



For Information Displays

Anthony Starks  
@ajstarks



Go is great in the back end

A wide-angle photograph of a dramatic sky. The upper portion of the image is dominated by a large, bright white cumulus cloud formation. This main cloud is textured and layered, with some darker, more solid areas on its right side. Below it, the sky transitions into a darker shade of blue, with several smaller, fluffy white clouds scattered across the lower half. The overall effect is one of a powerful, natural scene.

But sometimes it's about the picture

# Information Displays

## Packages and Tools

## Lots of pictures

## Why Go?



Edward Tufte

Display data for precise, effective, quick analysis,  
Small multiples, optimizing the data-ink ratio,  
Visual and beautiful evidence.



Nigel Holmes

Designing for the context: The data visualizer  
asks: “What’s the data?”, the infographics  
designer: “What’s the Story?”

An interesting arrangement  
of text and graphics designed  
to convey a message

# I hate pushing pixels

programs are so much  
better at it



Photo by deborahschillbach

Precision

Efficiency

Consistency



**Fatih Arslan**

@fatih

Following



If you don't learn your tool in and out, the tool will block you during the progress. Don't let the tool disrupt you. If there is too much disruption, it's time to switch to a new tool. If there is no such tool, it's time to write your own.

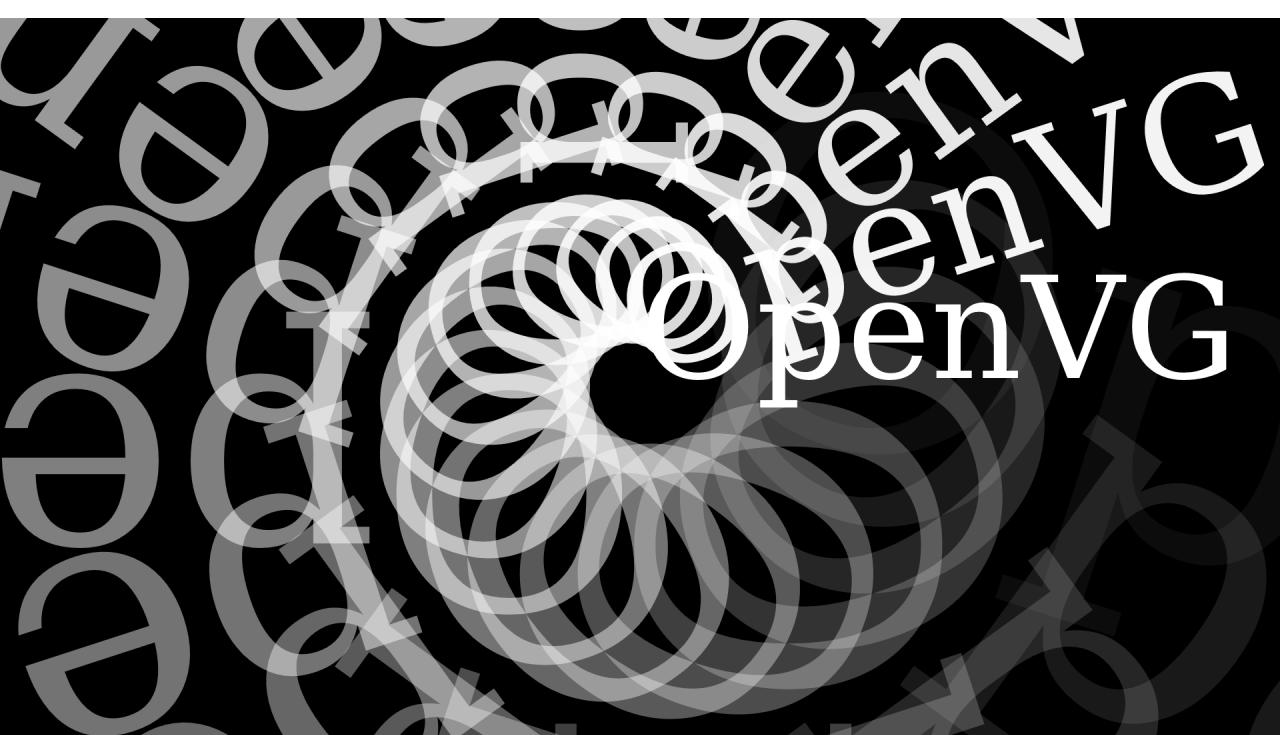
4:43 AM - 12 Apr 2018

# SVGo



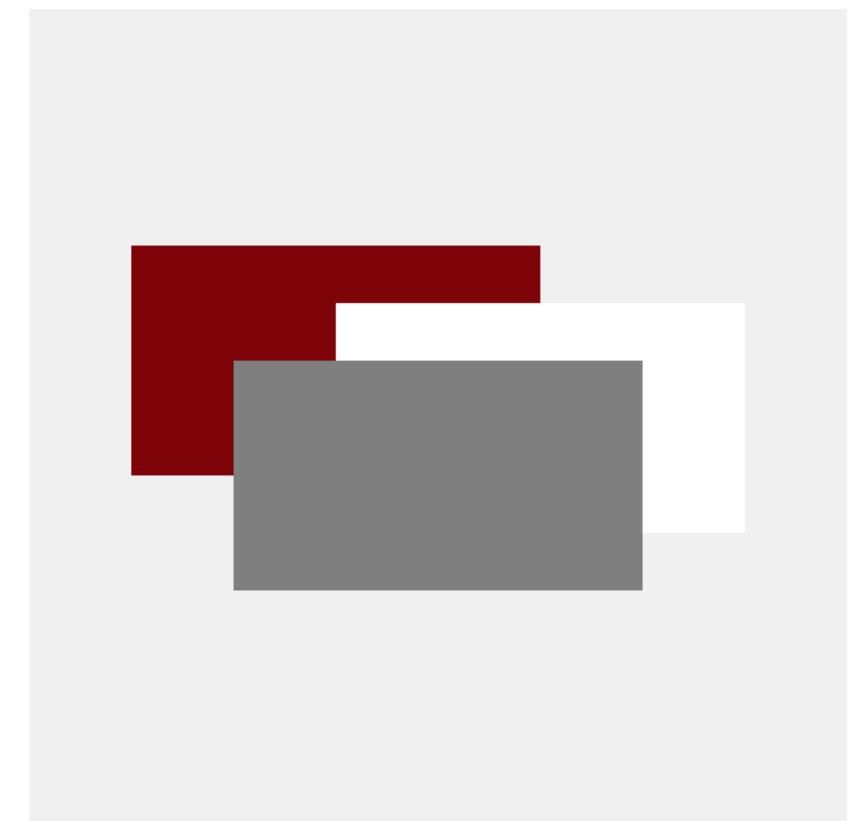
`go get github.com/ajstarks/svg/...`

# OpenVG



`go get github.com/ajstarks/openvg`

# Deck



`go get github.com/ajstarks/deck/`  
`go get github.com/ajstarks/deck/generate`  
`go get github.com/ajstarks/deck/cmd/pdfdeck/`  
`go get github.com/ajstarks/deck/cmd/dchart/`





Scale



Roundrect



rgb(164,198,57)

Line

Circle

Arc

Line

Rect

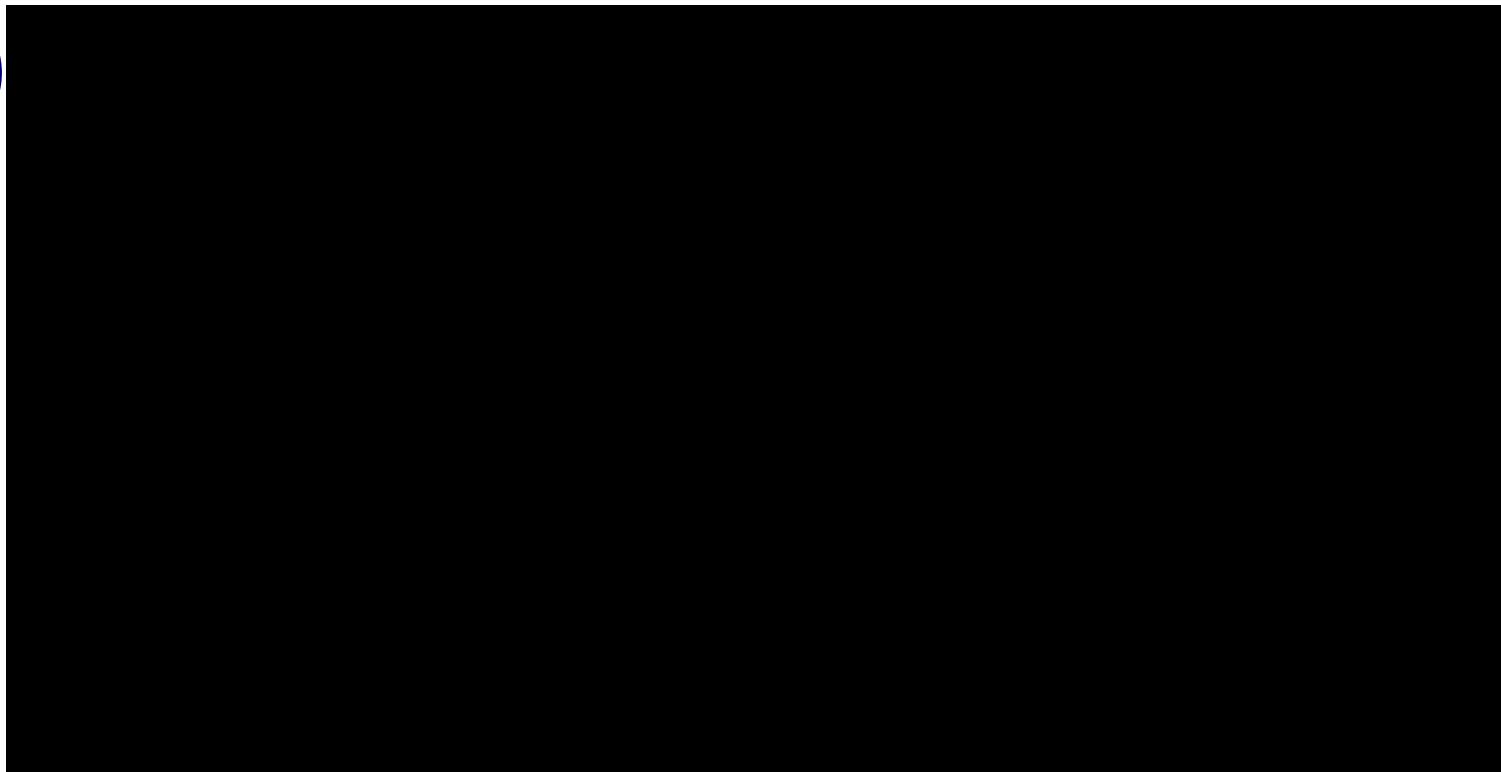
Element

Arguments

Rect (100,200,250,125)

