float64)



Rotate(r float64)

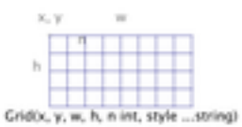


Text(x, y int, s string, style ...string)

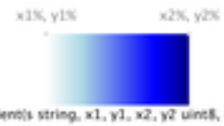
Textpath(s, pathid string, style ...string)

Path(s string, style ...string)

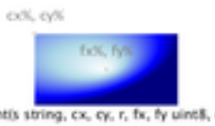
Image(x, y, w, h, int path string, style ...string)



Grid(x, y, w, h, n int, style ...string)



LinearGradient(s string, x1, y1, x2, y2 uint8, oc []Offcolor)



RadialGradient(s string, cx, cy, r, fx, fy uint8, oc []Offcolor)

RGB(r, g, b int)

RGBA(r, g, b int, opacity float64)

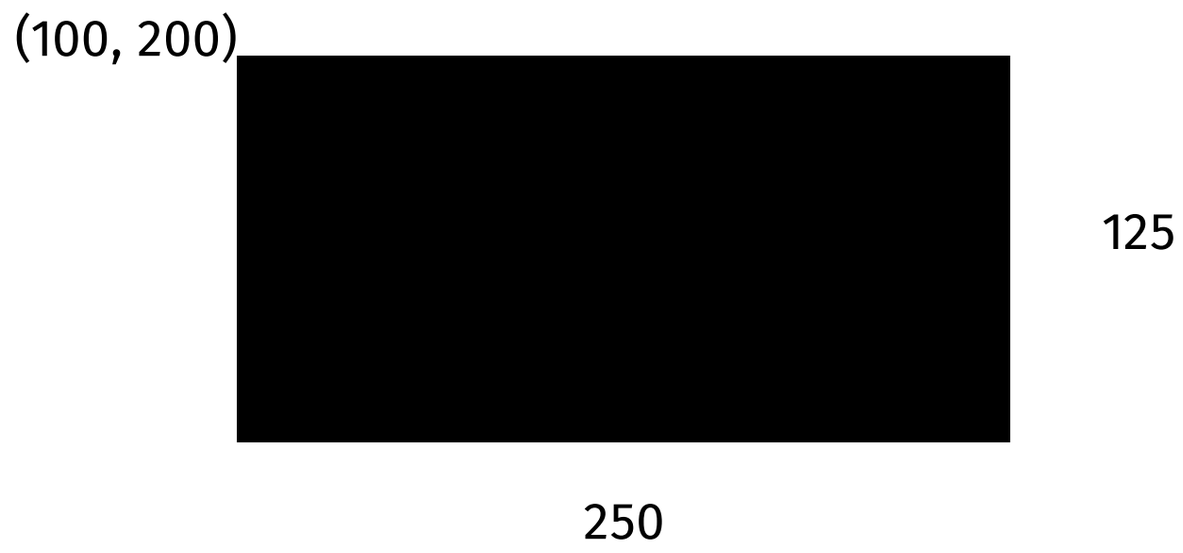
New(w io Writer)	specify destination
Start(w, h int, options ...string)/End()	begin/end the document
Startview(w, h, minx, miny, vw, vh int)	begin/end the document with viewport
Group(s ...string)/Gend()	begin/end group with attributes
Gstyle(s string)/Gend()	begin/end group style
Gtransform(s string)/Gend()	begin/end group transform
Gid(id string)/Gend()	begin/end group id
ClipPath(s ...string)/ClipEnd()	begin/end clip path
Def()/DefEnd()	begin/end a defintion block
Marker()/MarkerEnd()	begin/end markers
Pattern()/PatternEnd()	begin/end pattern
Desc(s string)	set the description element
Title(s string)	set the title element
Script(type, data ...string)	define a script
Mask(id string, x,y,w,h int, style ...string)/MaskEnd()	begin/end mask element
Link(href string, title string)/LinkEnd()	begin/end link to href, with a title
Use(x int, y int, link string, style ...string)	use defined objects

Textlines(x, y int, s []string, size, spacing int, fill, align string)

Element | Arguments

Rect (100,200,250,125)

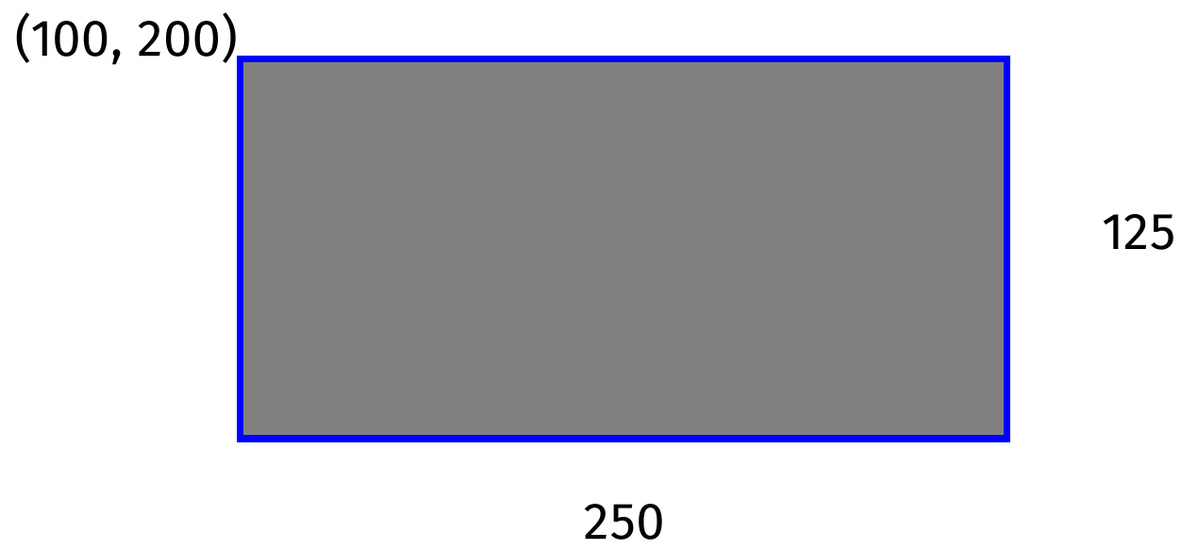
```
<rect x="100" y="200" width="250" height="125"/>
```



Element | Arguments | CSS Style

Rect (100,200,250,125, "fill:gray;stroke:blue")

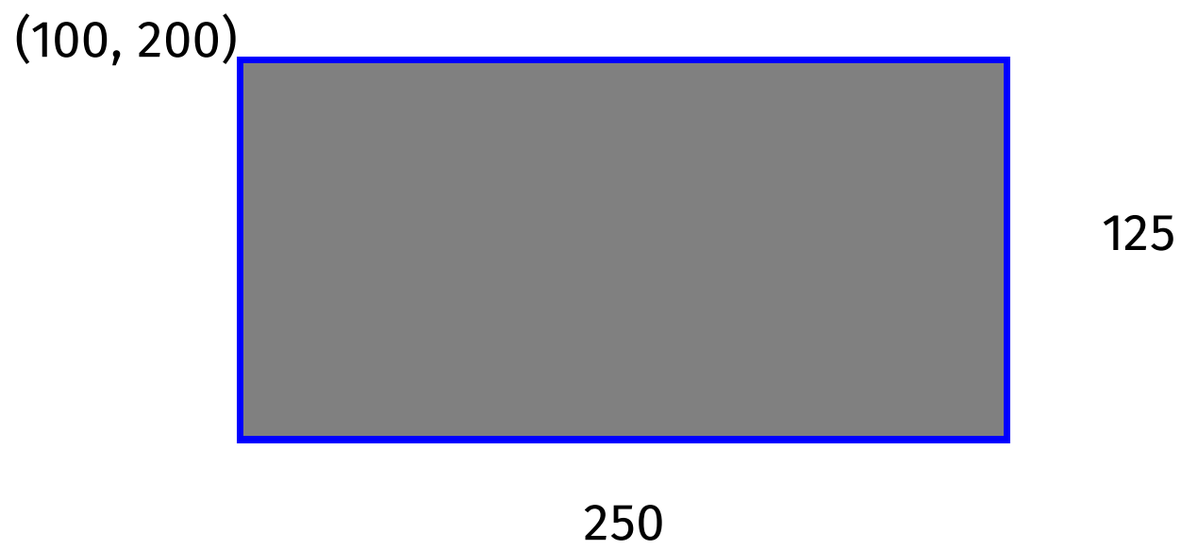
```
<rect x="100" y="200" width="250" height="125"
style="fill:gray;stroke:blue"/>
```

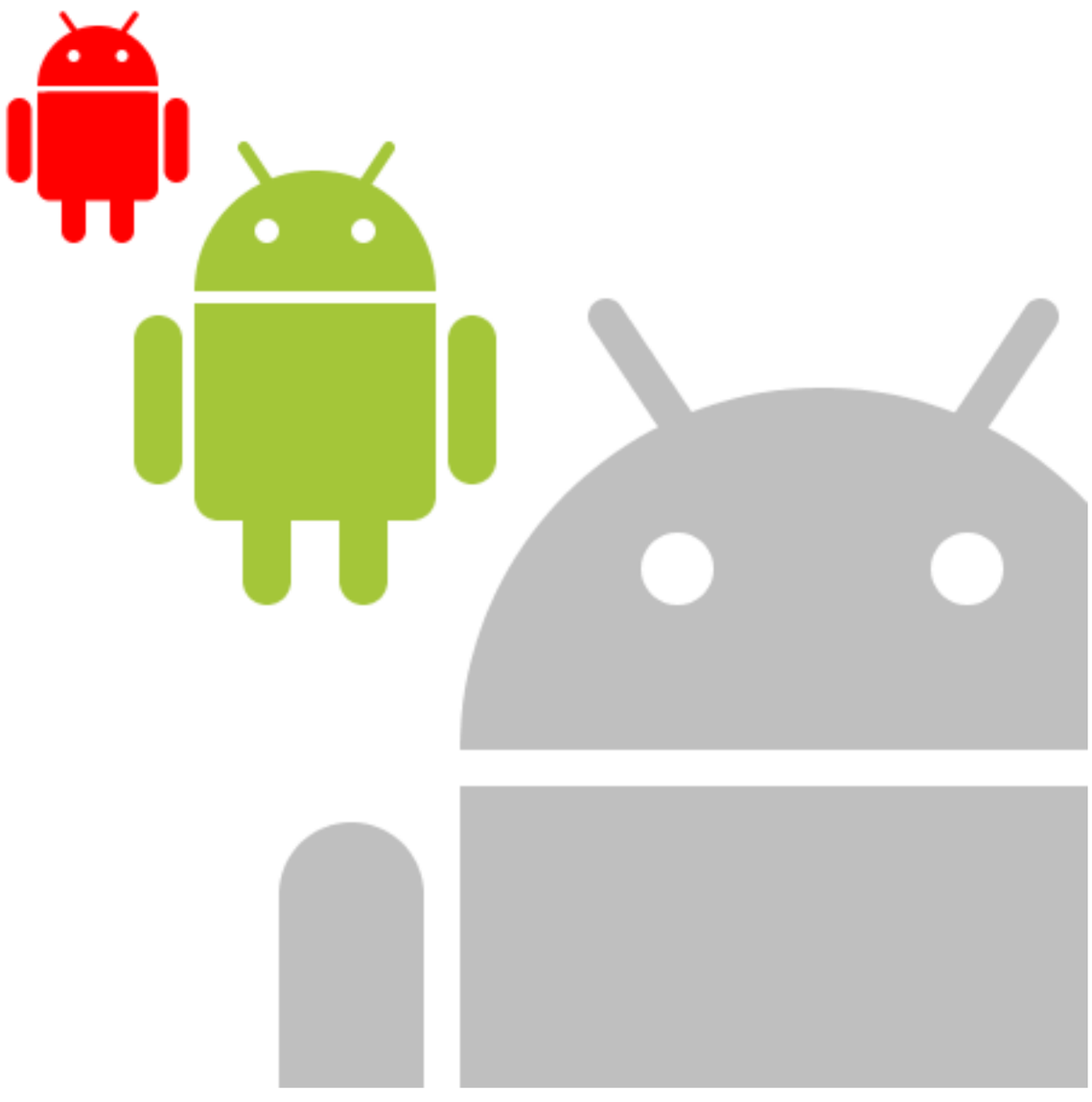


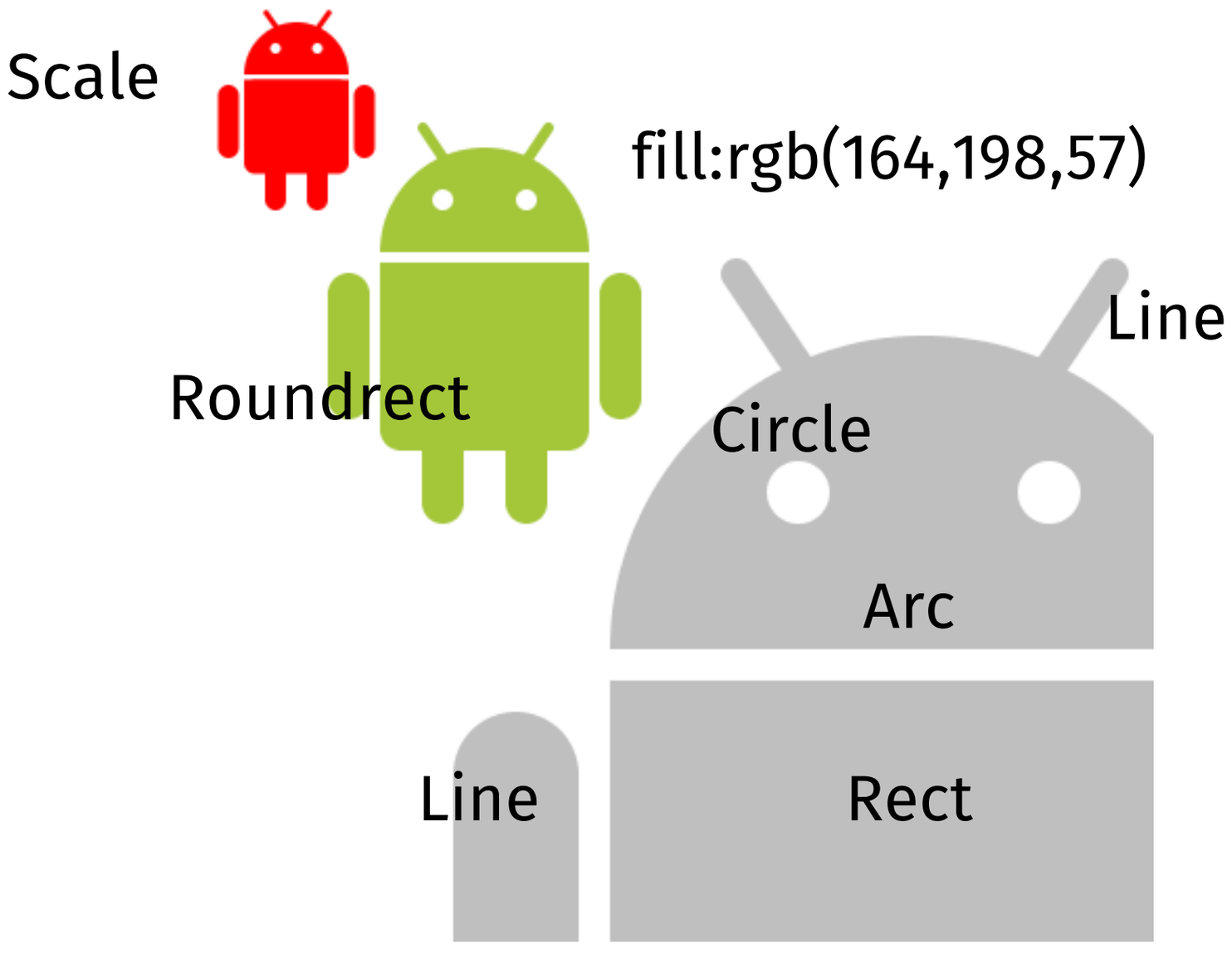
Element | Arguments | Attributes

```
Rect (100,200,250,125,
      `id="box"`, `fill="gray"`, `stroke="blue"`)
```

```
<rect x="100" y="200" width="250" height="125"
id="box" fill="gray" stroke="blue"/>
```