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

(100, 200)

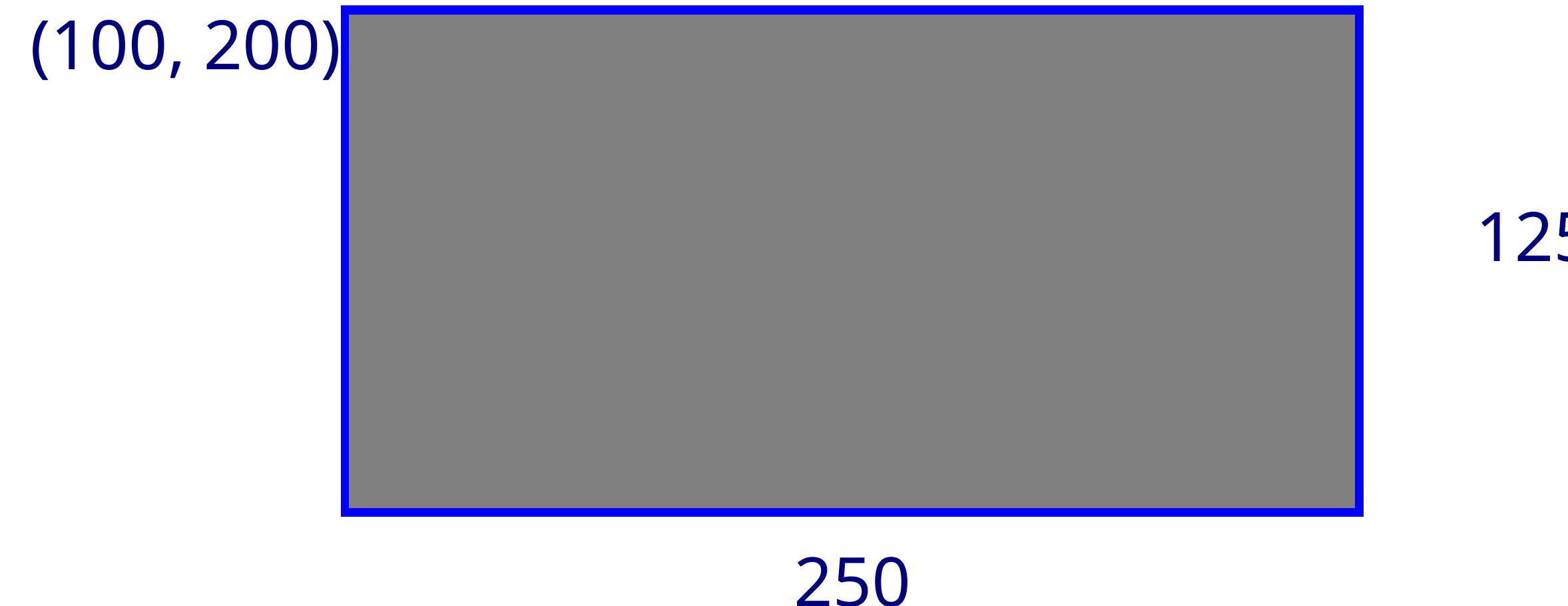


125

250

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

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"/>
```

(100, 200)



250

125

```
package main

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

func main() {
    canvas := svg.New(os.Stdout)
    width := 700
    height := 500
    style := "font-size:72pt;fill:white;text-anchor:middle"

    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Circle(width/2, height, width/2, "fill:rgb(77, 117, 232)")
    canvas.Text(width/2, height*3/4, "hello, world", style)
    canvas.End()
}
```



localhost:1999/hwsvg.go

```
package main

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

func main() {
    canvas := svg.New(os.Stdout)
    width := 700
    height := 500
    style := "font-size:72pt;fill:white;text-anchor:middle"

    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Circle(width/2, height, width/2, "fill:rgb(77, 117, 232)")
    canvas.Text(width/2, height*3/4, "hello, world", style)
    canvas.End()
}
```

(Shift-Enter to compile and run.)  Compile and run after each keystroke

The image shows a web-based development environment. On the left, there is a code editor window with a light gray background. It contains a Go program that generates an SVG image. The code imports the `svgo` library and creates a new canvas with a width of 700 and a height of 500 pixels. It then draws a blue circle at the center (width/2, height) with a radius of width/2 and a fill color of `rgb(77, 117, 232)`. Inside the circle, the text "hello, world" is displayed in white font, centered both horizontally and vertically within the circle. The text has a font size of 72pt and a style of "text-anchor:middle". At the bottom of the code editor, there is a note: "(Shift-Enter to compile and run.)" followed by a checkbox labeled "Compile and run after each keystroke". On the right side of the interface, the generated SVG image is displayed. The image consists of a solid black rectangular background with a single blue circle centered in the middle. The text "hello, world" is rendered in white within the blue circle. The browser's address bar at the top shows the URL "localhost:1999/hwsvg.go".

svgplay

```
localhost:1999/hwsvg.go
```

```
package main

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

func main() {
    canvas := svg.New(os.Stdout)
    width := 700
    height := 500
    style := "font-size:72pt;fill:white;text-anchor:middle"

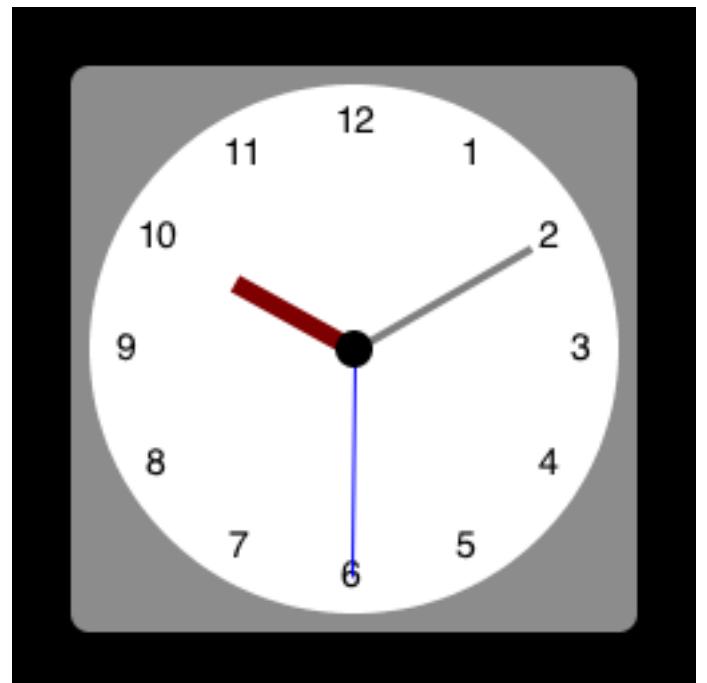
    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Circle(width/2, height, width/2, "fill:rgb(255,0,0)")
    canvas.Text(width/2, height*3/4, "hello, Mars", style)
    canvas.End()
}
```

(Shift-Enter to compile and run.)  Compile and run after each keystroke

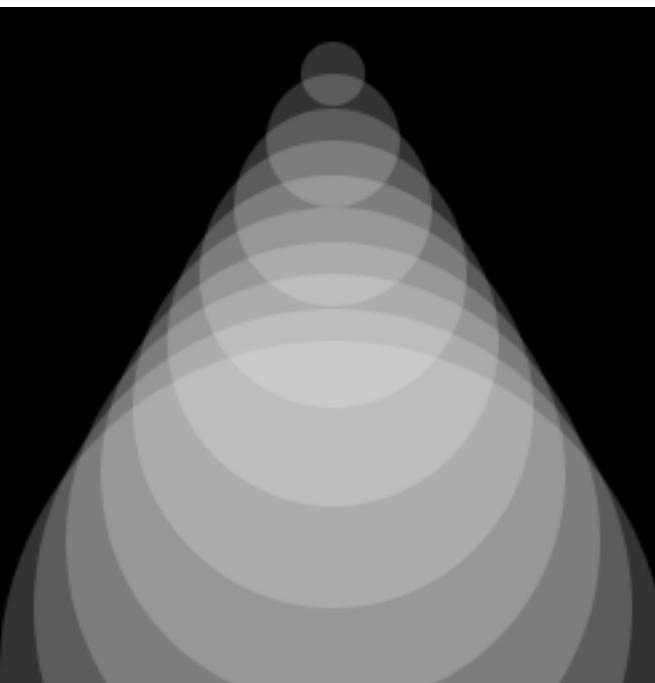
The screenshot shows a web-based development environment. On the left, a code editor displays a Go program named `hwsvg.go`. The code uses the `svggo` library to create a simple SVG visualization. It starts by importing the necessary packages, then defines a `main` function. Inside the `main` function, it creates a new `svg` canvas using `os.Stdout`, sets its width to 700 and height to 500 pixels, and specifies a font size of 72pt with white text and center alignment. The canvas then starts at these dimensions, draws a black rectangle, and places a large red circle at the center. Finally, it adds the text "hello, Mars" in white at the top-right quadrant of the circle. On the right side of the browser window, the resulting SVG visualization is displayed. It features a solid black rectangular background with a large, semi-transparent red circle centered on it. The text "hello, Mars" is rendered in a large, white, serif font, positioned in the upper-right area of the red circle.

svgplay

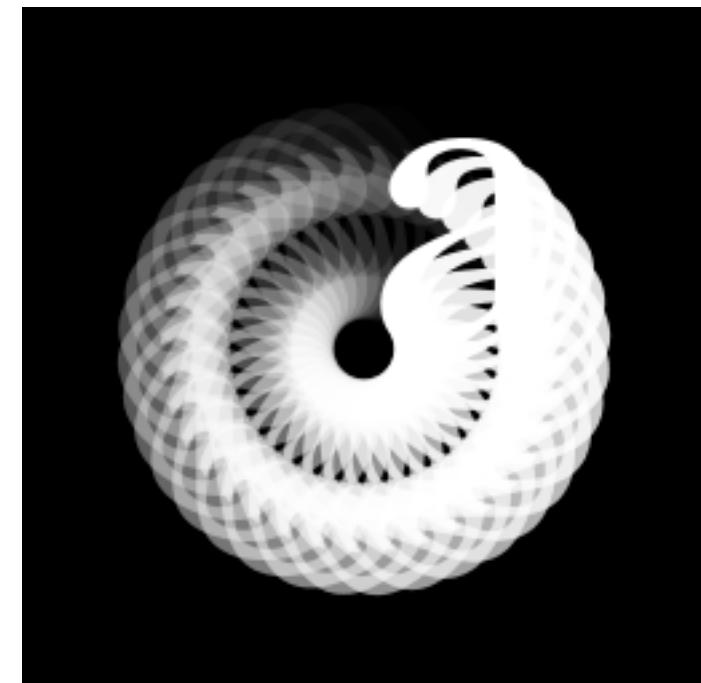
<https://ajstarks.org:1958/>



clock



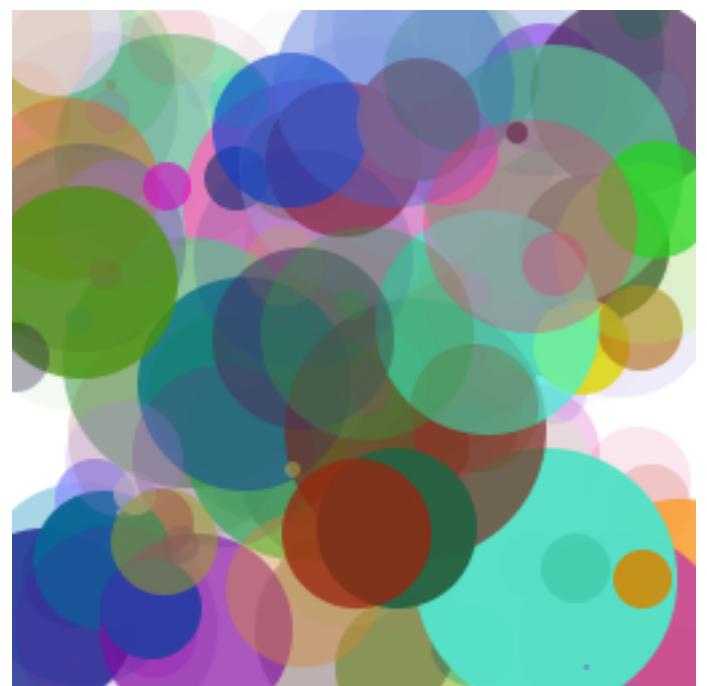
funnel



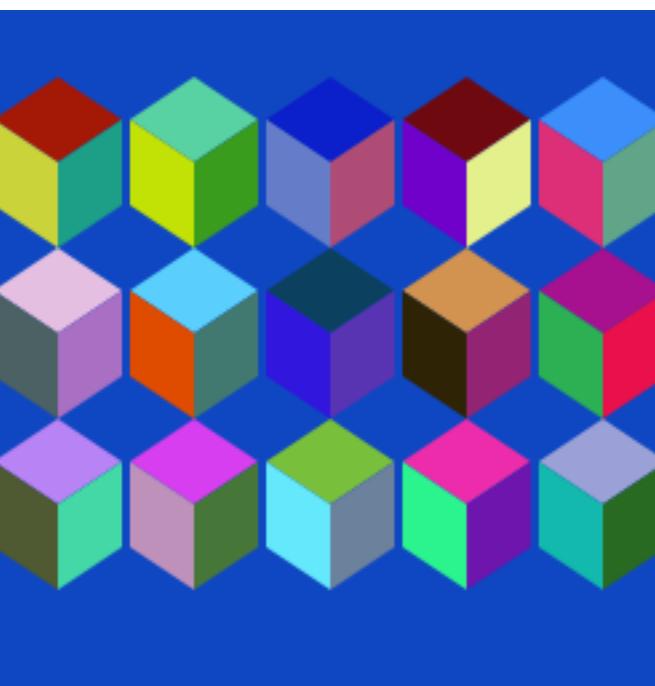
rotext



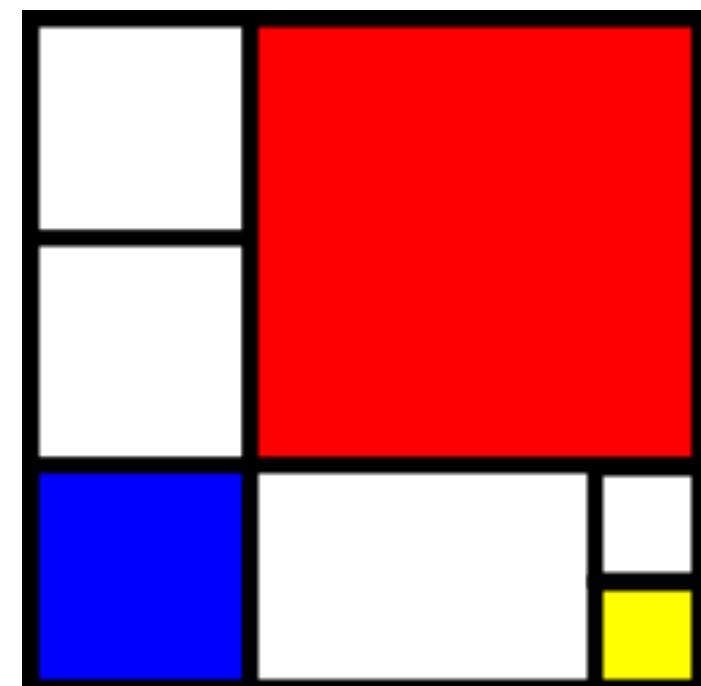
flower



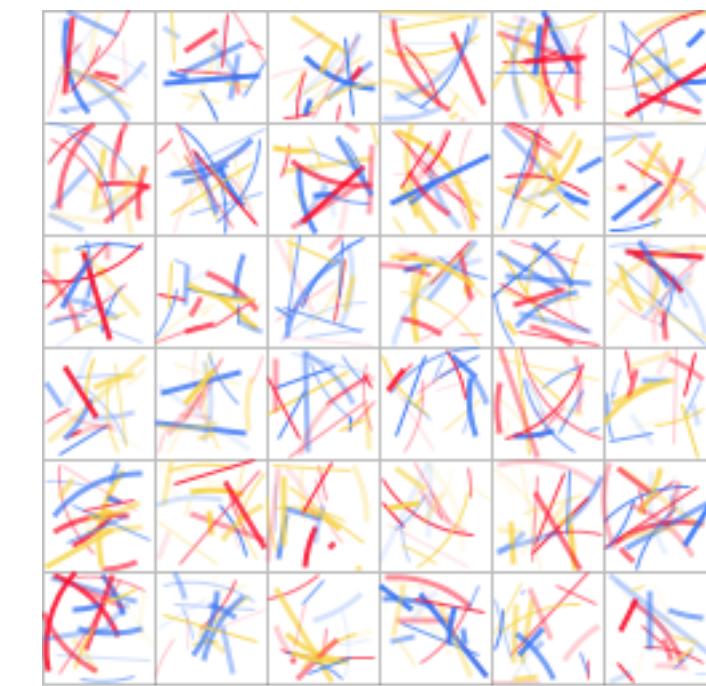
rshape



cube



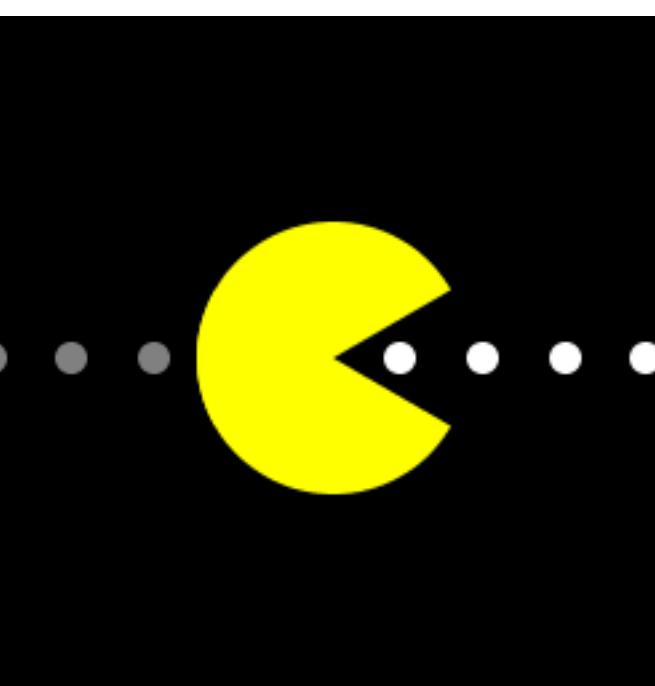
mondrian



lewitt



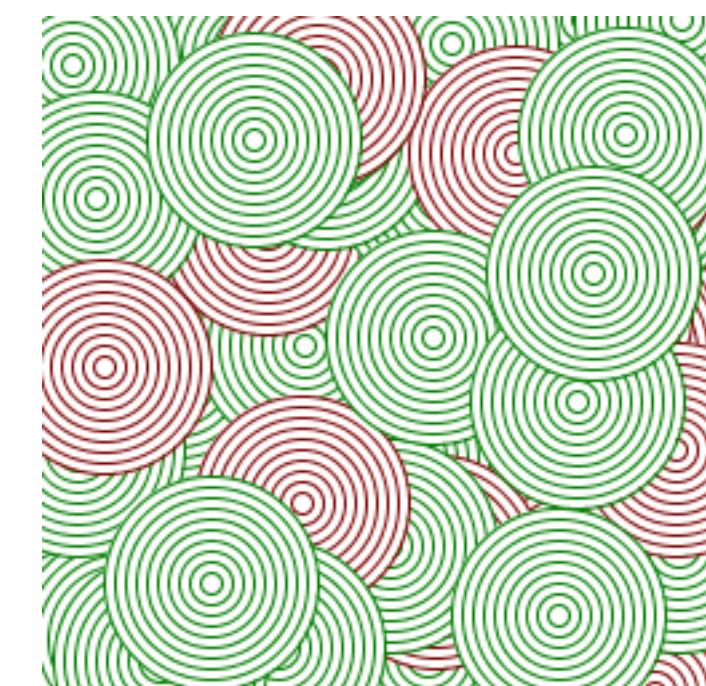
face



pacman



tux

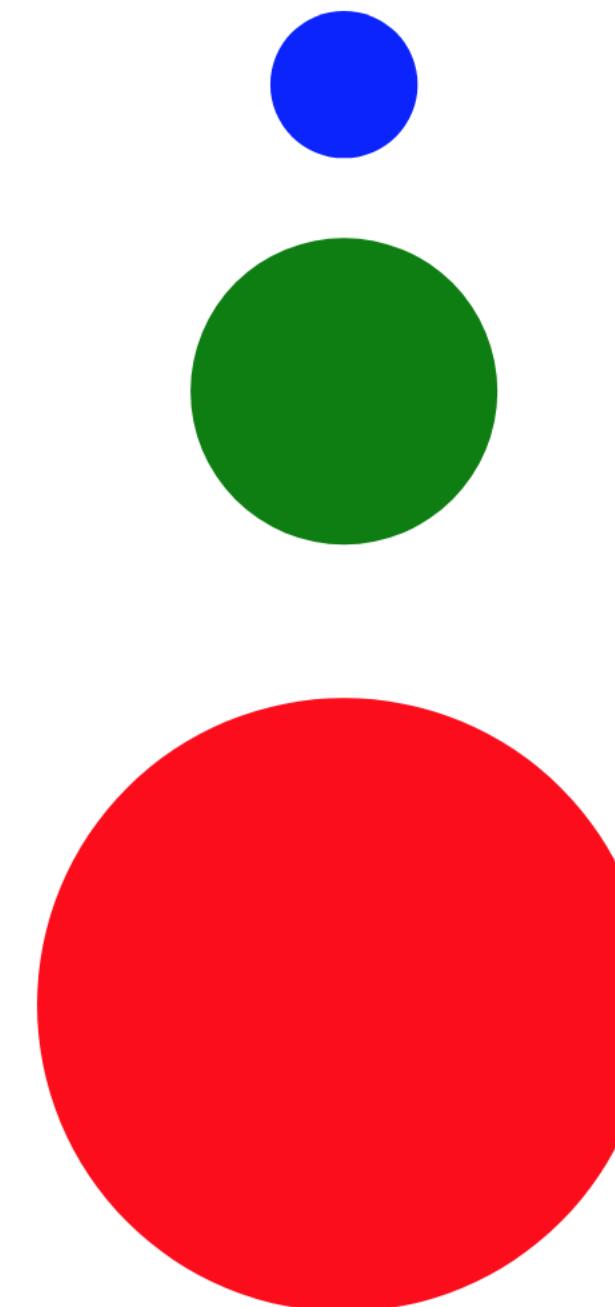


concentric

# Data