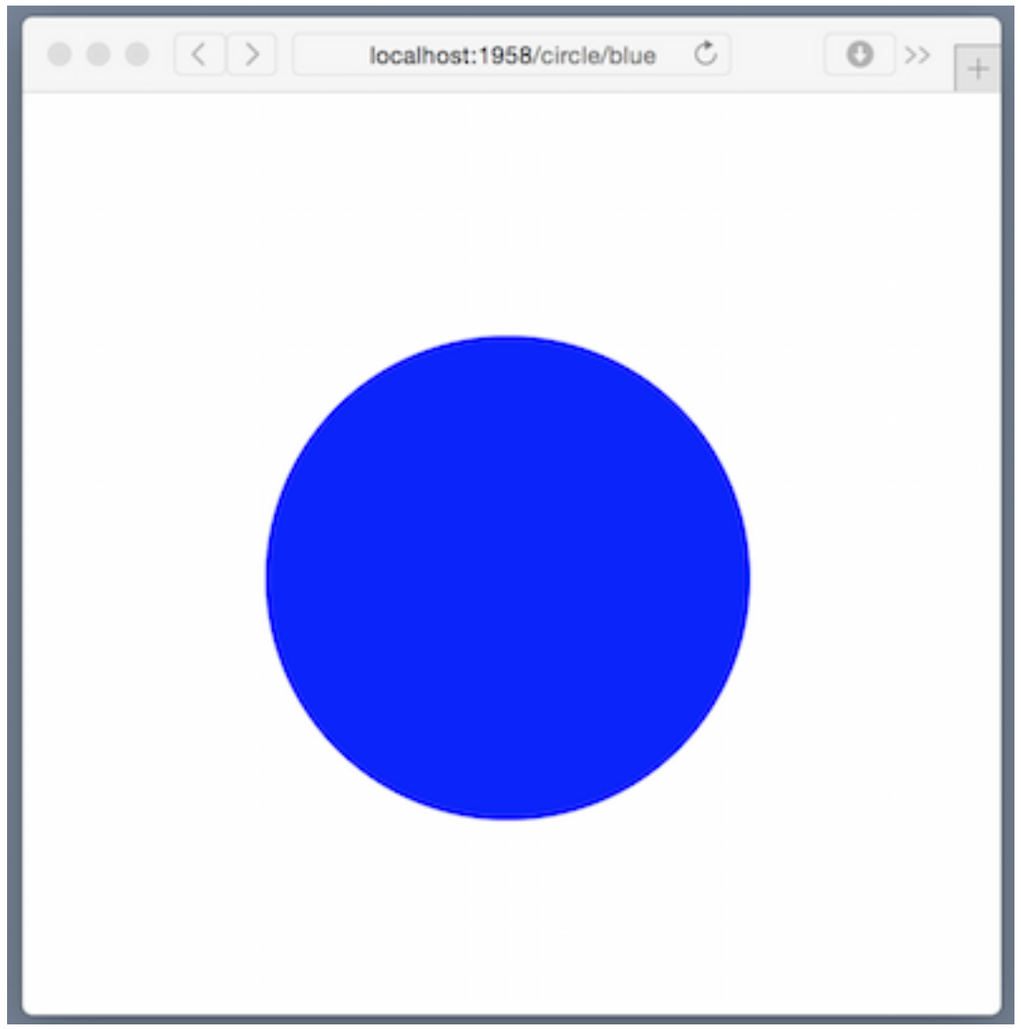





Scale
fill:rgb(164,198,57)
Roundrect
Line
Circle
Arc
Line
Rect

```
package main

import (
    "os"
    "github.com/ajstarks/svgo"
)

func main() {
    width := 600
    height := 338
    canvas := svg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height, "fill:black")
    canvas.Circle(width/2, height, width/2, "fill:rgb(44,77,232)")
    canvas.Text(width/2, height/2, "hello, world",
        "fill:white;font-size:60pt;font-family:serif;text-anchor:middle")
    canvas.End()
}
```

```
package main

import (
    "github.com/ajstarks/svgo"
    "log"
    "net/http"
    "strings"
)

const defaultstyle = "fill:rgb(127,0,0)"

func main() {
    http.Handle("/circle/", http.HandlerFunc(circle))
    err := http.ListenAndServe("localhost:1958", nil)
    if err != nil {
        log.Println("ListenAndServe:", err)
    }
}

func circle(w http.ResponseWriter, req *http.Request) {
    w.Header().Set("Content-Type", "image/svg+xml")
    s := svg.New(w)
    s.Start(500, 500)
    s.Title("Circle")
    s.Circle(250, 250, 125, shapestyle(req.URL.Path))
    s.End()
}

func shapestyle(path string) string {
    i := strings.LastIndex(path, "/") + 1
    if i > 0 && len(path[i:]) > 0 {
        return "fill:" + path[i:]
    }
    return defaultstyle
}
```

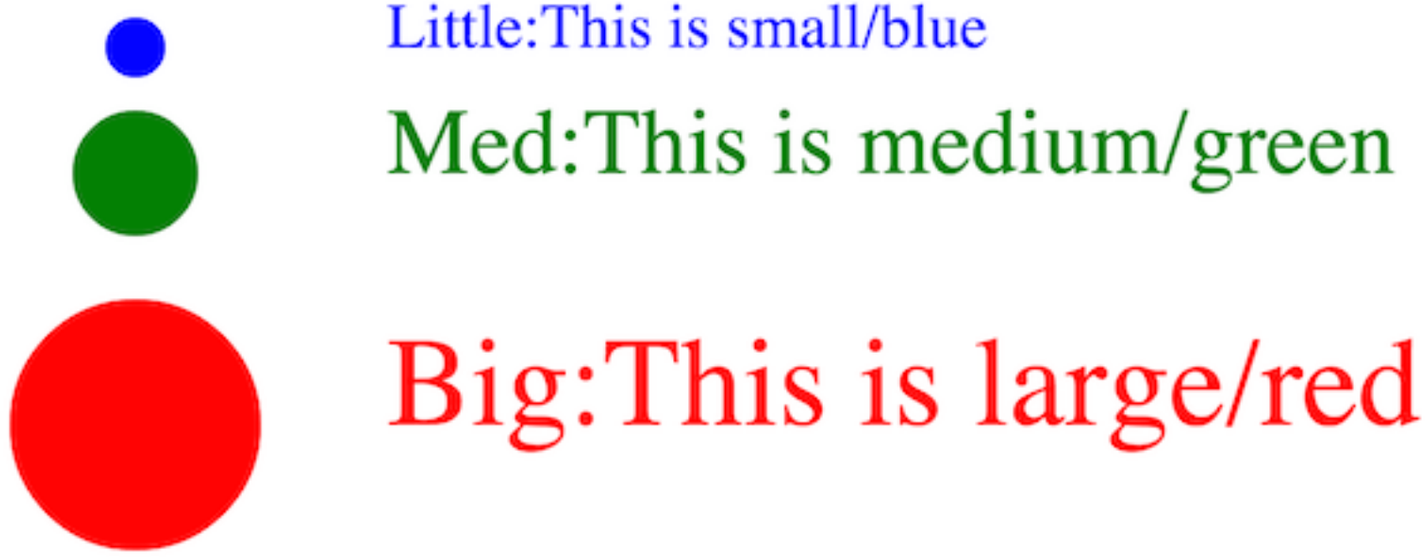


Read/Parse/Draw Pattern

Data

Picture

```
<thing top="100" left="100" sep="100">
   <item width="50"  height="50"  name="Little" color="blue">This is small</item>
   <item width="75"  height="100" name="Med"    color="green">This is medium</item>
   <item width="100" height="200" name="Big"    color="red">This is large</item>
</thing>
```



```
package main

// Imports
import (
    "encoding/xml"
    "flag"
    "fmt"
    "github.com/ajstarks/svgo"
    "io"
    "os"
)

type Thing struct {
    Top  int    `xml:"top,attr"`
    Left int    `xml:"left,attr"`
    Sep  int    `xml:"sep,attr"`
    Item []item `xml:"item"`
}

type item struct {
    Width  int    `xml:"width,attr"`
    Height int    `xml:"height,attr"`
    Name   string `xml:"name,attr"`
    Color  string `xml:"color,attr"`
    Text   string `xml:",chardata"`
}

var (
    width  = flag.Int("w", 1024, "width")
    height = flag.Int("h", 768, "height")
    canvas = svg.New(os.Stdout)
)

// Open the file
func dothing(location string) {
    f, err := os.Open(location)
    if err != nil {
        fmt.Fprintf(os.Stderr, "%v\n", err)
        return
    }
    defer f.Close()
    readthing(f)
}

// Read the file, loading the defined structure
func readthing(r io.Reader) {
    var t Thing
    if err := xml.NewDecoder(r).Decode(&t); err != nil {
        fmt.Fprintf(os.Stderr, "%v\n", err)
        return
    }
    drawthing(t)
}

// use the items of "thing" to make the picture
func drawthing(t Thing) {
    x := t.Left
    y := t.Top
    for _, v := range t.Item {
        style := fmt.Sprintf("font-size:%dpx;fill:%s", v.Width/2, v.Color)
        canvas.Circle(x, y, v.Height/4, "fill:"+v.Color)
        canvas.Text(x+t.Sep, y, v.Name+":"+v.Text+"/"+v.Color, style)
        y += v.Height
    }
}

func main() {
    flag.Parse()
    for _, f := range flag.Args() {
        canvas.Start(*width, *height)
        dothing(f)
        canvas.End()
    }
}
```

Imports

Define the input data structures

Define flags and output destination

Read the input

Parse and load the data structures

Draw

Main

Imports

```
import (
	"flag"
	"fmt"
	"github.com/ajstarks/svgo"
	"io"
	"os"
	"encoding/xml"
)
```

The main

```
func main() {
	flag.Parse()
	for _, f := range flag.Args() {
		canvas.Start(*width, *height)
		dothing(f)
		canvas.End()
	}
}
```

Define the input data structures

```
type Thing struct {
      Top  int `xml:"top,attr"`
      Left int `xml:"left,attr"`
      Sep  int `xml:"sep,attr"`
      Item []item `xml:"item"`
}
```

```
type item struct {
      Width  int    `xml:"width,attr"`
      Height int    `xml:"height,attr"`
      Name   string `xml:"name,attr"`
      Color  string `xml:"color,attr"`
      Text   string `xml:",chardata"`
}
```

```
<thing top="100" left="100" sep="100">
  <item width="50"  height="50"  name="Little" color="blue">This is small</item>
  <item width="75"  height="100" name="Med"    color="green">This is medium</item>
  <item width="100" height="200" name="Big"    color="red">This is large</item>
</thing>
```

Define flags and output destination

```
var (
	width  = flag.Int("w", 1024, "width")
	height = flag.Int("h", 768, "height")
	canvas = svg.New(os.Stdout)
)
```

Read the input

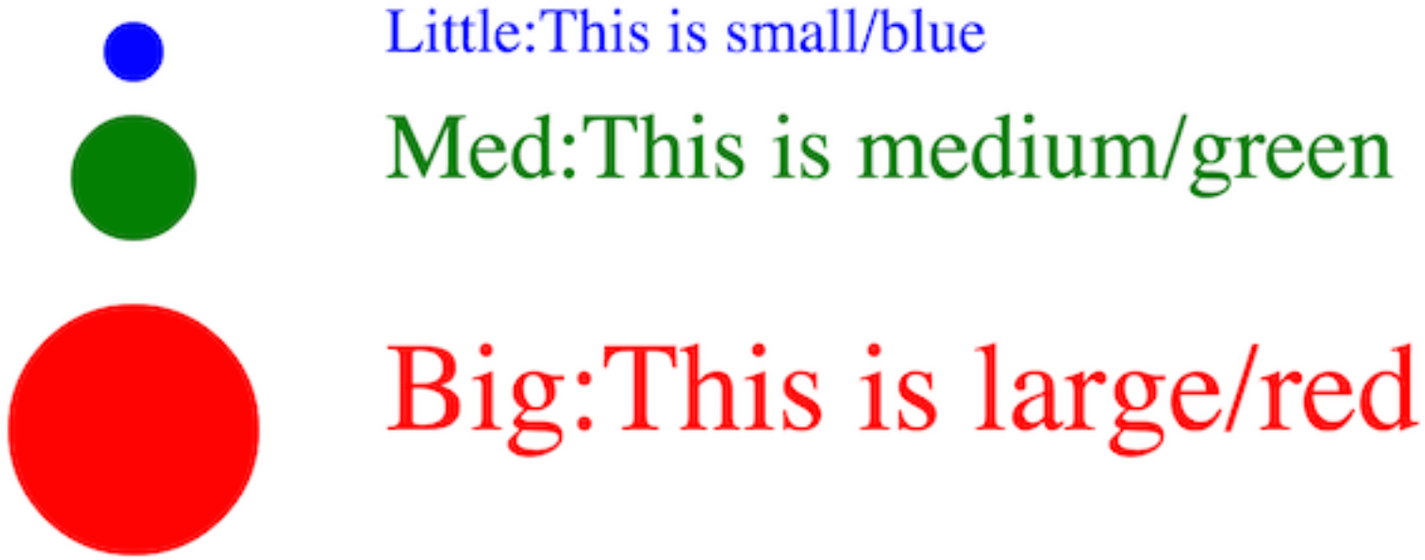
```
func dothing(location string) {
	f, err := os.Open(location)
	if err != nil {
		fmt.Fprintf(os.Stderr, "%v\n", err)
		return
	}
	defer f.Close()
	readthing(f)
}
```

Parse and load the data structures

```
func readthing(r io.Reader) {
	var t Thing
	if err := xml.NewDecoder(r).Decode(&t); err != nil {
		fmt.Fprintf(os.Stderr, "%v\n", err)
		return
	}
	drawthing(t)
}
```

Use the items of "thing" to make the picture

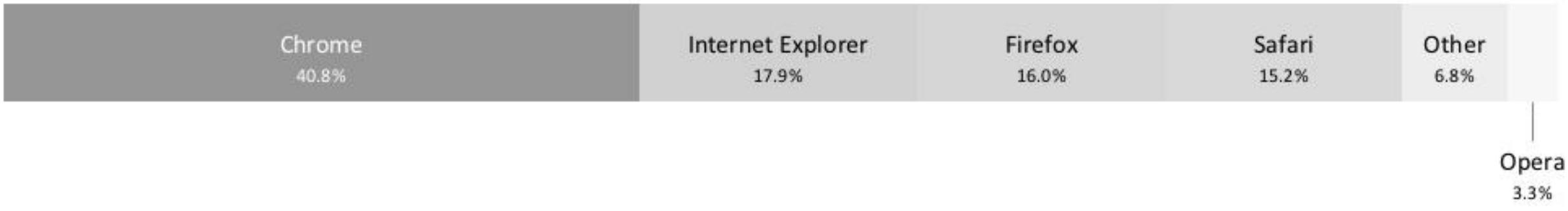
```
func drawthing(t Thing) {
      x := t.Left
      y := t.Top
      for _, v := range t.Item {
          style := fmt.Sprintf("font-size:%dpx;fill:%s", v.Width/2, v.Color)
          canvas.Circle(x, y, v.Height/4, "fill:"+v.Color)
          canvas.Text(x+t.Sep, y, v.Name+":"+v.Text+"/"+v.Color, style)
          y += v.Height
      }
}
```