```
<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>
```

# Picture



Little:This is small/blue

Med:This is medium/green

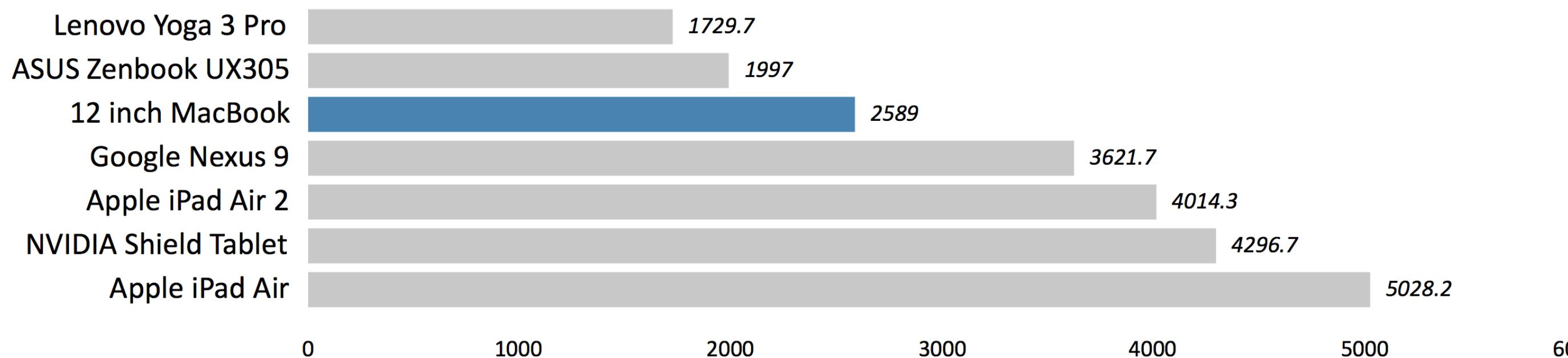
Big:This is large/red

```

<barchart title="2015 MacBook Benchmarks">
  <note>Source: AnandTech, April 14, 2015</note>
  <bdata scale="0,6000,1000"
    title="Mozilla Kraken 1.1 (native browser, milliseconds)">
    <bitem value="1729.7" name="Lenovo Yoga 3 Pro"/>
    <bitem value="1997.0" name="ASUS Zenbook UX305"/>
    <bitem color="steelblue" value="2589.0" name="12 inch MacBook"/>
    <bitem value="3621.7" name="Google Nexus 9"/>
    <bitem value="4014.3" name="Apple iPad Air 2"/>
    <bitem value="4296.7" name="NVIDIA Shield Tablet"/>
    <bitem value="5028.2" name="Apple iPad Air"/>
  </bdata>
</barchart>

```

*Mozilla Kraken 1.1 (native browser, milliseconds)*



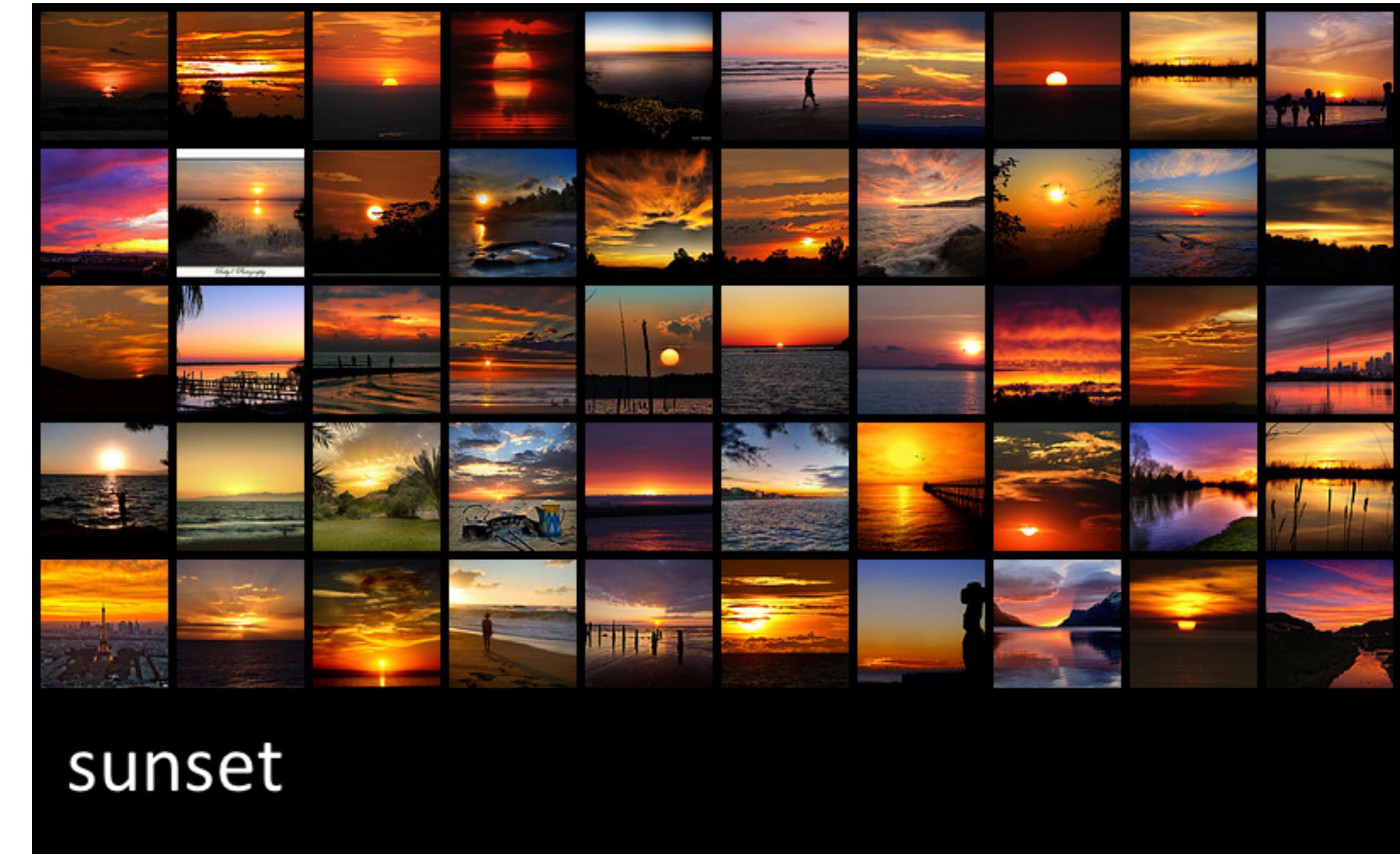
# 2015 MacBook Benchmarks

*Source: AnandTech, April 14, 2015*

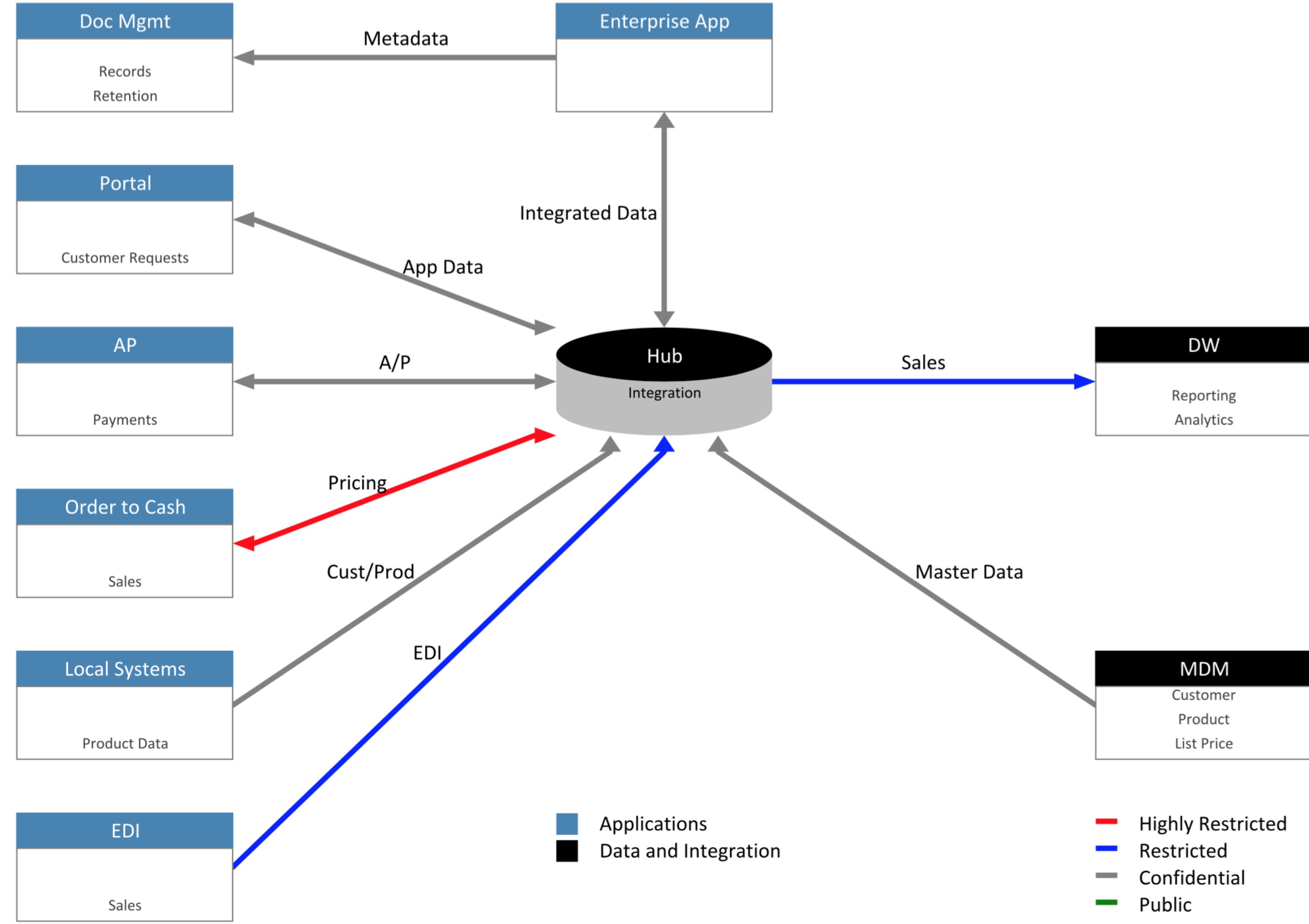
# f50 sunset

```
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>
```

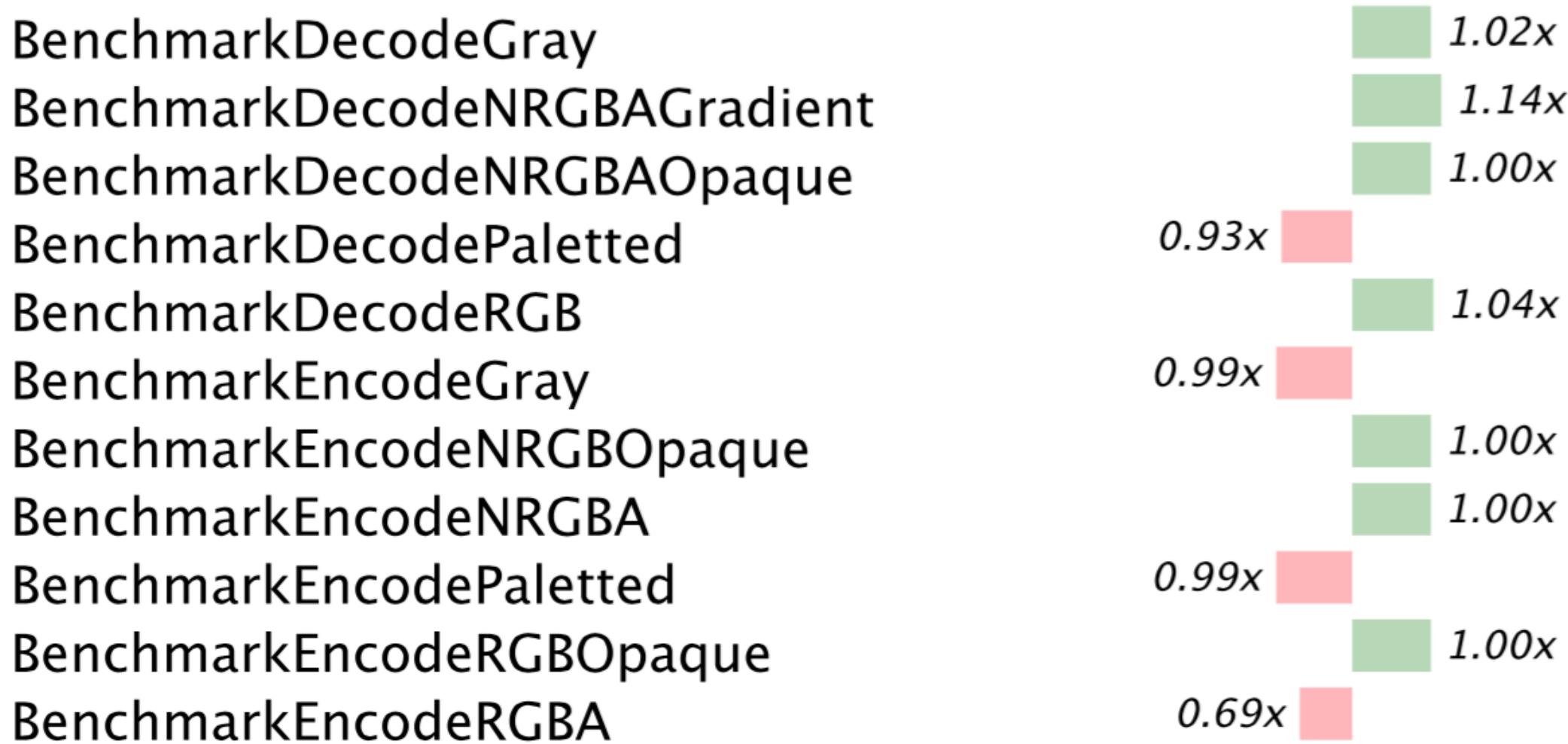
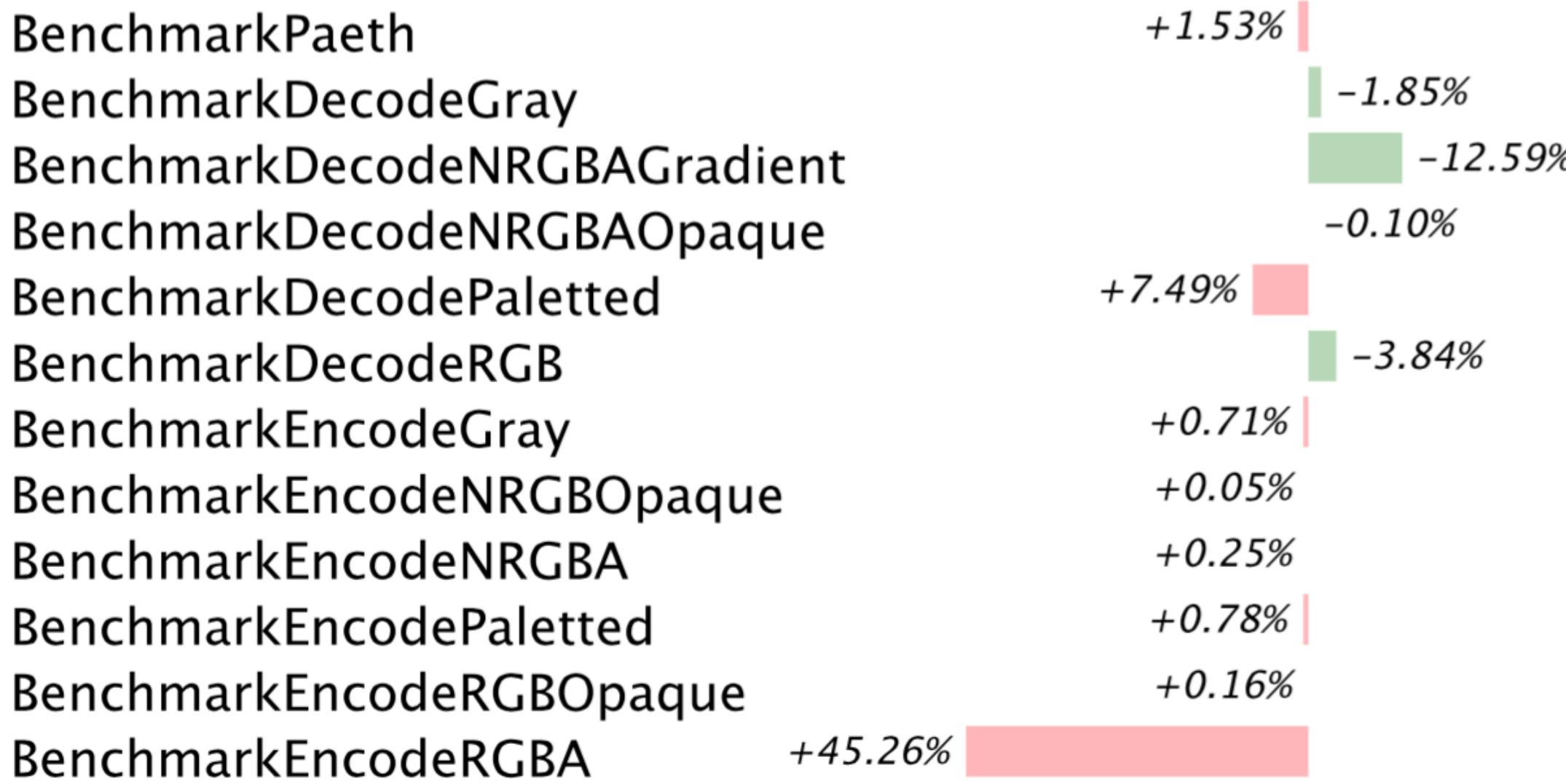


# SVGo Clients



# benchpng.txt

*image/png package: Go 1.3 vs Go 1.4*