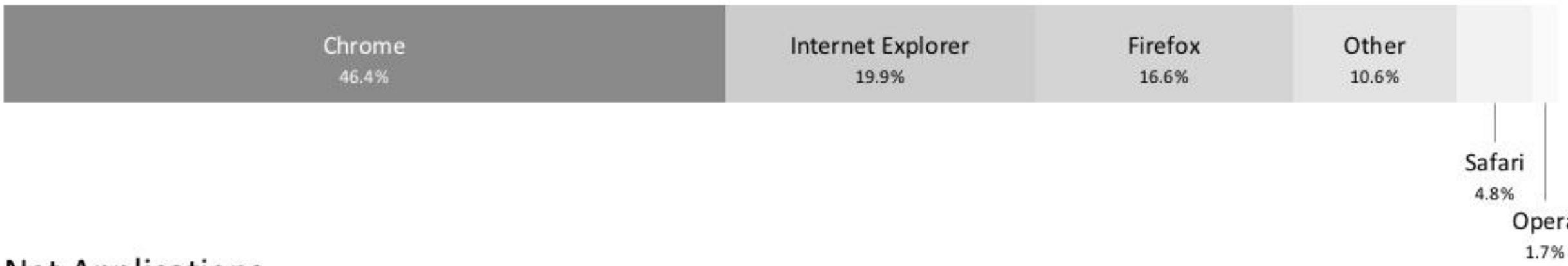
Browser Market Share

StatCounter

W3C Counter



Wikimedia

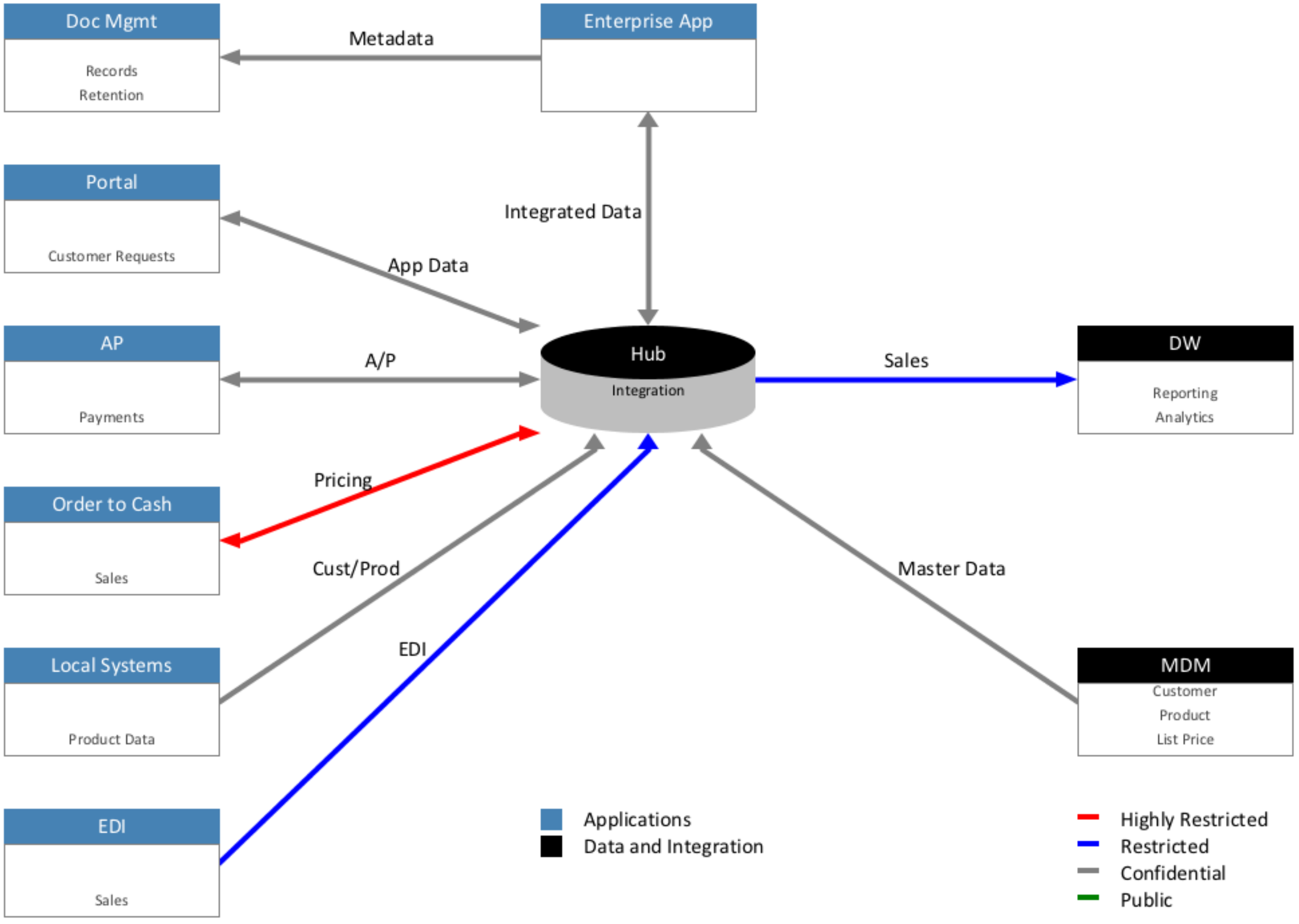


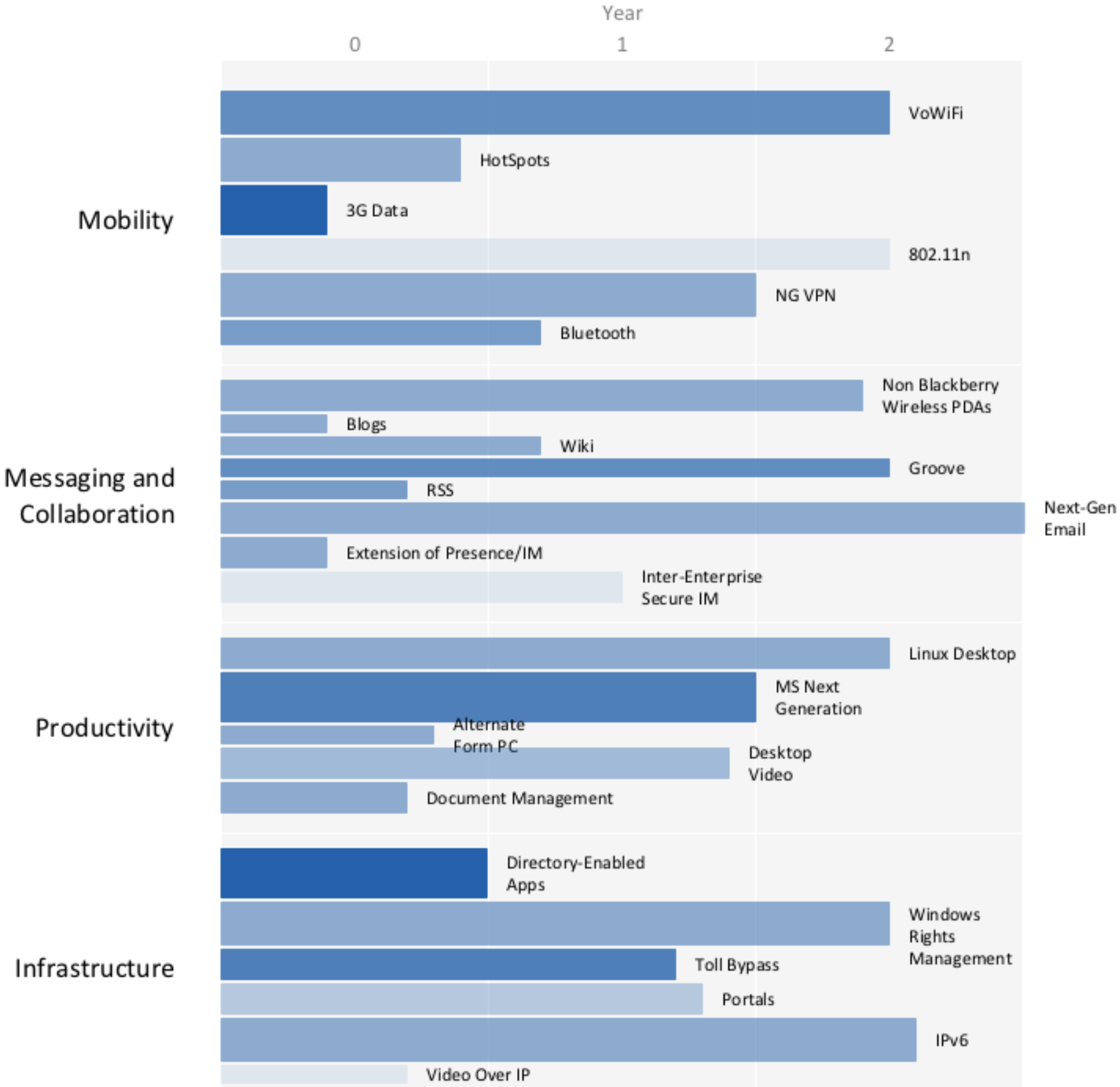
Net Applications

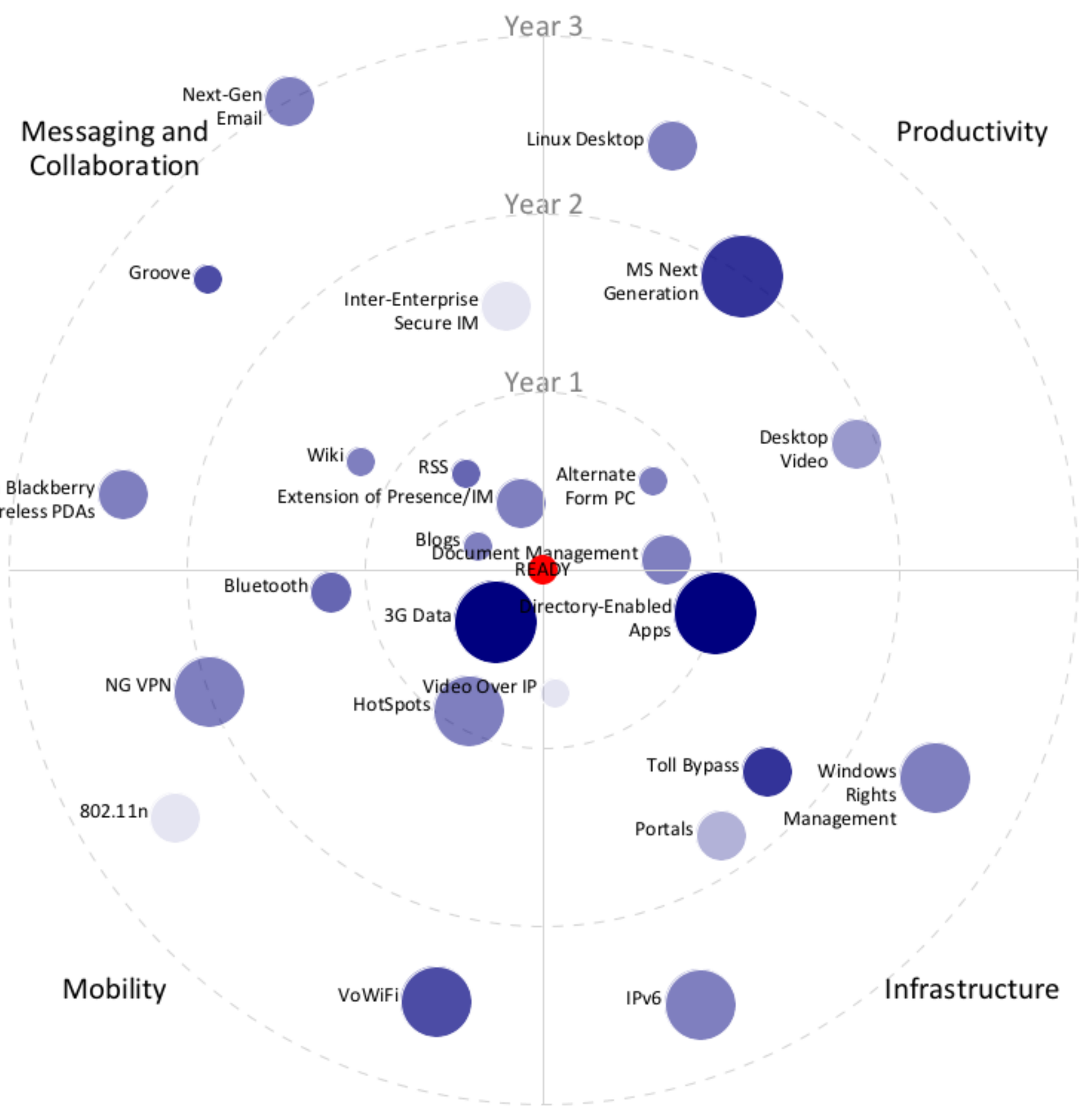
pmap -stagger -showtitle -p -t "Browser Market Share" bs.xml

```
<pmap>
  <pdata legend="StatCounter">
    <item value="51.3">Chrome</item>
    <item value="21.3">Internet Explorer</item>
    <item value="18.8">Firefox</item>
    <item value="5.1">Safari</item>
    <item value="2.1">Other</item>
    <item value="1.4">Opera</item>
  </pdata>
...
</pmap>
```

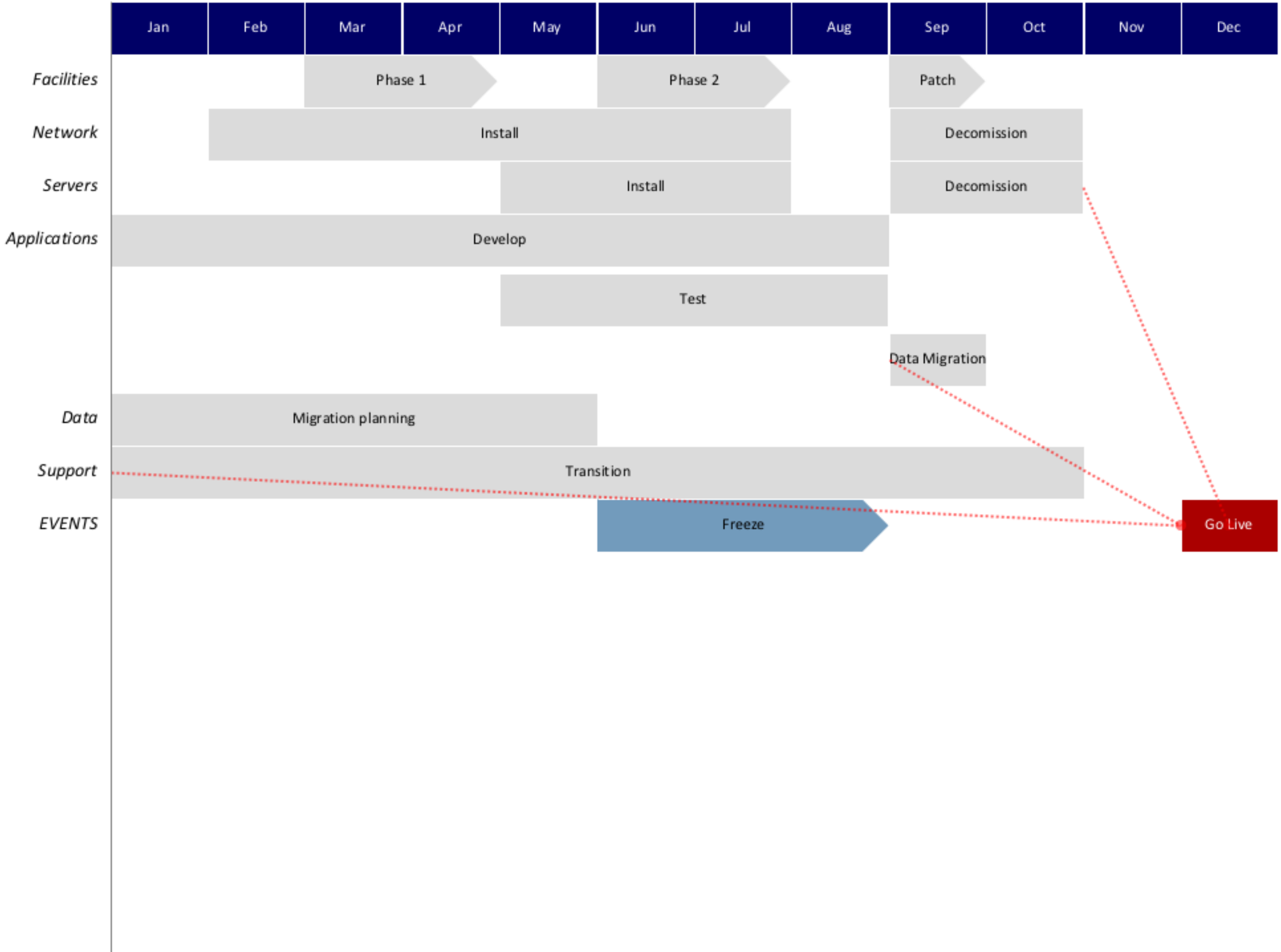
SVGo Clients


Doc Mgmt
Records
Retention
Metadata
Enterprise App
Portal
Customer Requests
App Data
Integrated Data
AP
Payments
A/P
Hub
Integration
Sales
DW
Reporting
Analytics
Pricing
Order to Cash
Sales
Cust/Prod
EDI
Local Systems
Product Data
Master Data
MDM
Customer
Product
List Price
EDI
Sales
Applications
Data and Integration
Highly Restricted
Restricted
Confidential
Public


Year
0
1
2
Mobility
VoWiFi
HotSpots
3G Data
802.11n
NG VPN
Bluetooth
Messaging and Collaboration
Non Blackberry Wireless PDAs
Blogs
Wiki
Groove
RSS
Next-Gen Email
Extension of Presence/IM
Inter-Enterprise Secure IM
Productivity
Linux Desktop
MS Next Generation
Alternate Form PC
Desktop Video
Document Management
Infrastructure
Directory-Enabled Apps
Windows Rights Management
Toll Bypass
Portals
IPv6
Video Over IP


Year 3
Year 2
Year 1
Messaging and Collaboration
Productivity
Mobility
Infrastructure
Next-Gen Email
Linux Desktop
Groove
Inter-Enterprise Secure IM
MS Next Generation
Non Blackberry Wireless PDAs
Wiki
RSS
Extension of Presence/IM
Alternate Form PC
Desktop Video
Blogs
Document Management
READY
Bluetooth
3G Data
Directory-Enabled Apps
NG VPN
Video Over IP
HotSpots
Toll Bypass
Windows Rights Management
802.11n
Portals
VoWiFi
IPv6

PLANNING TO PLAN



Project Timeline

Meetings

Initiation	KickOff	Checkpoint	Checkpoint	Checkpoint	Management Update	Checkpoint	Checkpoint	Checkpoint
08/01	08/09	08/16	08/23	08/30	09/06	09/13	09/20	09/27

Process/Policy

Draft Processes	Update Policy	Report Policy/Process	Define Business Case	Final Policy Process	Approval
08/16	08/23	09/06	09/13	09/27	10/01