```
structlayout -json bytes.Reader | structlayout-svg -t "bytes.Reader" > reader.svg
```

# Go Programming Language Release History

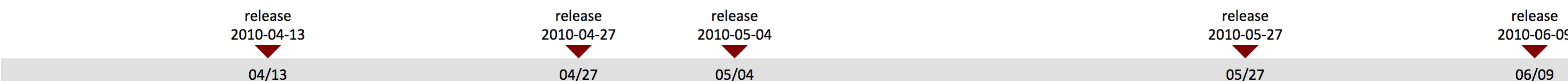
4Q 2009



1Q 2010



2Q 2010



3Q 2010



4Q 2010





## Sample Bullet Graph

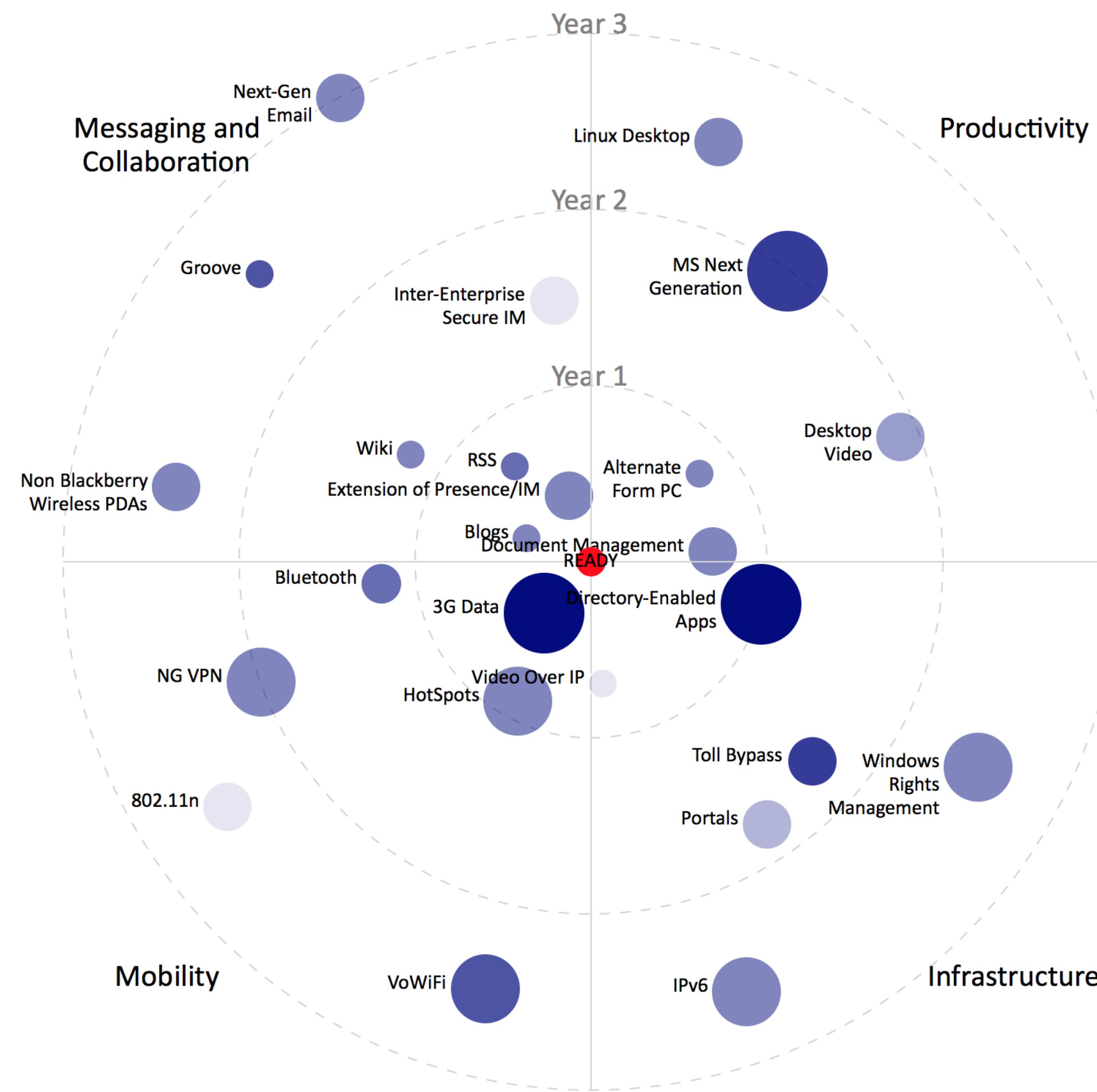
The bullet graph features a single, primary measure (for example, current year-to-date revenue) compares that measure to one or more other measures to enrich its meaning, for example, compared to a target), and displays it in the context of qualitative ranges of performance, such as poor, satisfactory,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



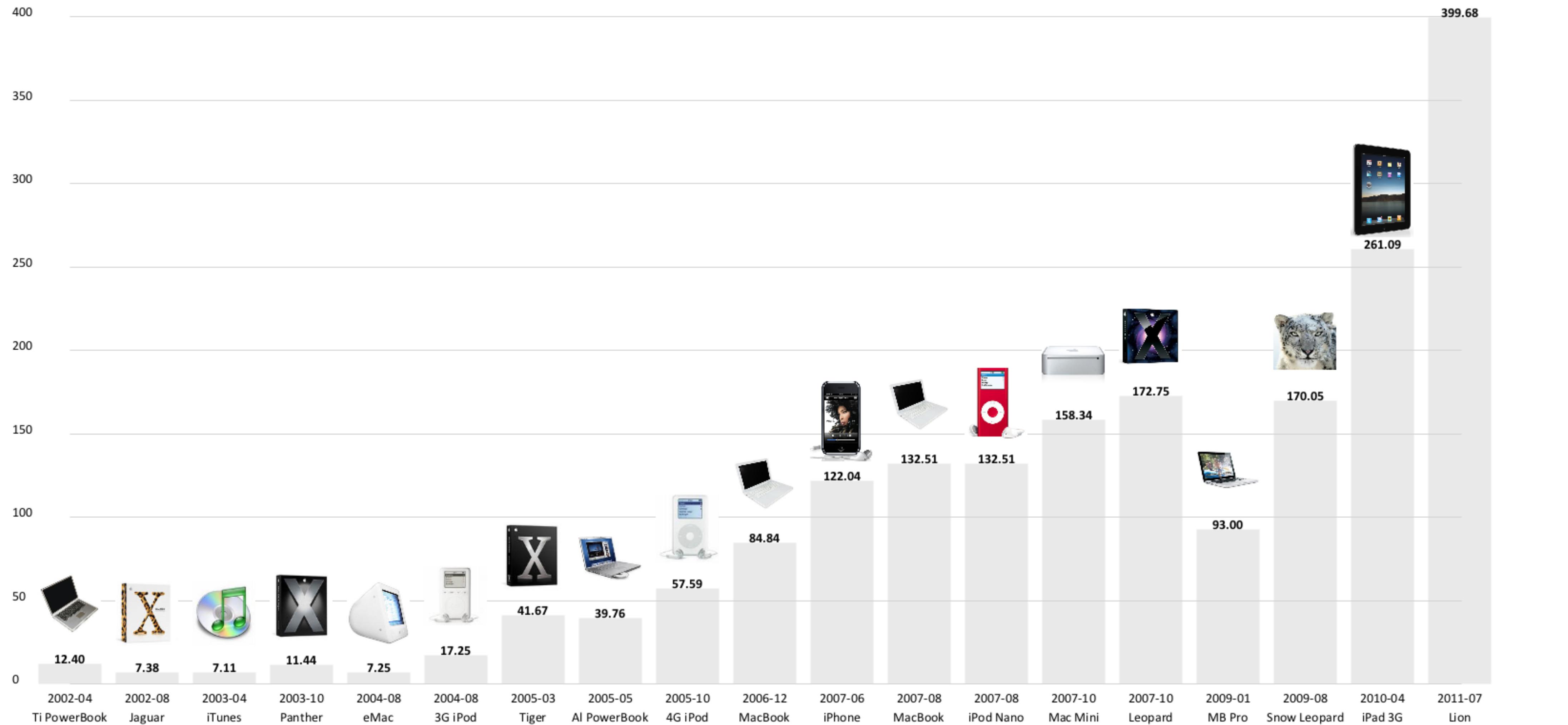
# PLANNING TO PLAN



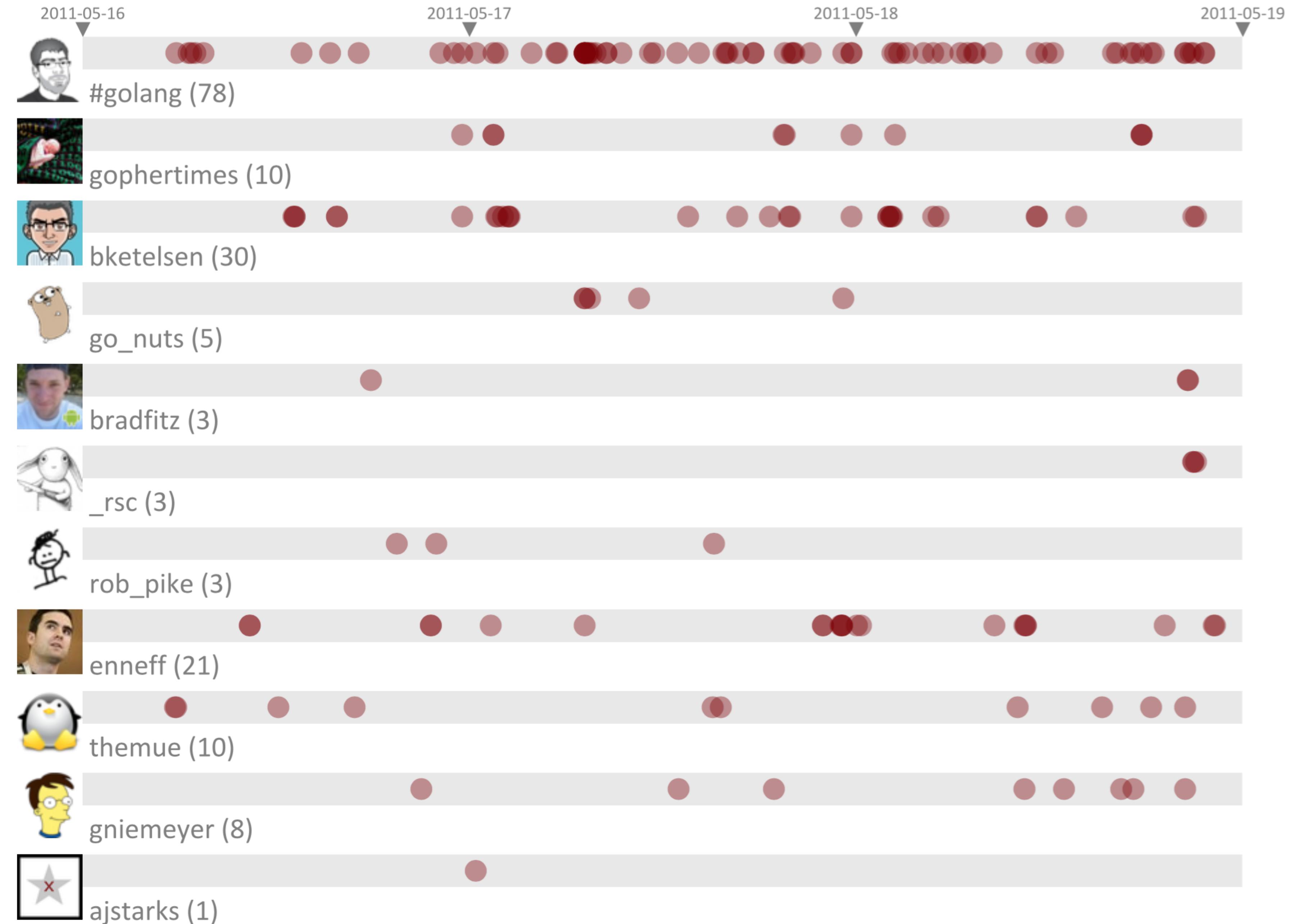




# Apple Purchases and Stock Price



# Twitter Update Frequency



You are here



# JONES

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

# MONTIEL



# DKNG

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

# DDL



# ANDERSON

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

# DKNG



CONTINO  
CASSARO



PUTNAM  
STOCKS



WHITE  
TAYLOR



HALL  
WARREN



STRAWBERRY LUNA  
DOUBLENAUT



JONES  
MONTIEL



DKNG  
DDL



SHAWNA X  
STEVENS



TAYLOR  
JONES



PUTNAM  
RAJKUMAR



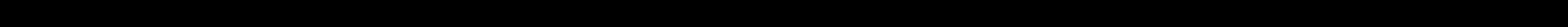
REYES  
WHITE



ANDERSON  
DKNG



TAYLOR  
WHITE



RAJKUMAR  
ANDERSON



ANDERSON  
WHITE





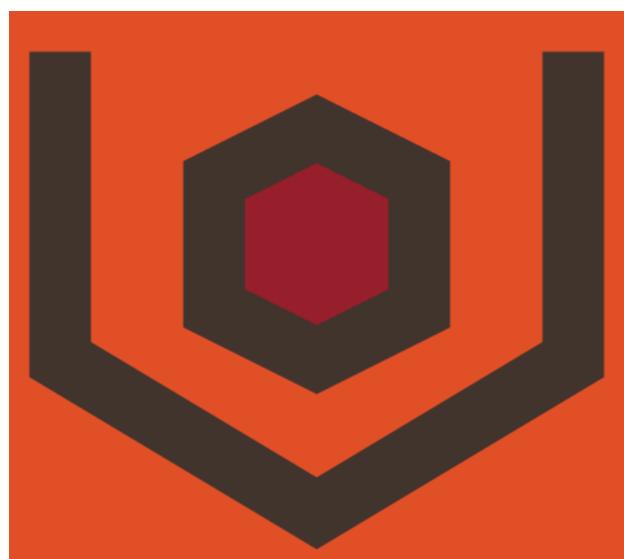








```
canvas.Def()  
canvas.Gid("unit")  
canvas.Polyline(xl, yl, "fill:none")  
canvas.Polygon(xp, yp)  
canvas.Gend()
```

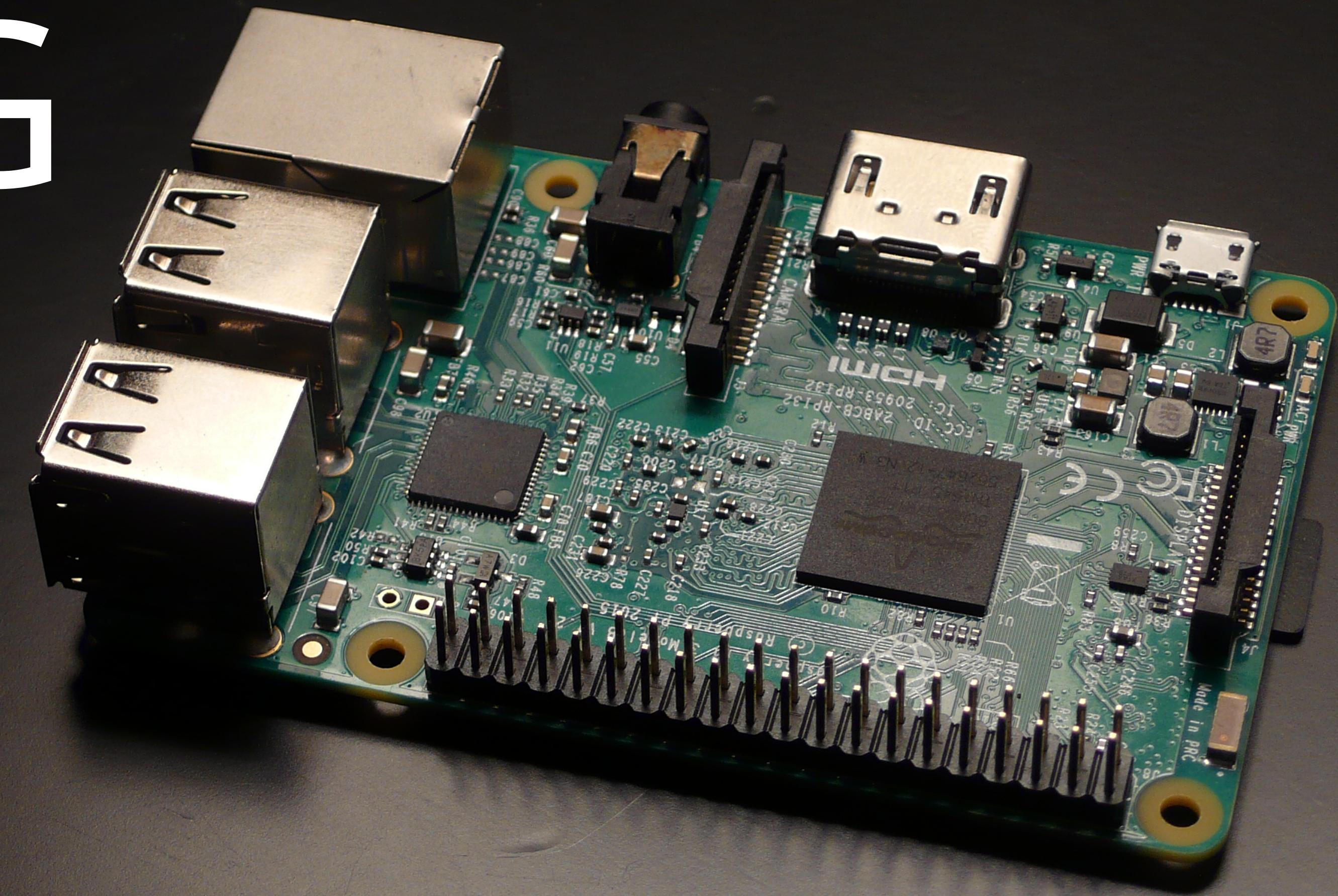


```
canvas.Gid("runit")  
canvas.TranslateRotate(150, 180, 180)  
canvas.Use(0, 0, "#unit")  
canvas.Gend()  
canvas.Gend()  
canvas.DefEnd()
```



```
for y := 0; < height; y += 130 {  
    for x := 100; x < width; x+=100 {  
        canvas.Use(x, y, "#unit")  
        canvas.Use(x, y, "#runit")  
    }  
}
```

# OpenVG



- 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  
@ejstarks  
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 surprised at the size of the resulting binary.