Communication

Messages Defined	Communication Plan	Final Communications
08/30	09/13	09/27

Ability to Execute

Definite

Likely

Moderate

Unlikely

Thing 13

Item 16

Thing 9

Thing 1

Item 18

Thing 21

Thing 8

Thing 10

Thing 2

Thing 12

Thing 7

Item 15

Item 17

Item 20

Thing 6

Thing 3

Thing 14

Thing 4

Thing 5

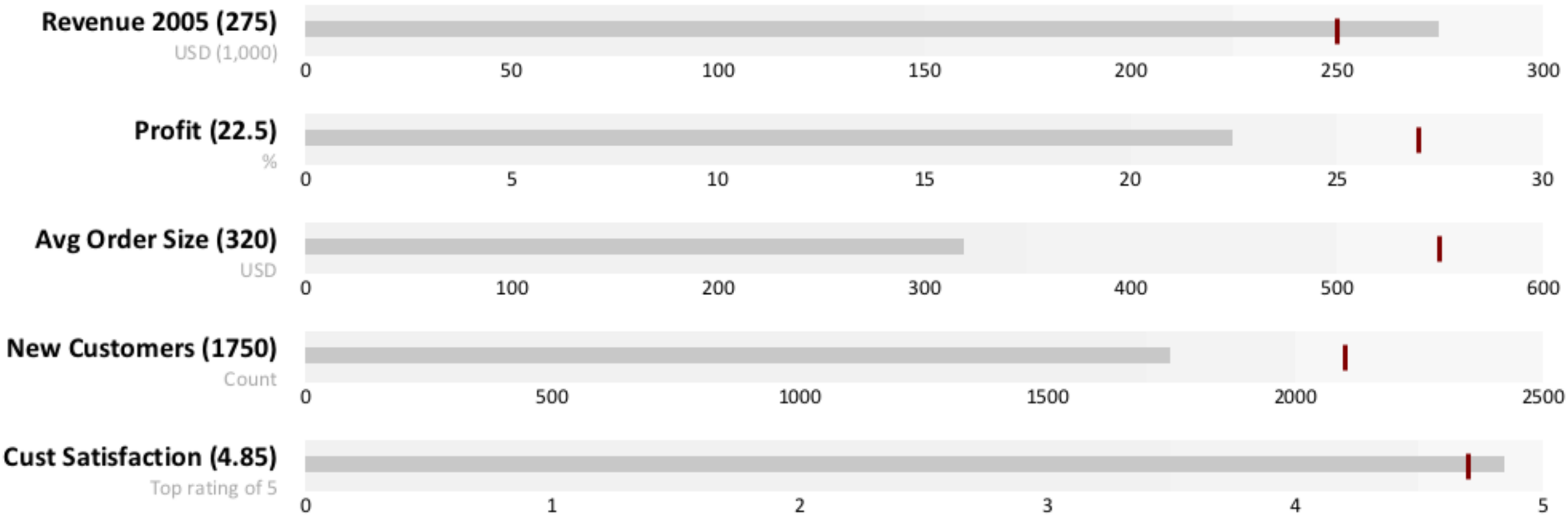
Weak

Moderate

Strong

Extereme

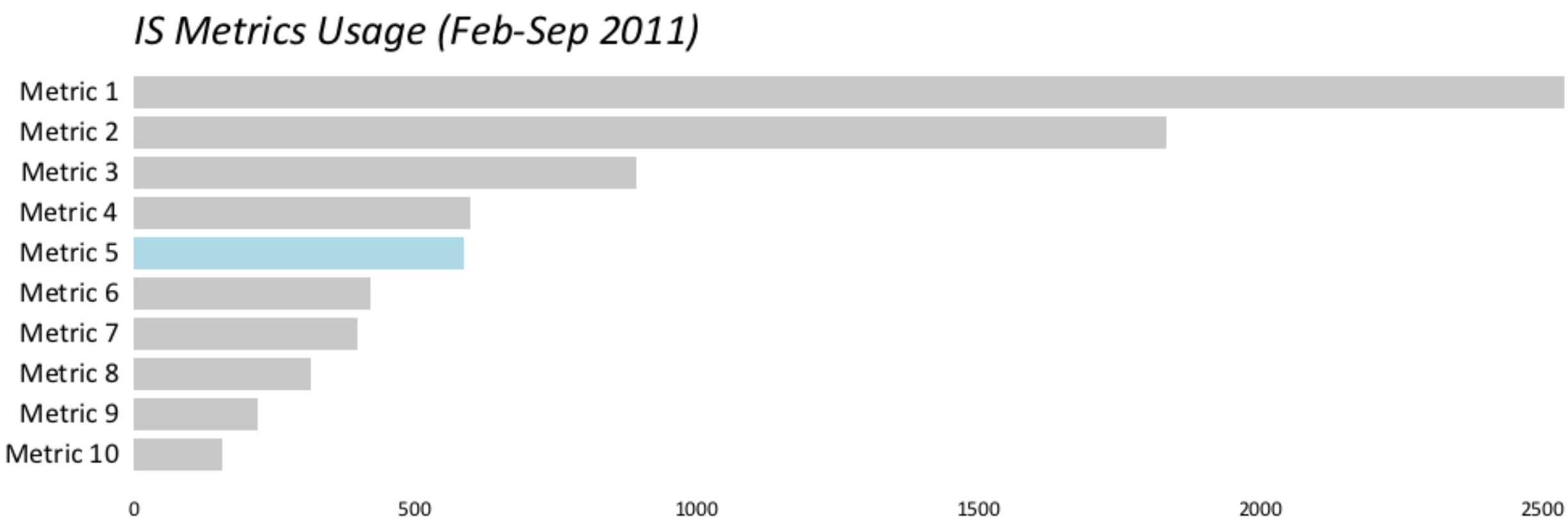
Strategic Alignment



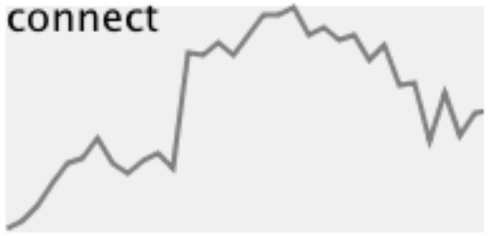
Sample Bullet Graph

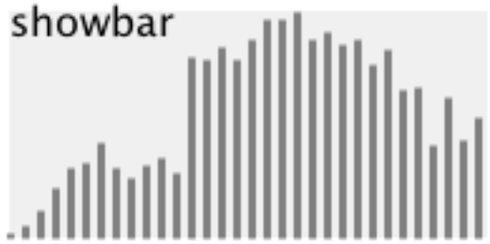
The bullet graph features a single, primary measure (for example, current year-to-date revenue) compares that measure to one or more other measures to enrich its meaning, for example, compared to a target), and displays it in the context of qualitative ranges of performance, such as poor, satisfactory, and good.

The qualitative ranges are displayed as varying intensities of a single hue to make them discernible by those who are color blind and to restrict the use of colors on the dashboard to a minimum.

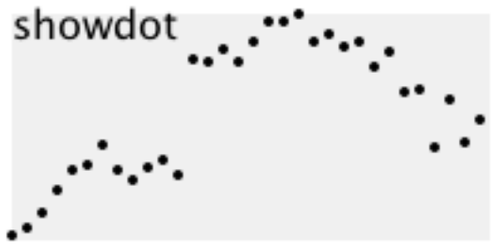

IS Metrics Usage (Feb-Sep 2011)
Metric 1
Metric 2
Metric 3
Metric 4
Metric 5
Metric 6
Metric 7
Metric 8
Metric 9
Metric 10
0
500
1000
1500
2000
2500

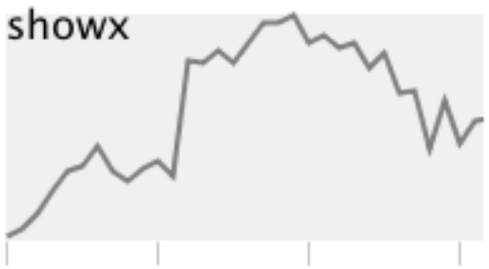
showbg


connect


showbar

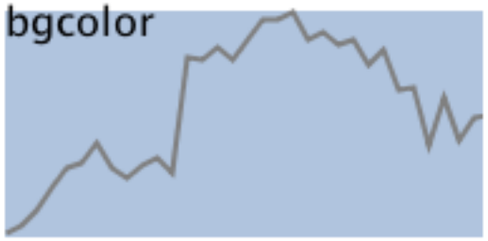

area

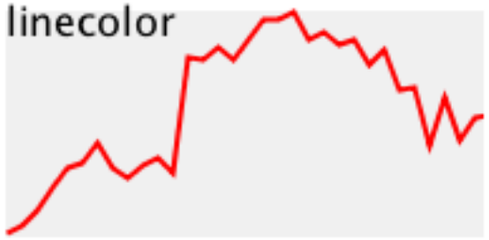

showdot

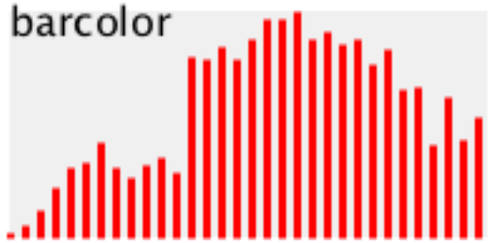

showx


623.0
589.8
556.5
523.2
490.0
showy

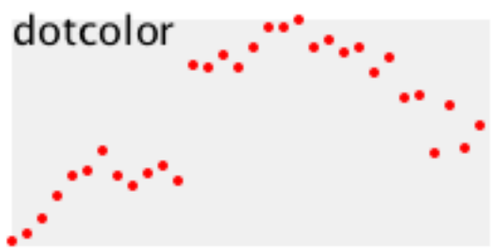

test.d


bgcolor


linecolor


barcolor


hcolor


dotcolor


labelcolor


fontsize


font

JONES
MONTIEL

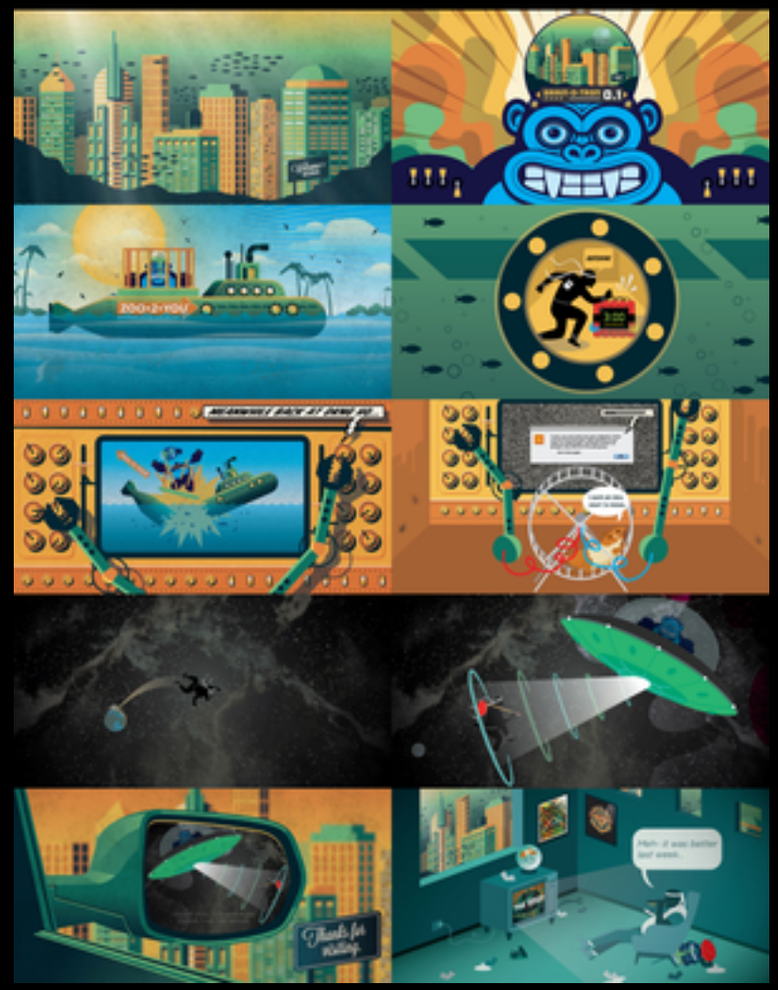
• LAYER TENNIS SEASON 4 WEEK 6 MATCH COMMENTARY BY MIKE MONTEIRO •

ANDERSON
DKNG

• LAYER TENNIS SEASON 4 PLAYOFF QUARTERFINAL COMMENTARY BY BRYAN BEDELL •

DKNG
DDL

• LAYER TENNIS SEASON 4 WEEK 7 MATCH COMMENTARY BY JOHN GRUBER •



CONTINO
CASSARO

PUTNAM
STOCKS

WHITE
TAYLOR

HALL
WARREN

STRAWBERRYLUNA
DOUBLENAUT

JONES
MONTIEL

DKNG
DDL

SHAWNA X
STEVENS

TAYLOR
JONES

PUTNAM
RAJKUMAR

REYES
WHITE

ANDERSON
DKNG

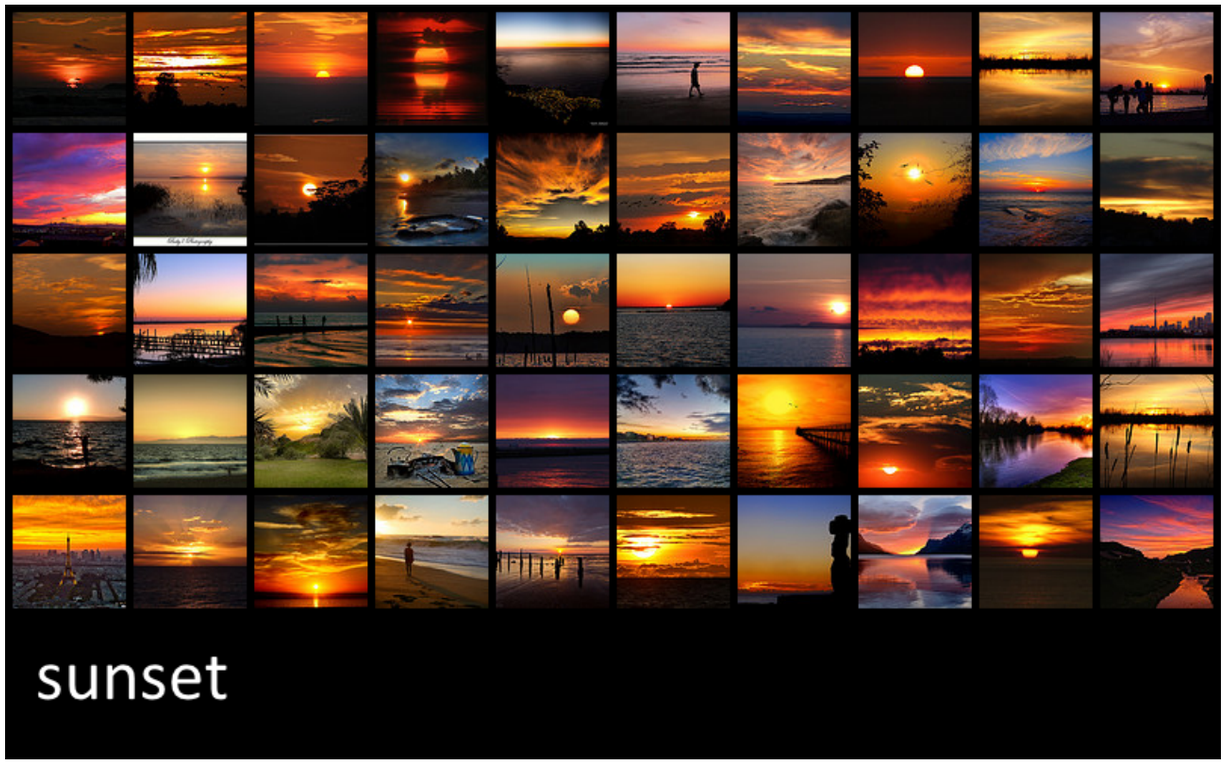
TAYLOR
WHITE

RAJKUMAR
ANDERSON

ANDERSON
WHITE

f50 sunset

https://api.flickr.com/services/rest/
?method=flickr.photos.search
&api_key=...
&text=sunset
&per_page=50
&sort=interestingness-desc




```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
   <photos page="1" pages="105615" perpage="50" total="5280747">
      <photo id="4671838925" ... secret="b070f3363e" server="4068" farm="5 ... />
      <photo id="3590142202" ... secret="c46752e4d8" server="2441" farm="3" .../>
      ...
 </photos>
</rsp>
```

Benchviz

ajstarks.org:2012/benchviz/

Benchviz

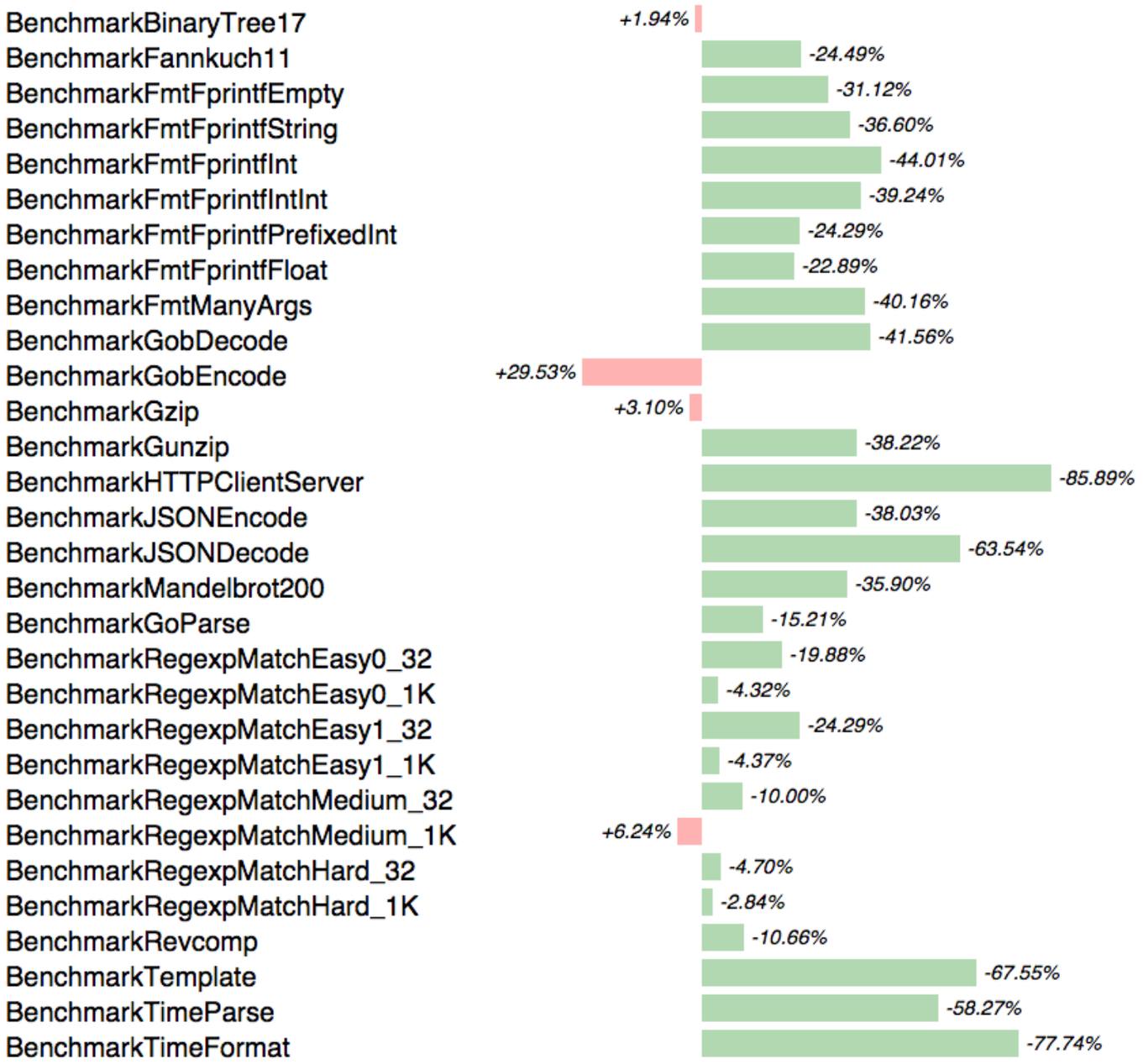
Paste the output of benchcmp below, then hit Benchviz

```
benchmark                          old ns/op     new ns/op     delta
BenchmarkBinaryTree17              2147483647    2147483647    +1.94%
BenchmarkFannkuch11                2147483647    2147483647    -24.49%
BenchmarkFmtFprintfEmpty                  143            98    -31.12%
BenchmarkFmtFprintfString                 459           291    -36.60%
BenchmarkFmtFprintfInt                    359           201    -44.01%
BenchmarkFmtFprintfIntInt                 553           336    -39.24%
BenchmarkFmtFprintfPrefixedInt            494           374    -24.29%
BenchmarkFmtFprintfFloat                  712           549    -22.89%
BenchmarkFmtManyArgs                     2196          1314    -40.16%
BenchmarkGobDecode                   20777120      12142514    -41.56%
BenchmarkGobEncode                   11127380      14413551    +29.53%
BenchmarkGzip                       472108600     486750544     +3.10%
BenchmarkGunzip                     217252300     134219751    -38.22%
BenchmarkHTTPClientServer              373216         52646    -85.89%
BenchmarkJSONEncode                 73926150      45809669    -38.03%
BenchmarkJSONDecode                284564400     103747698    -63.54%
BenchmarkMandelbrot200               7495750       4805043    -35.90%
BenchmarkGoParse                     7575885       6423567    -15.21%
BenchmarkRegexpMatchEasy0_32             161           129    -19.88%
BenchmarkRegexpMatchEasy0_1K             440           421     -4.32%
BenchmarkRegexpMatchEasy1_32             140           106    -24.29%
BenchmarkRegexpMatchEasy1_1K            1006           962     -4.37%
BenchmarkRegexpMatchMedium_32            230           207    -10.00%
BenchmarkRegexpMatchMedium_1K          73466         78051     +6.24%
BenchmarkRegexpMatchHard_32             3743          3567     -4.70%
BenchmarkRegexpMatchHard_1K           120456        117033     -2.84%
BenchmarkRevcomp                  1020450000     911687353    -10.66%
BenchmarkTemplate                  414601800     134526858    -67.55%
BenchmarkTimeParse                      1136           474    -58.27%
BenchmarkTimeFormat                     2390           532    -77.74%
```

ajstarks.org:2012/benchviz

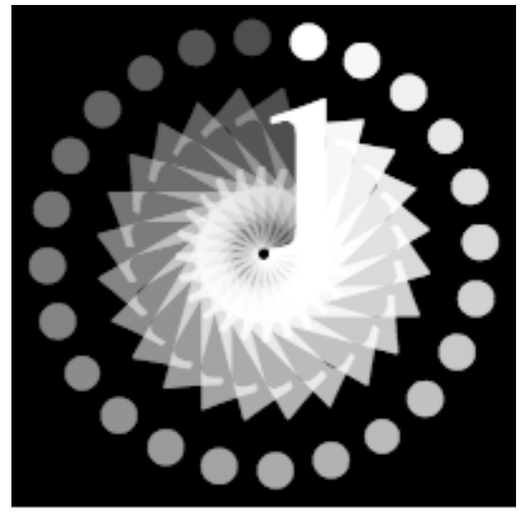
ajstarks.org:2012/benchviz/viz/

Benchmarks

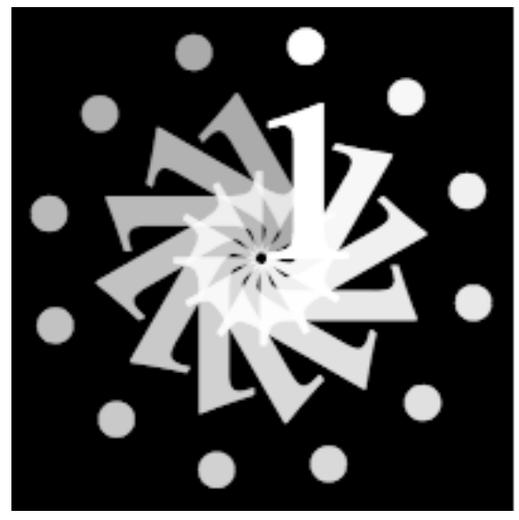


```
package main

import (
    "os"
    "github.com/ajstarks/svgo"
)

func main() {

    width := 200
    height := 200
    a := 1.0
    ai := 0.03
    ti := 15.0

    canvas := svg.New(os.Stdout)
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Gstyle("font-family:serif;font-size:100pt")

    for t := 0.0; t <= 360.0; t += ti {
        canvas.TranslateRotate(width/2, height/2, t)
        canvas.Text(0, 0, "i", canvas.RGBA(255, 255, 255, a))
        canvas.Gend()
        a -= ai
    }
    canvas.Gend()
    canvas.End()
}
```

ti = 15



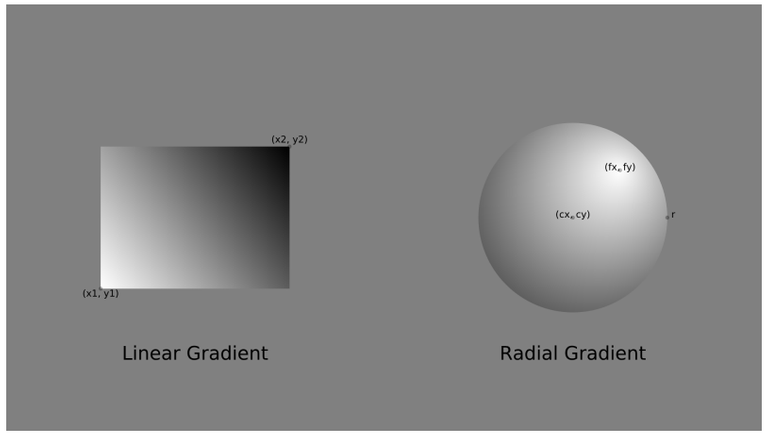
ti = 30

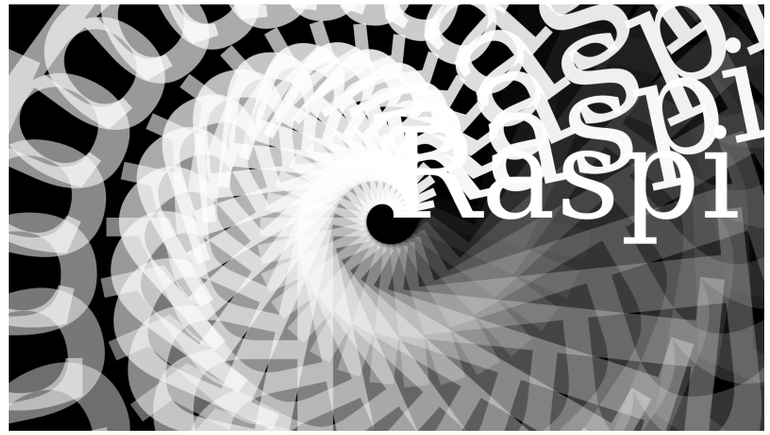


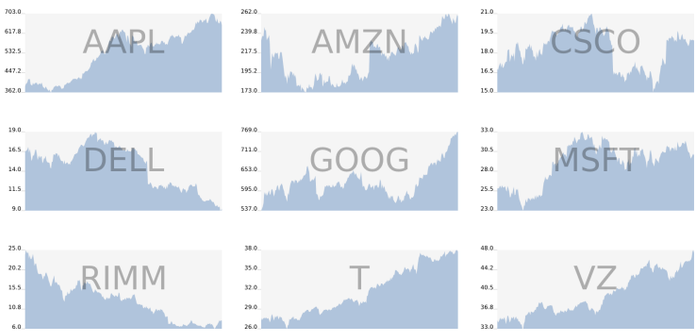
ti = 45

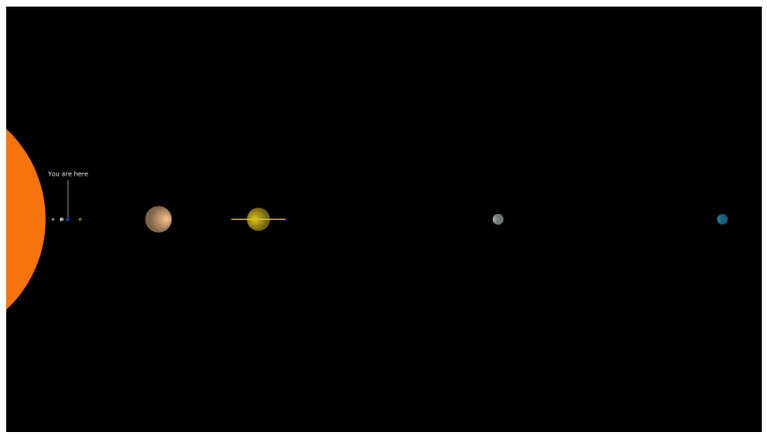
- Circle
- Ellipse
- Rectangle
- Rounded Rectangle
- Line
- Polyline
- Polygon
- Arc
- Quadratic Bezier
- Cubic Bezier
- Image

OpenVG on the Raspberry Pi









Why I use Go

Anthony Starks

@ajstarks
ajstarks@gmail.com


```
package main
import "fmt"

func main() {
    fmt.Println("hello, world")
}
```

This is the traditional hello, world program.
It's six lines long, and imports the fmt package,
leading some to be suprised at the size of the resulting binary.

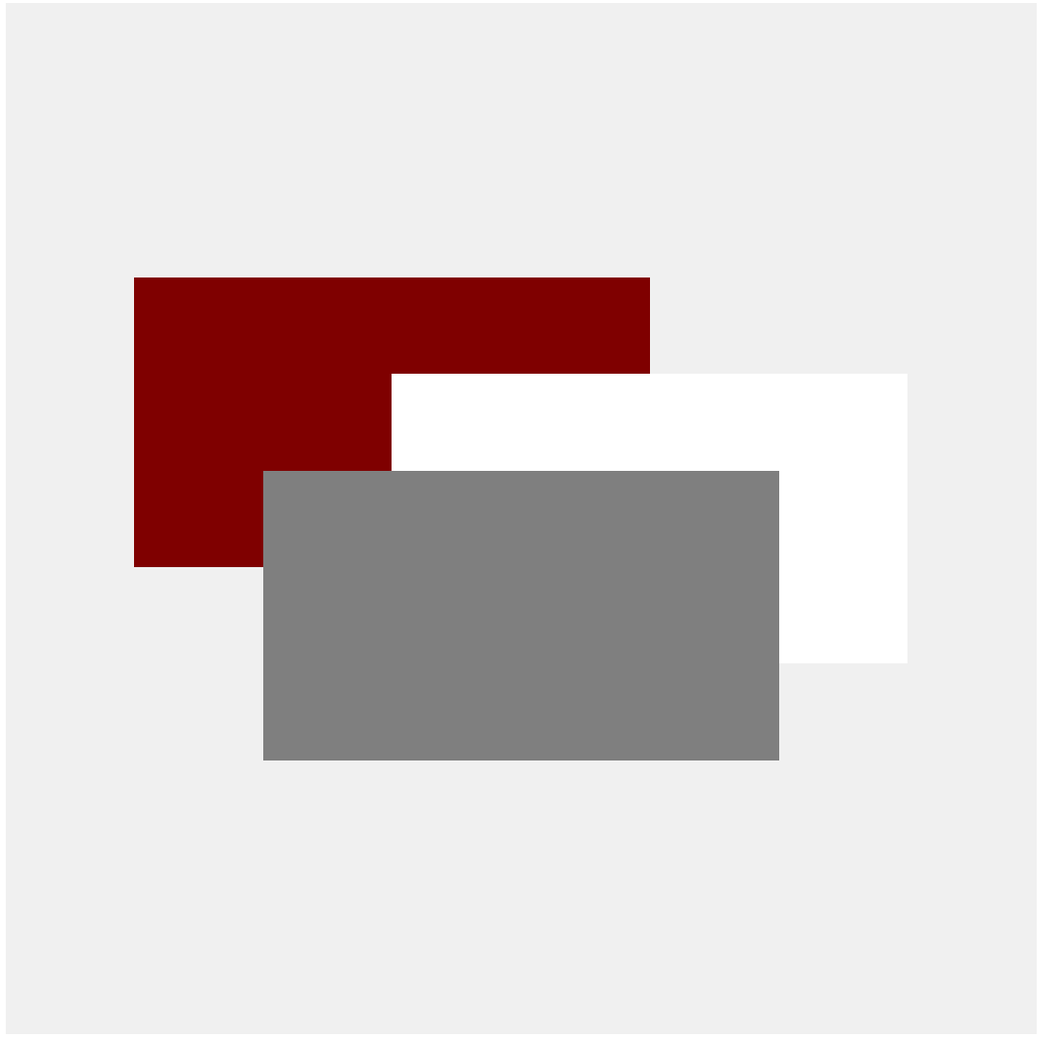
openvg

```
package main

import (
    "bufio"
    "github.com/ajstarks/openvg"
    "os"
)

func main() {
    width, height := openvg.Init()
    w2 := openvg.VGfloat(width / 2)
    h2 := openvg.VGfloat(height / 2)
    w := openvg.VGfloat(width)

    openvg.Start(width, height)                  // Start the picture
    openvg.BackgroundColor("black")              // Black background
    openvg.FillRGB(44, 100, 232, 1)              // Big blue marble
    openvg.Circle(w2, 0, w)                      // The "world"
    openvg.FillColor("white")                    // White text
    openvg.TextMid(w2, h2, "hello, world", "serif", width/10) // Greetings
    openvg.End()                                 // End the picture
    bufio.NewReader(os.Stdin).ReadBytes('\n') // Pause until [RETURN]
    openvg.Finish()                              // Graphics cleanup
}
```

Openvg Functions

```
Circle      (x, y, r VGfloat)
Ellipse     (x, y, w, h VGfloat)
Rect        (x, y, w, h VGfloat)
Roundrect   (x, y, w, h, rw, rh VGfloat)
Line        (x1, y1, x2, y2 VGfloat)
Polyline    (x, y []VGfloat)
Polygon     (x, y []VGfloat)
Arc         (x, y, w, h, sa, aext VGfloat)
Qbezier     (sx, sy, cx, cy, ex, ey VGfloat)
Cbezier     (sx, sy, cx, cy, px, py, ex, ey VGfloat)
Image       (x, y VGfloat, w, h int, s string)
Text        (x, y VGfloat, s string, font string, size int)
TextMid     (x, y VGfloat, s string, font string, size int)
TextEnd     (x, y VGfloat, s string, font string, size int)
```

Deck



a Go package for presentations

Start the deck

Set the canvas size

Begin a slide

Place an image

Draw some text

Make a bullet list

End the list

Draw a line

Draw a rectangle

Draw an ellipse

Draw an arc

Draw a quadratic bezier

Draw a polygon

End the slide

End of the deck

```
<deck>
  <canvas width="1024" height="768" />
  <slide bg="white" fg="black">
    <image xp="70" yp="60" width="256" height="179" name="work.png" caption="Desk"/>
    <text  xp="20" yp="80" sp="3" link="http://goo.gl/Wm05Ex">Deck elements</text>
    <list  xp="20" yp="70" sp="2" type="bullet">
      <li>text, list, image</li>
      <li>line, rect, ellipse</li>
      <li>arc, curve, polygon</li>
    </list>
    <line    xp1="20" yp1="10" xp2="30" yp2="10"/>
    <rect    xp="35"  yp="10" wp="4" hr="75" color="rgb(127,0,0)"/>
    <ellipse xp="45"  yp="10" wp="4" hr="75" color="rgb(0,127,0)"/>
    <arc     xp="55"  yp="10" wp="4" hp="3" a1="0" a2="180" color="rgb(0,0,127)"/>
    <curve   xp1="60" yp1="10" xp2="75" yp2="20" xp3="70" yp3="10" />
    <polygon xc=75 75 80" yc="8 12 10" color="rgb(0,0,127)"/>
  </slide>
</deck>
```

Anatomy of a Deck

Deck elements

- text, list, image
- line, rect, ellipse
- arc, curve, polygon

Desk

10 20 30 40 50 60 70 80 90

90

80

70

60

50

40

30

20

10

Percent Grid

10%, 50%

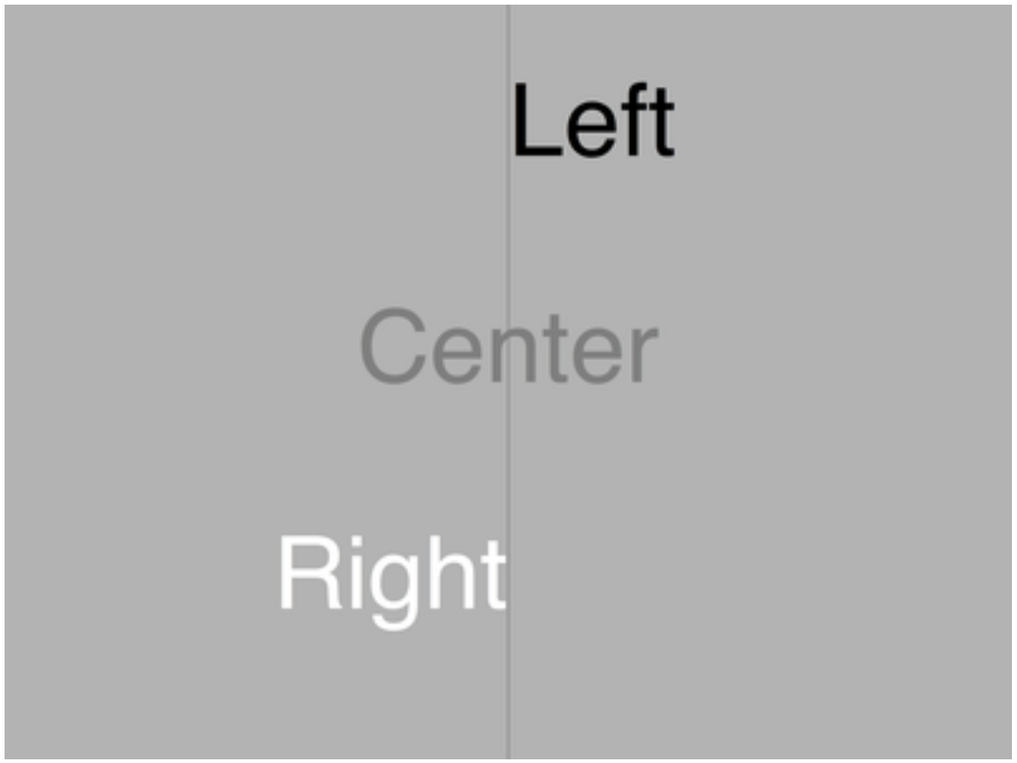
Hello

50%, 50%

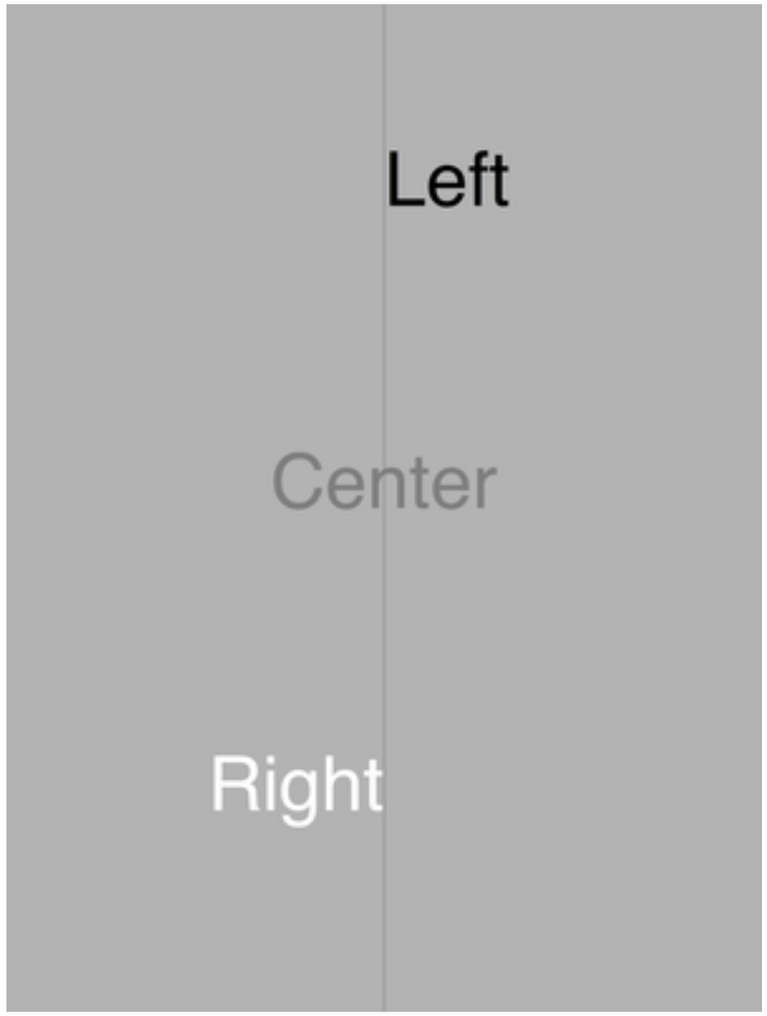
90%, 50%

Percentage-based layout

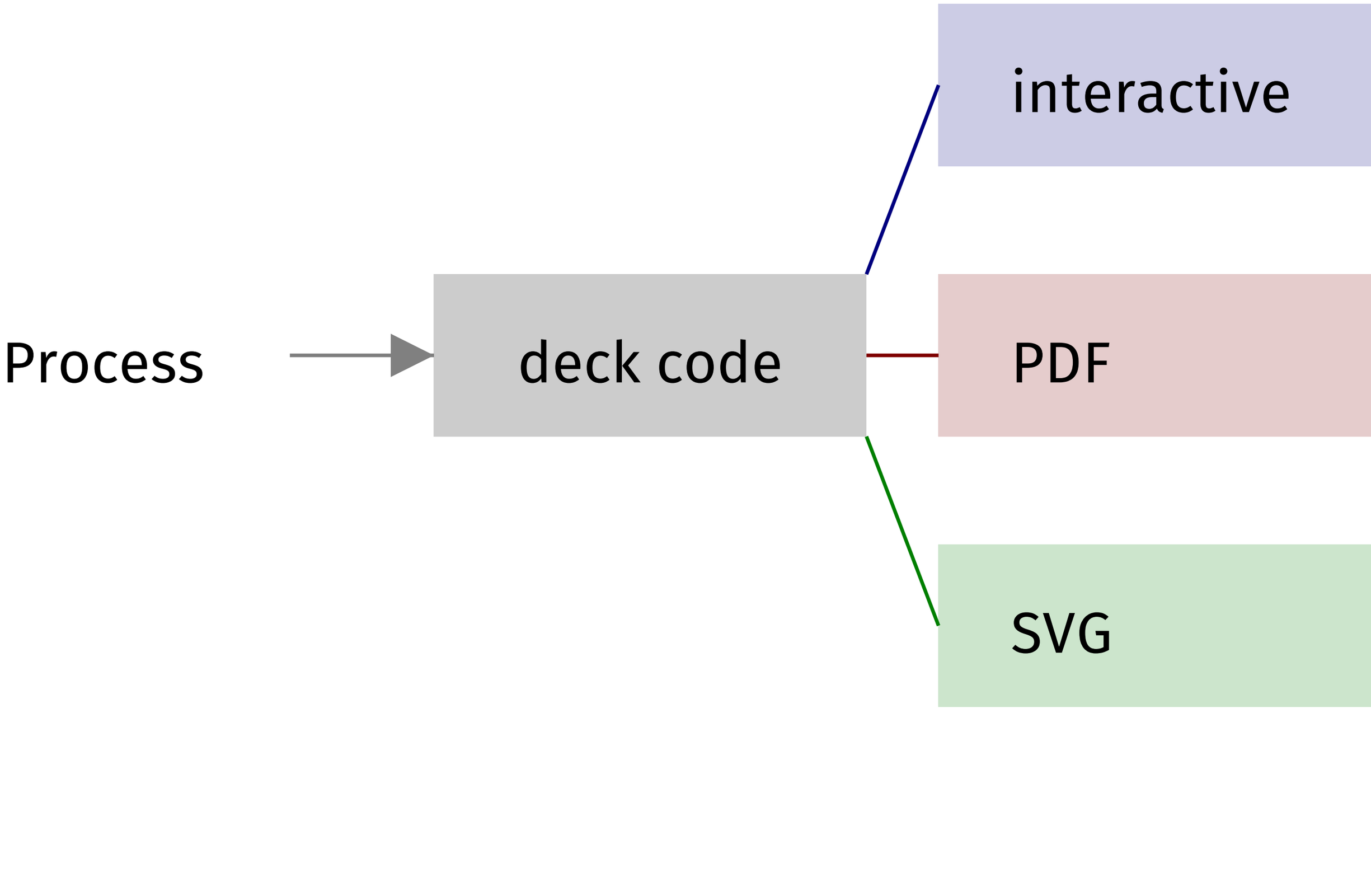
Scaling the canvas



Landscape



Portrait


interactive
Process
deck code
PDF
SVG

NewSlides (where io.Writer, w, h int)

StartDeck()/EndDeck()

StartSlide (colors ...string) EndSlide()

Arc (x, y, w, h, size, a1, a2 float64, color string, opacity ...float64)

Circle (x, y, w float64, color string, opacity ...float64)

Code (x, y float64, s string, size, margin float64, color string, opacity ...float64)

Curve (x1, y1, x2, y2, x3, y3, size float64, color string, opacity ...float64)

Ellipse (x, y, w, h float64, color string, opacity ...float64)

Image (x, y float64, w, h int, name string)

Line (x1, y1, x2, y2, size float64, color string, opacity ...float64)

List (x, y, size float64, items []string, ltype, font, color string)

Polygon (x, y []float64, color string, opacity ...float64)

Rect (x, y, w, h float64, color string, opacity ...float64)

Square (x, y, w float64, color string, opacity ...float64)

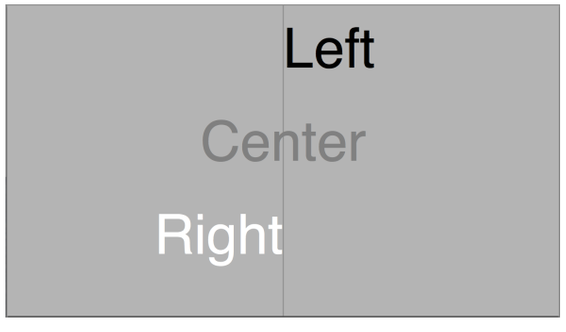
Text (x, y float64, s, font string, size float64, color string, opacity ...float64)

TextBlock (x, y float64, s, font string, size, margin float64, color string, opacity ...float64)

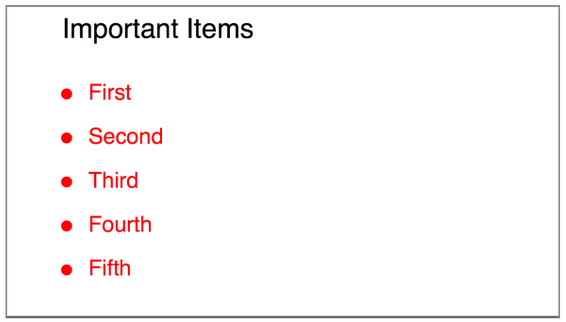
TextEnd (x, y float64, s, font string, size float64, color string, opacity ...float64)

TextMid (x, y float64, s, font string, size float64, color string, opacity ...float64)

```
package main

import (
    "github.com/ajstarks/deck/generate"
    "os"
)

func main() {
    deck := generate.NewSlides(os.Stdout, 1600, 900) // 16x9 deck to standard output
    deck.StartDeck()                                 // start the deck

    // Text alignment
    deck.StartSlide("rgb(180,180,180)")
    deck.Text(50, 80, "Left", "sans", 10, "black")
    deck.TextMid(50, 50, "Center", "sans", 10, "gray")
    deck.TextEnd(50, 20, "Right", "sans", 10, "white")
    deck.Line(50, 100, 50, 0, 0.2, "black", 20)
    deck.EndSlide()
```

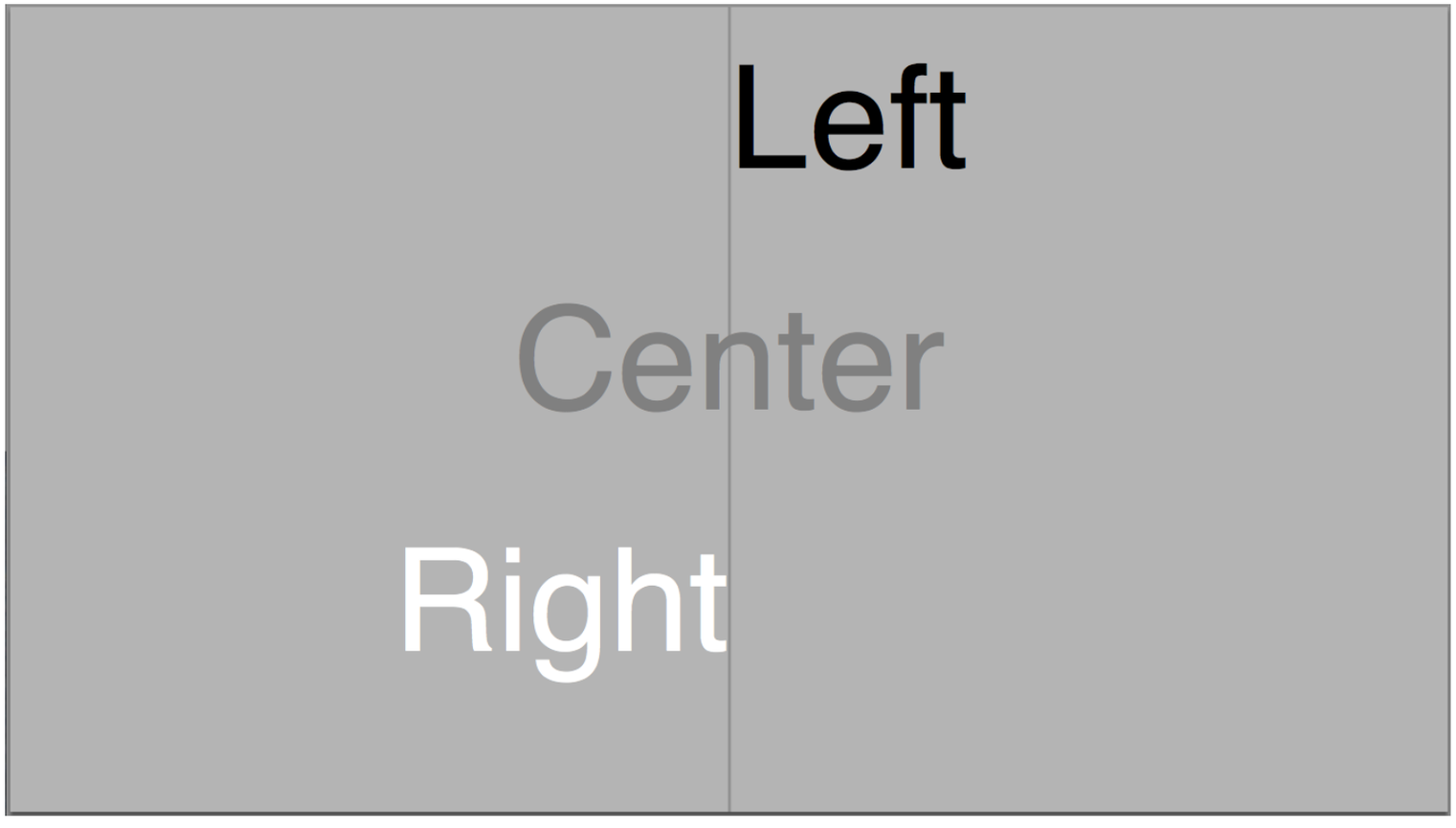



```
    // List
    items := []string{"First", "Second", "Third", "Fourth", "Fifth"}
    deck.StartSlide()
    deck.Text(10, 90, "Important Items", "sans", 5, "")
    deck.List(10, 70, 4, items, "bullet", "sans", "red")
    deck.EndSlide()
```

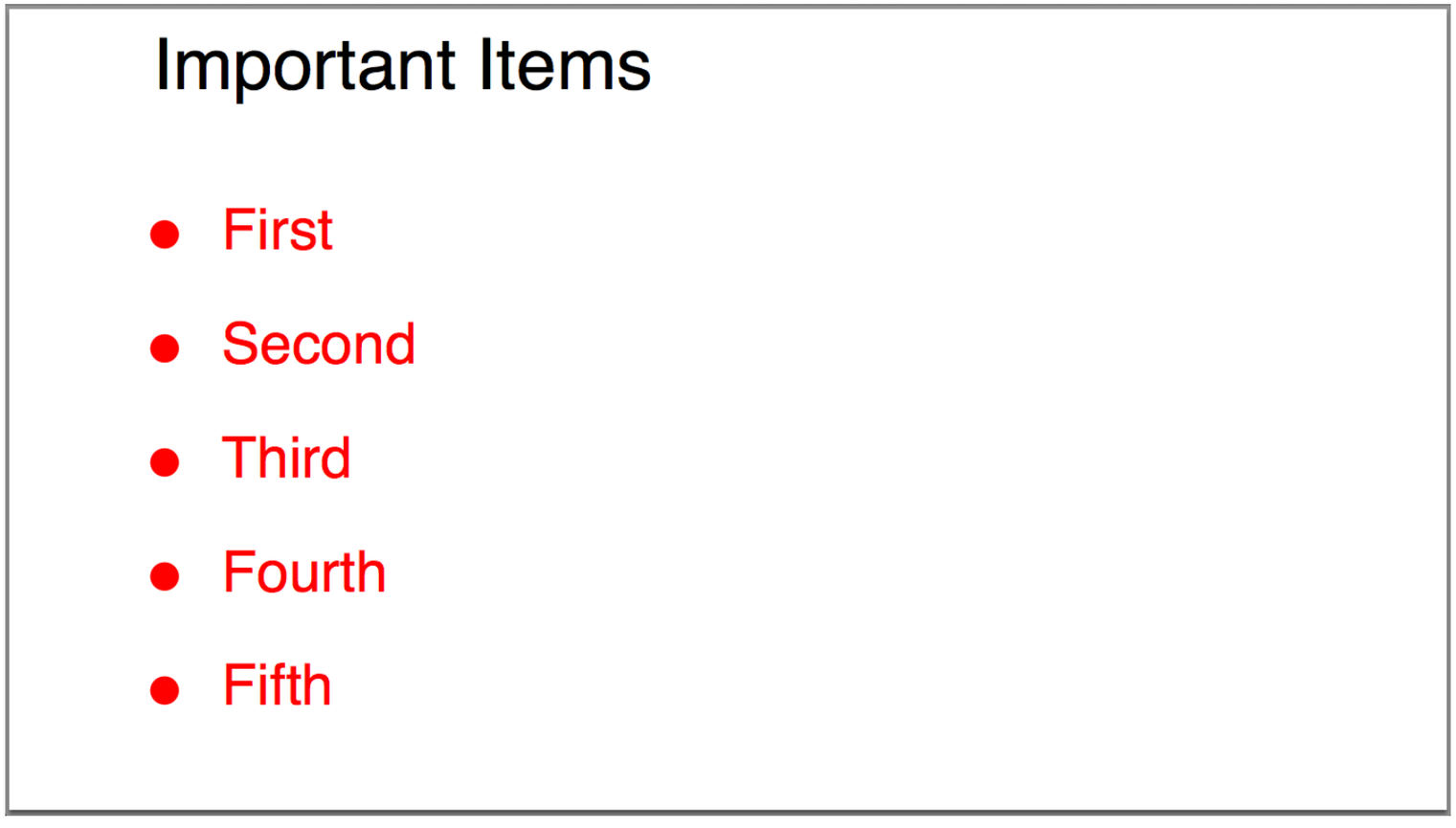



```
    // Picture with text annotation
    quote := "Yours is some tepid, off-brand, generic ‘cola’. What I’m making is “Classic Coke”"
    person := "Heisenberg"
    deck.StartSlide("black", "white")
    deck.Image(50, 50, 1440, 900, "classic-coke.png")
    deck.TextBlock(10, 80, quote, "sans", 2.5, 30, "")
    deck.Text(65, 15, person, "sans", 1.2, "")
    deck.EndSlide()

    deck.EndDeck() // end the deck
}
```

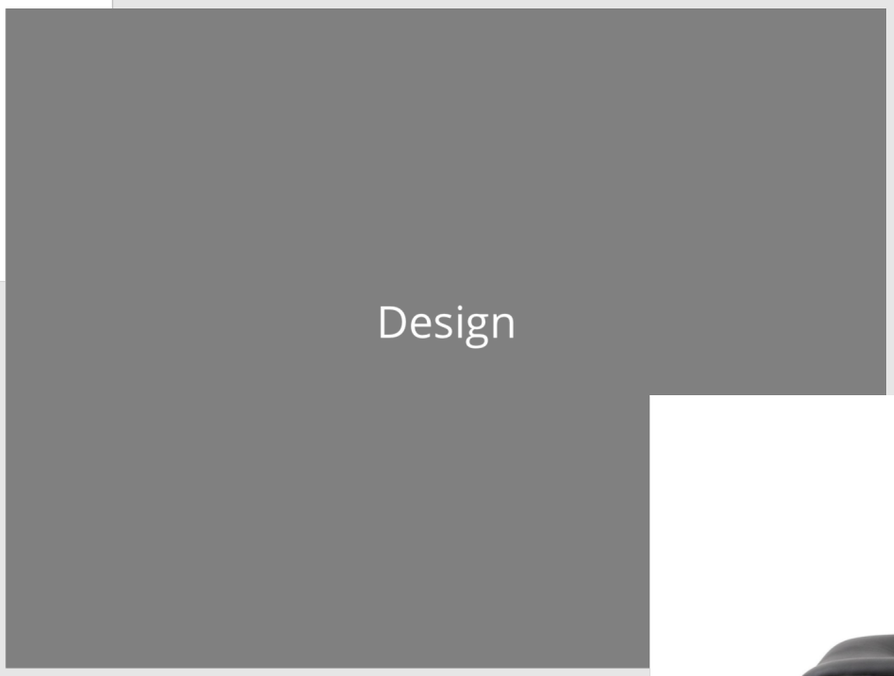
```
// Text alignment
deck.StartSlide("rgb(180,180,180)")
deck.Text(50, 80, "Left", "sans", 10, "black")
deck.TextMid(50, 50, "Center", "sans", 10, "gray")
deck.TextEnd(50, 20, "Right", "sans", 10, "white")
deck.Line(50, 100, 50, 0, 0.2, "black", 20)
deck.EndSlide()
```

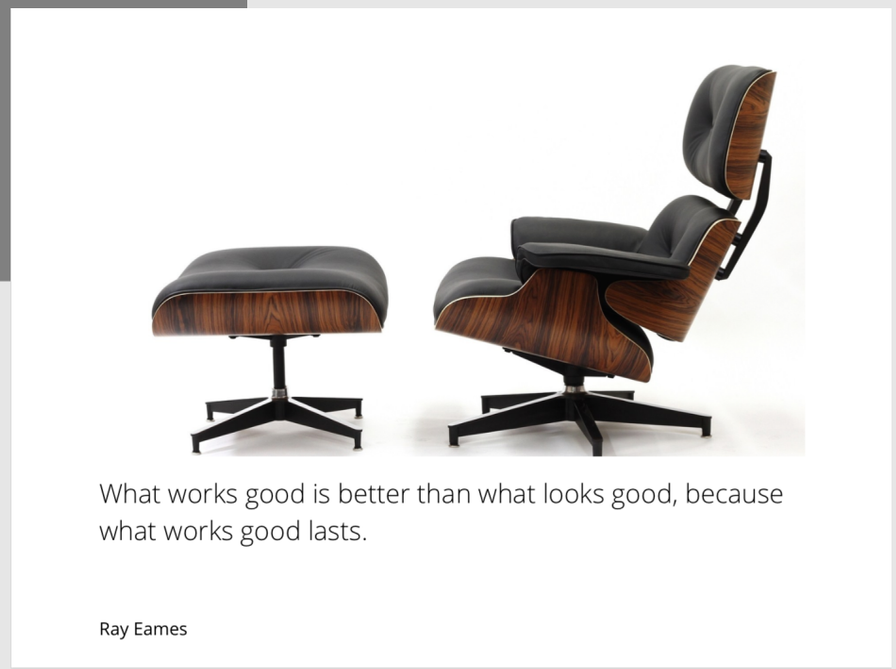


```
// List
items := []string{"First", "Second", "Third", "Fourth", "Fifth"}
deck.StartSlide()
deck.Text(10, 90, "Important Items", "sans", 5, "")
deck.List(10, 70, 4, items, "bullet", "sans", "red")
deck.EndSlide()
```



```
// Picture with text annotation
quote := "Yours is some tepid, off-brand, generic ‘cola’. What I’m making is “Classic Coke”"
person := "Heisenberg"
deck.StartSlide("black", "white")
deck.Image(50, 50, 1440, 900, "classic-coke.png")
deck.TextBlock(10, 80, quote, "sans", 2.5, 30, "")
deck.Text(65, 15, person, "sans", 1.2, "")
deck.EndSlide()
```





capgen slides.txt | pdfdeck ... > slides.pdf

```
# Designing for People
title    A View of User Experience: Designing for People    Anthony Starks / ajstarks@gmail.com / @ajstarks
section    Design    gray    white
caption    eames.png    Ray Eames    What works good is better than what looks good, because what works good lasts.
```

Design Examples

Top

Left

Right

Bottom

10%

30%

70%

10%

Header (top 10%)

Summary (30%)

Detail (70%)

Footer (bottom 10%)

My Story

Section

One

Two

Three

Four

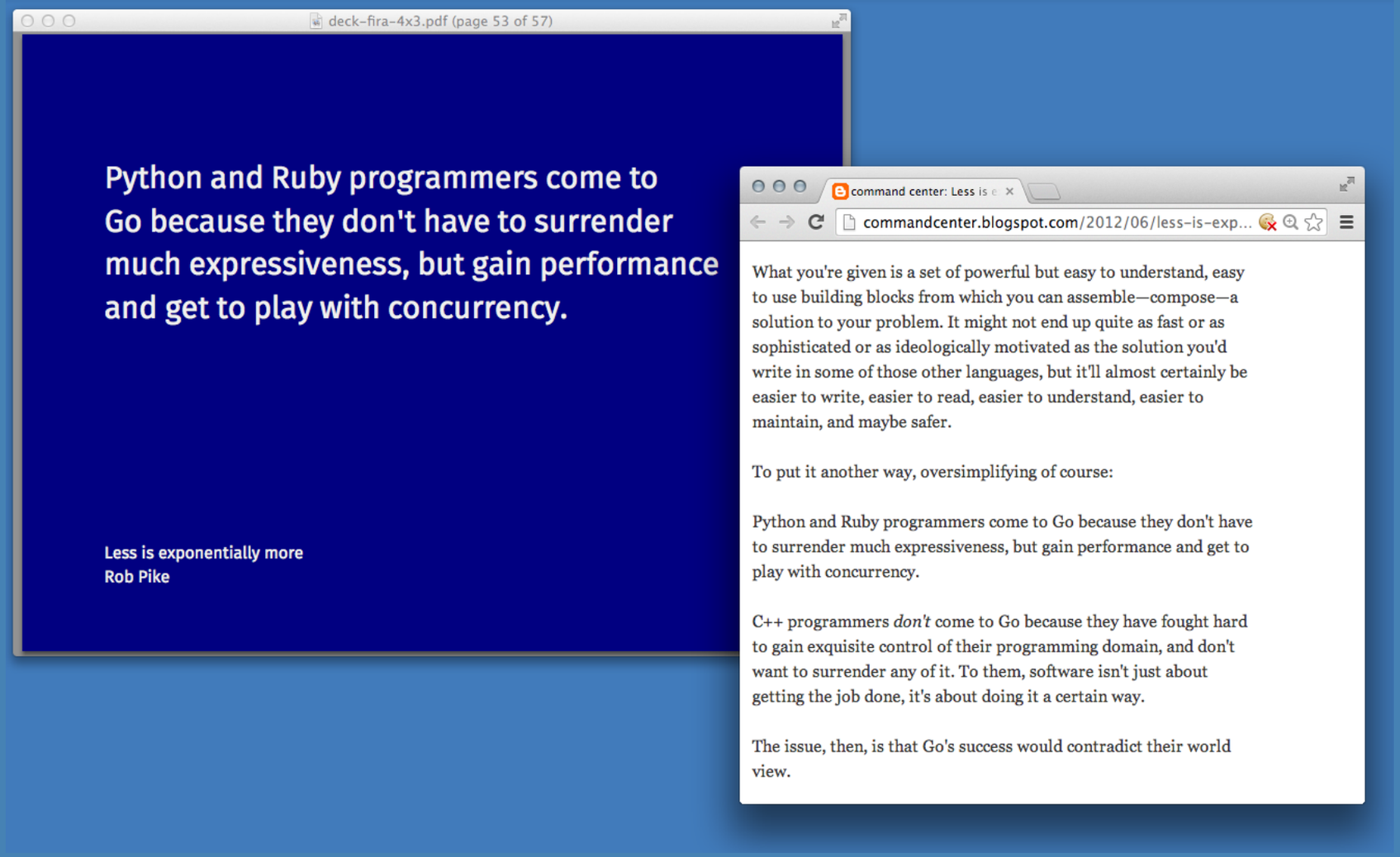
Five

Six

A. Person, January 2015

Document Links

Add web and mailto links with the link attribute of the text element.



Once rendered as a PDF, clicking on the link opens the default browser or email client.

BOS

Virgin America 351

Gate B38

8:35am

SFO

On Time

JFK

US Airways 1207

Gate C31C

5:35pm

IND

Delayed

AAPL	110.22	+0.97 (0.89%)
AMZN	294.74	+3.33 (1.14%)
FB	76.45	-0.27 (0.35%)
GOOG	496.18	+3.63 (0.74%)
MSFT	46.35	-0.24 (0.53%)

Closing Price, 2015-01-13

Two Columns

One

Two

Three

Four

Tree and Sky

Five

Six

Seven

Eight

Rocks

go

build	compile packages and dependencies
clean	remove object files
env	print Go environment information
fix	run go tool fix on packages
fmt	run gofmt on package sources
generate	generate Go files by processing source
get	download and install packages and dependencies
install	compile and install packages and dependencies
list	list packages
run	compile and run Go program
test	test packages
tool	run specified go tool
version	print Go version
vet	run go tool vet on packages

This is not a index card

Rich

Can't buy me love

Bliss

Worse

Better

Misery

We have each other

Poor

A few months ago, I had a look at the brainchild of a few serious heavyweights working at Google. Their project, the Go programming language, is a static typed, c lookalike, semicolon-less, self formatting, package managed, object oriented, easily parallelizable, cluster fuck of genius with an unique class inheritance system. It doesn't have one.

The Go Programming Language

is a static typed,

c lookalike,

semicolon-less,

self formatting,

package managed,

object oriented,

easily parallelizable,

cluster fuck of genius

with an unique class inheritance system.

Andrew Mackenzie-Ross, OBJECTIVE-C LESSONS FROM GO

The Go Programming Language

is a static typed,

c lookalike,

semicolon-less,

self formatting,

package managed,

object oriented,

easily parallelizable,

cluster fuck of genius

with an unique class inheritance system.

Andrew Mackenzie-Ross, OBJECTIVE-C LESSONS FROM GO

The Go Programming Language

is a static typed, c lookalike, semicolon-less, self formatting, package managed, object oriented, easily parallelizable, cluster fuck of genius with an unique class inheritance system

It doesn't have one.

Andrew Mackenzie-Ross, OBJECTIVE-C LESSONS FROM GO

So, the next time you're about to make a subclass, think hard and ask yourself

what would Go do

Andrew Mackenzie-Ross

Python and Ruby programmers come to Go because they don't have to surrender much expressiveness, but gain performance and get to play with concurrency.

Less is exponentially more
Rob Pike

You must not blame me if I do talk to the clouds.

FOR, LO,

the winter is past,
the rain is over and gone;
The flowers appear on the earth;
the time for the singing of birds is come,
and the voice of the turtle is heard in our land.

Song of Solomon 2:11-12

Good Design

is innovative

makes a product useful

is aesthetic

makes a product understandable

is unobtrusive

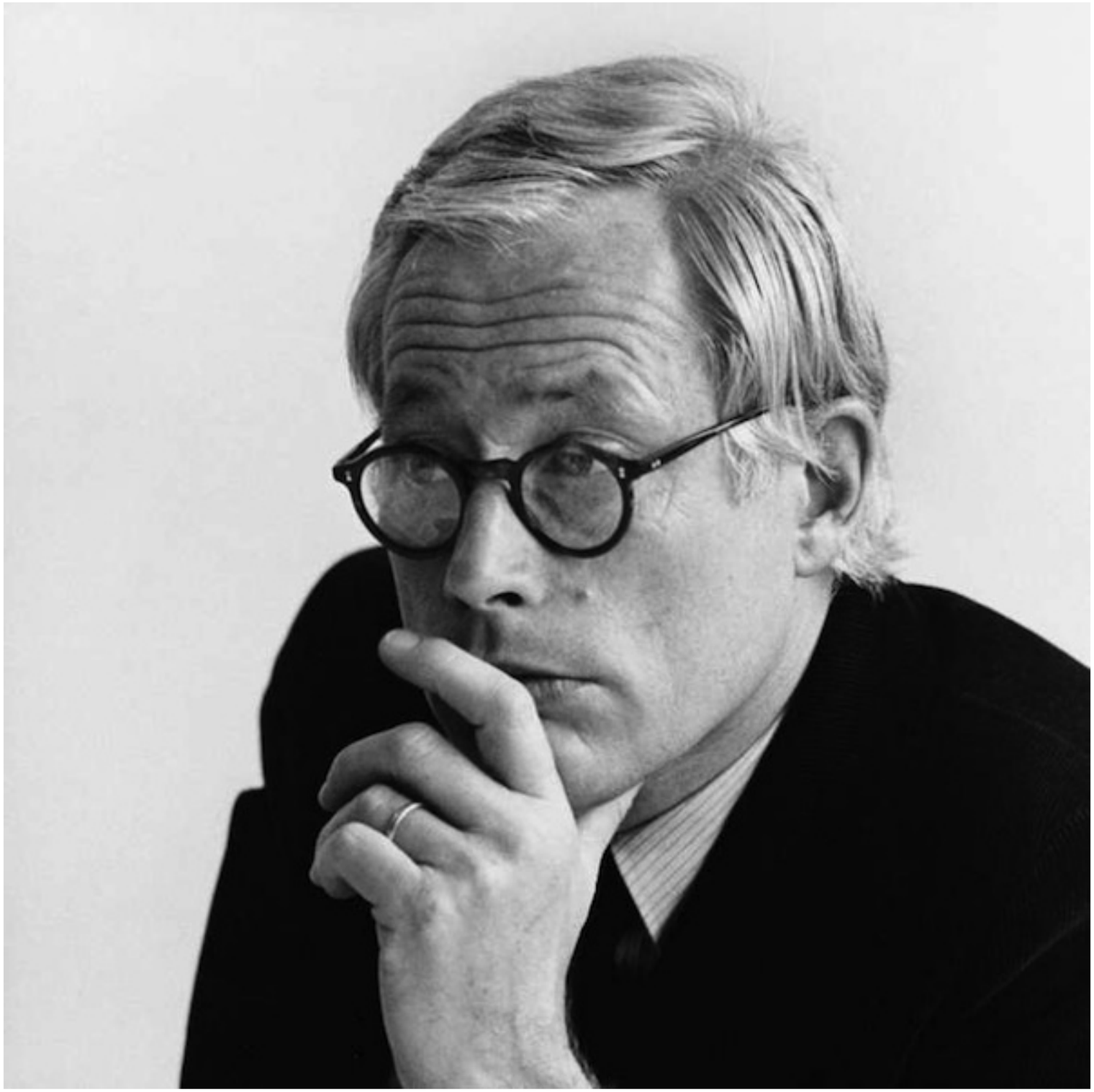
is honest

is long-lasting

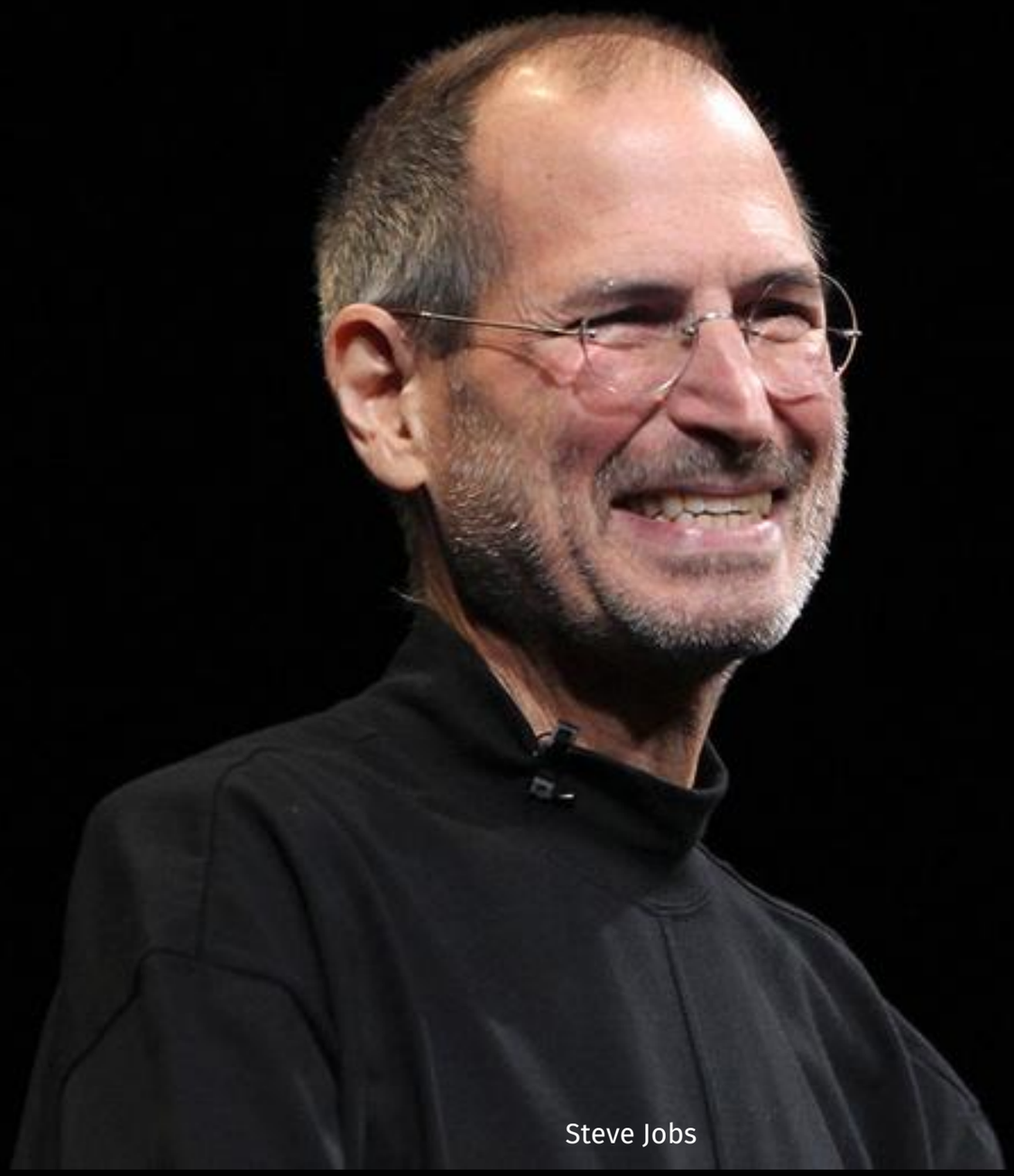
is thorough down to the last detail

is environmentally-friendly

is as little design as possible

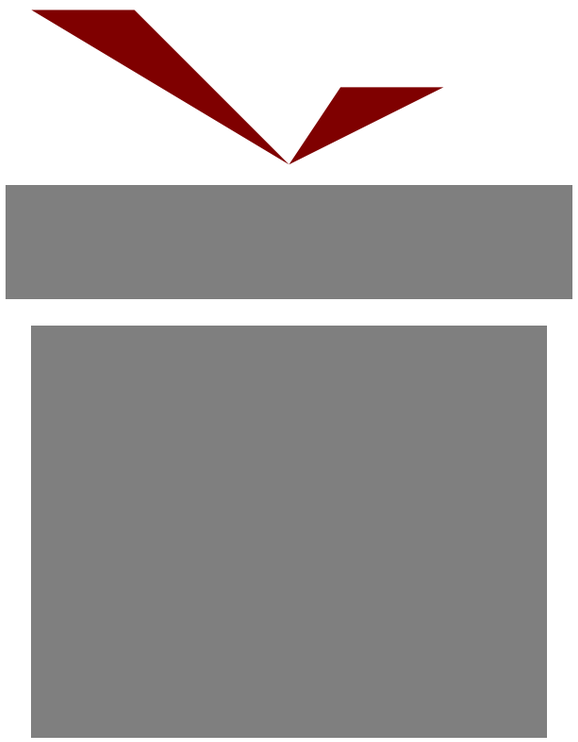


Dieter Rams



You’ve got to start with the customer experience and work back toward the technology, not the other way around.

Steve Jobs



gift

Go Image Filtering Toolkit

gift -contrast 80 -blur 4 rocks.jpg > abstract.jpg

-blur=0: blur value

-brightness=-200: brightness value (-100, 100)

-contrast=-200: contrast value (-100, 100)

-crop="": crop x1,y1,x2,y2

-edge=false: edge

-emboss=false: emboss

-fliph=false: flip horizontal

-flipv=false: flip vertical

-gamma=0: gamma value

-gray=false: grayscale

-hue=-200: hue value (-180, 180)

-invert=false: invert

-max=0: local maximum (kernel size)

-mean=0: local mean filter (kernel size)

-median=0: local median filter (kernel size)

-min=0: local minimum (kernel size)

-resize="": resize w,h

-rotate=0: rotate specified degrees counter-clockwise

-saturation=-200: saturation value (-100, 500)

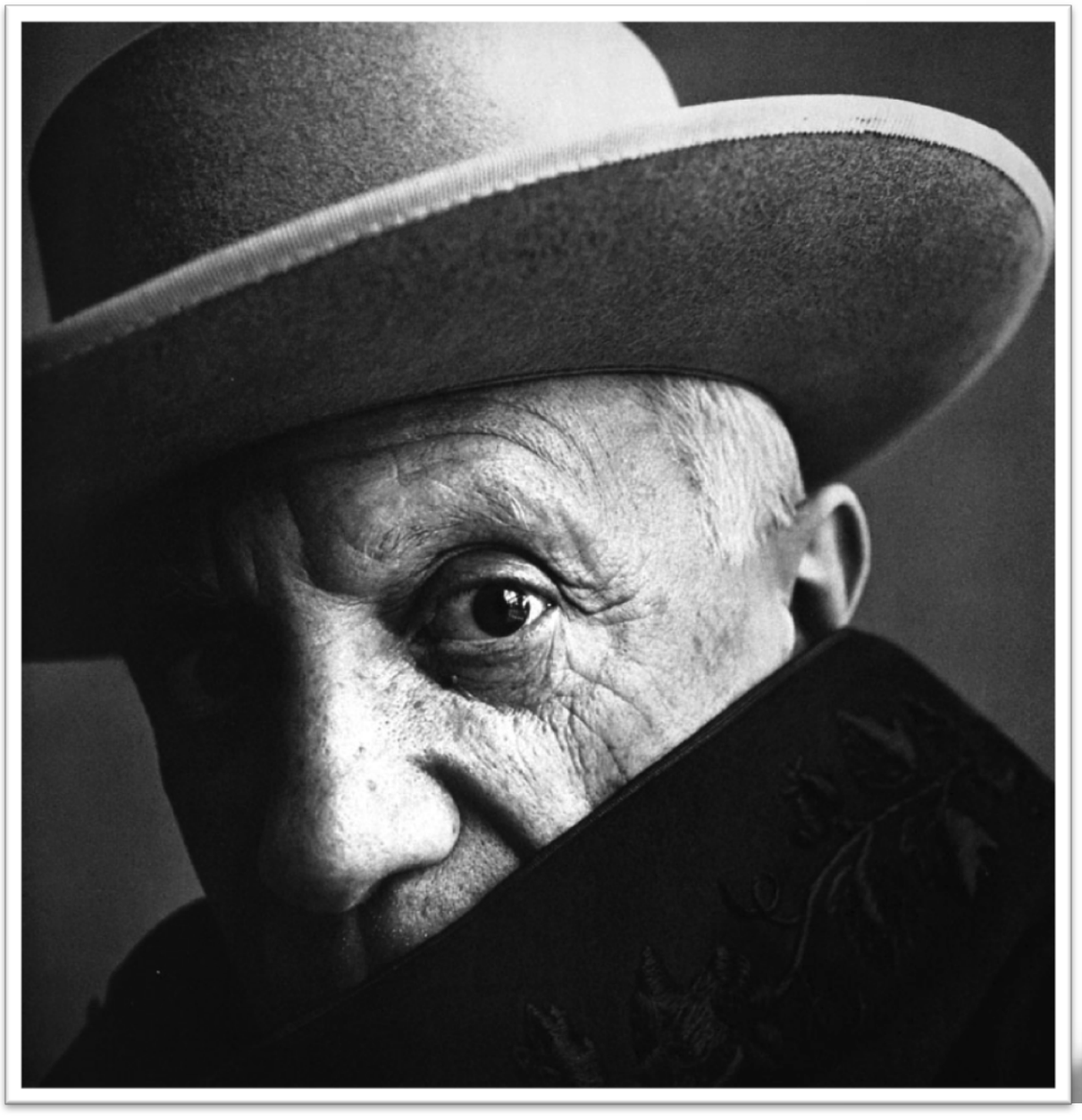
-sepia=-1: sepia percentage (0-100)

-sigmoid="": sigmoid contrast (midpoint,factor)

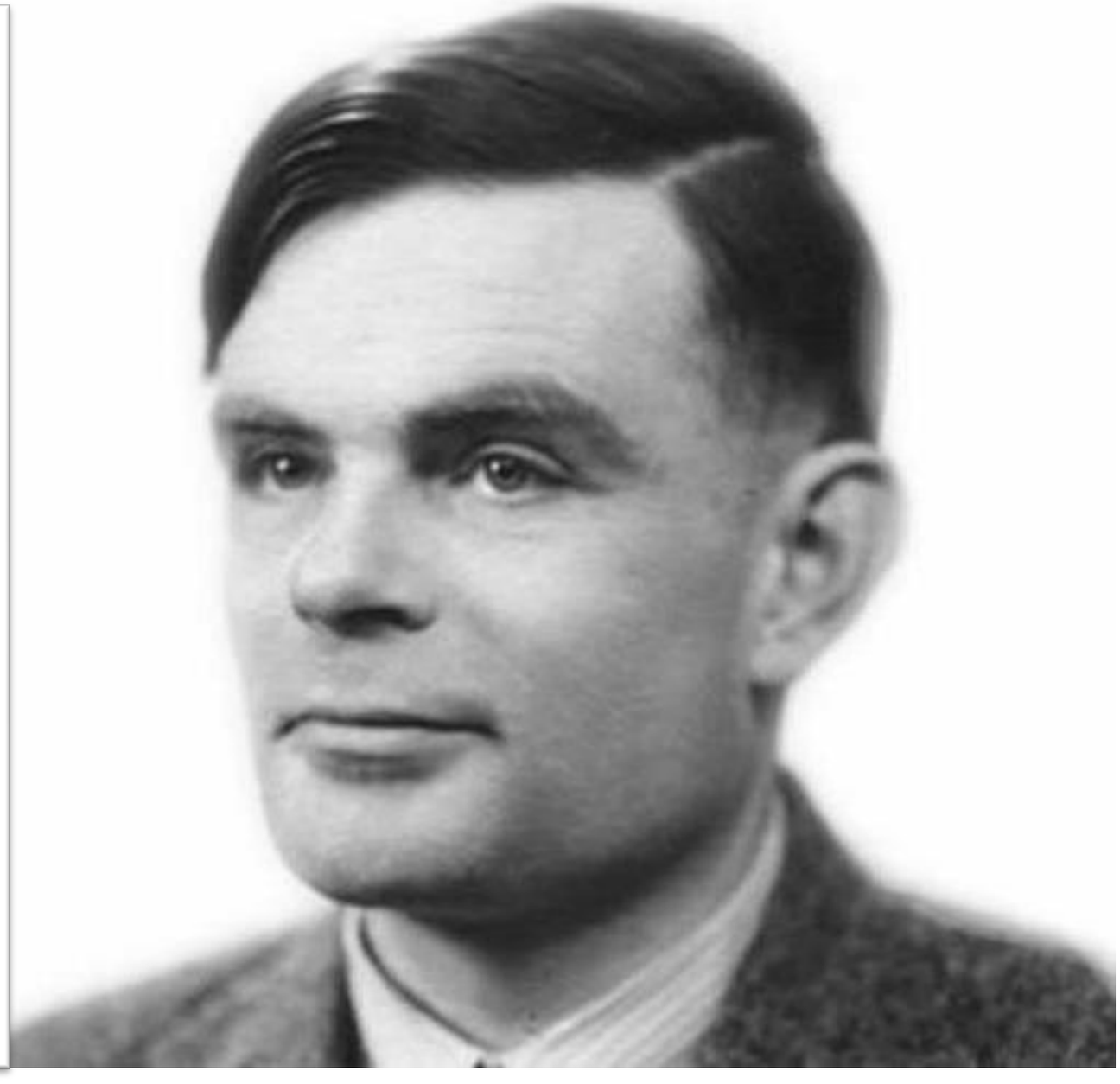
-transpose=false: flip horizontally and rotate 90° counter-clockwise

-transverse=false: flips vertically and rotate 90° counter-clockwise

-unsharp="": unsharp mask (sigma,amount,threshold)



Picasso



Turing

Thank you

github.com/ajstarks

@ajstarks

ajstarks@gmail.com

flickr.com/photos/ajstarks

speakerdeck.com/ajstarks