```
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, font string, size int)

**TextMid** (x, y VGfloat, s, font string, size int)

**TextEnd** (x, y VGfloat, s, font string, size int)

Possible Light Rain

83°



90% Chance of precipitation

Thursday August 2

6:26 pm

'All I Did Was Be Black': Police Are Called on College Student Eating Lunch

Tennessee Primary Election Results

Bipartisan Senate Group Pushes 'Crushing' Punishments to Thwart Russia

Trump Officials on Russian Meddling Threat: Democracy 'Is in the Crosshairs'

Are Journalists the Enemy of the People? Ivanka Trump Says They're Not

Mostly Cloudy

47°



(feels like 45°)

Sunday March 13

3:03 am

Farhad and Mike Discuss the Apple Case and a Go-Playing Computer Program  
Getting the Weather Report From Windows  
The Government Answers Apple in the iPhone Case  
Apple and U.S. Bitterly Turn Up Volume in iPhone Privacy Fight  
F.C.C. Proposes Privacy Rules for Internet Providers



MECHANICAL DRAWING

GEOMETRICAL DRAWING

Mostly Cloudy

36° 

(feels like 29°)

29% Chance of precipitation

Tuesday February 23

9:41 pm

Justice Alito Addresses Prospect of an 8-Member Court

Michigan: State Senate Approves Aid to Help Flint Residents Pay Water Bills

Told to 'Take the Gloves Off,' John Kasich Says the Race Is Out of His Hands

Utah: Polygamist Sect Leaders Are Arrested on Fraud Charges

North Carolina: State Lawmakers Target Restroom Anti-Bias Law in Charlotte



HDMI Pi

Clear

66°



Friday April 14

5:55 pm

White House to Keep Its Visitor Logs Secret

Georgia Officers are Fired After Kicking Man at Traffic Stop

Does a Uniform Keep Officers in Line? The Baltimore Chief Thinks So

Wisconsin Man Who Mailed Manifesto to Trump Is Captured After 10-Day Manhunt

California Today: GoPros, Audiobooks and Other Perks of the Library



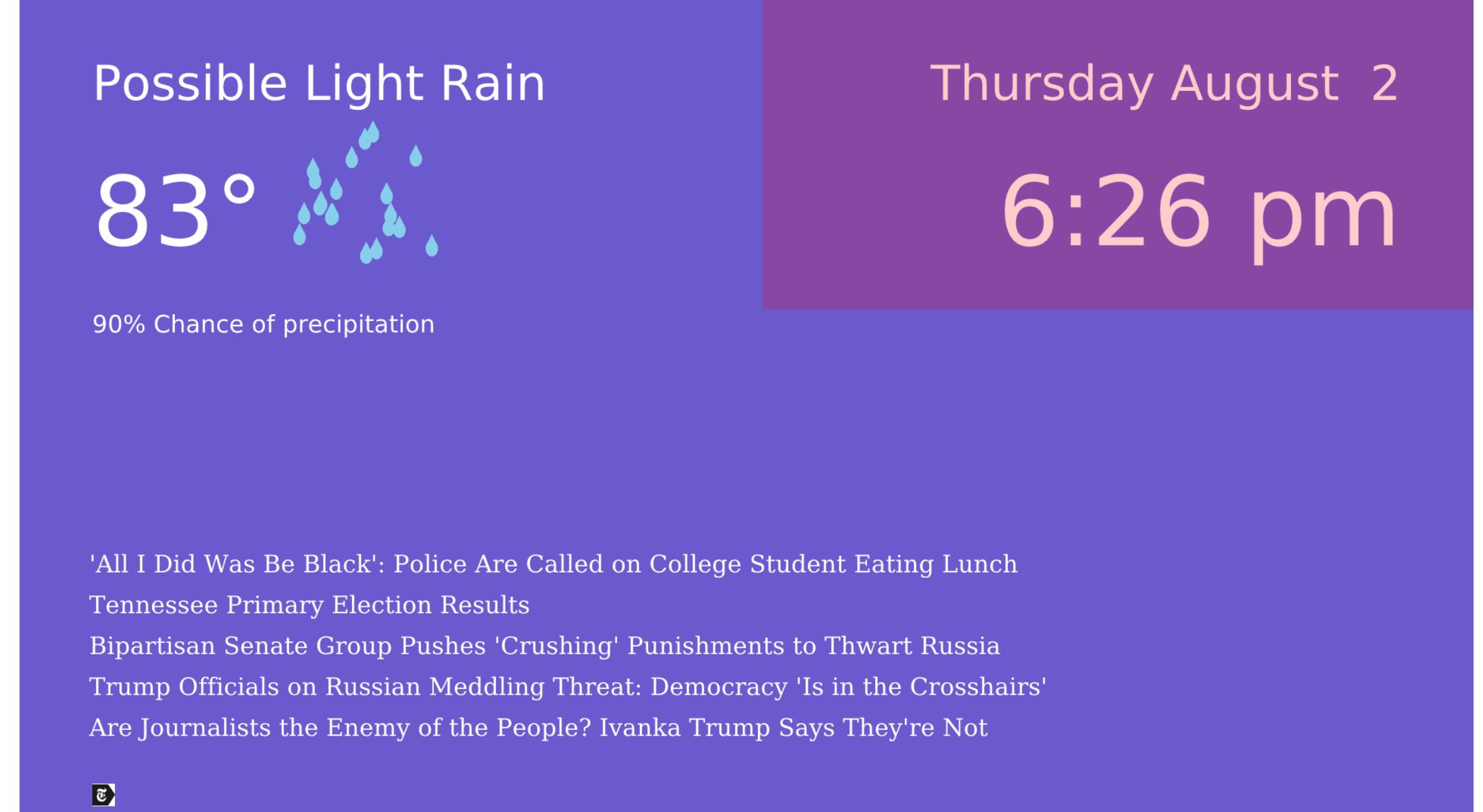
```
func main() {  
  
    //...  
  
    dateticker := time.NewTicker(1 * time.Minute)  
    weatherticker := time.NewTicker(5 * time.Minute)  
    headticker := time.NewTicker(10 * time.Minute)  
    sigint := make(chan os.Signal, 1)  
    signal.Notify(sigint, os.Interrupt)  
  
    for {  
        select {  
            case <-weatherticker.C:  
                canvas.weather(*location)  
            case <-dateticker.C:  
                canvas.clock(*smartcolor)  
            case <-headticker.C:  
                canvas.headlines(*section, *thumb)  
            case <-sigint:  
                openvg.Finish()  
                os.Exit(0)  
        }  
    }  
}
```



```
func main() {  
  
    //...  
  
    dateticker := time.NewTicker(1 * time.Minute)  
    weatherticker := time.NewTicker(5 * time.Minute)  
    headticker := time.NewTicker(10 * time.Minute)  
    sigint := make(chan os.Signal, 1)  
    signal.Notify(sigint, os.Interrupt)  
  
    for {  
        select {  
            case <-weatherticker.C:  
                canvas.weather(*location)  
            case <-dateticker.C:  
                canvas.clock(*smartcolor)  
            case <-headticker.C:  
                canvas.headlines(*section, *thumb)  
            case <-sigint:  
                openvg.Finish()  
                os.Exit(0)  
        }  
    }  
}
```



```
func main() {  
  
    //...  
  
    dateticker := time.NewTicker(1 * time.Minute)  
    weatherticker := time.NewTicker(5 * time.Minute)  
    headticker := time.NewTicker(10 * time.Minute)  
    sigint := make(chan os.Signal, 1)  
    signal.Notify(sigint, os.Interrupt)  
  
    for {  
        select {  
            case <-weatherticker.C:  
                canvas.weather(*location)  
            case <-dateticker.C:  
                canvas.clock(*smartcolor)  
            case <-headticker.C:  
                canvas.headlines(*section, *thumb)  
            case <-sigint:  
                openvg.Finish()  
                os.Exit(0)  
        }  
    }  
}
```



```
func main() {  
  
    //...  
  
    dateticker := time.NewTicker(1 * time.Minute)  
    weatherticker := time.NewTicker(5 * time.Minute)  
    headticker := time.NewTicker(10 * time.Minute)  
    sigint := make(chan os.Signal, 1)  
    signal.Notify(sigint, os.Interrupt)  
  
    for {  
        select {  
            case <-weatherticker.C:  
                canvas.weather(*location)  
            case <-dateticker.C:  
                canvas.clock(*smartcolor)  
            case <-headticker.C:  
                canvas.headlines(*section, *thumb)  
            case <-sigint:  
                openvg.Finish()  
                os.Exit(0)  
        }  
    }  
}
```



# deck



a Go package for presentations

```
<deck>
  <slide bg="black" fg="white">
    <ellipse xp="50" yp="0" wp="100" hr="100" color="blue"/>
    <text      xp="50" yp="25" sp="10" align="c">hello, world</text>
  </slide>
</deck>
```



```
Start the deck <deck>

Set the canvas size <canvas width="1200" height="900"/>

Begin a slide <slide bg="white" fg="black">

Draw some text <text xp="50" yp="90" sp="5" align="c">Deck elements</text>

Place an image <image name="follow.jpg" caption="Dreams" xp="70" yp="60" width="640" height="480" scale="80"/>

Make a bullet list <list xp="10" yp="70" sp="3" type="bullet">

    <li>text, image, list</li>
    <li>rect, ellipse, polygon</li>
    <li>line, arc, curve</li>

End the list </list>

Draw a rectangle <rect xp="15" yp="20" wp="8" hp="6" color="rgb(127,0,0)"/>

Draw a ellipse <ellipse xp="27.5" yp="20" wp="8" hp="6" color="rgb(0,127,0)"/>

Draw a polygon <polygon xc="37 37 45" yc="17 23 20" color="rgb(0,0,127)"/>

Draw a line <line xp1="50" yp1="20" xp2="60" yp2="20"/>

Draw an arc <arc xp="70" yp="20" wp="10" hp="8" a1="0" a2="180" color="rgb(0,0,127)"/>

Draw a bezier <curve xp1="80" yp1="20" xp2="95" yp2="30" xp3="90" yp3="20"/>

End the slide </slide>

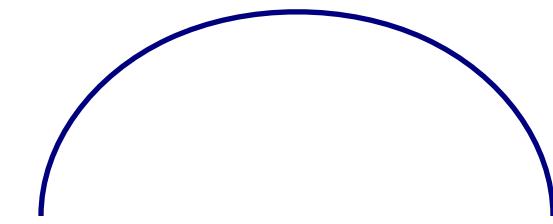
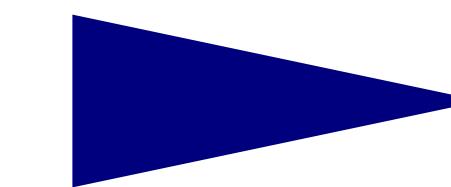
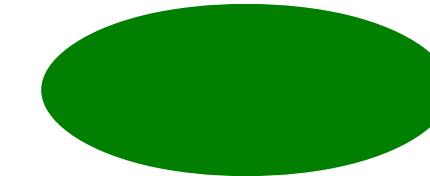
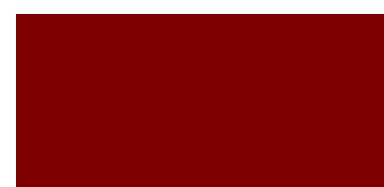
End of the deck </deck>
```

# Deck elements

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



Dreams



**Percent Grid**

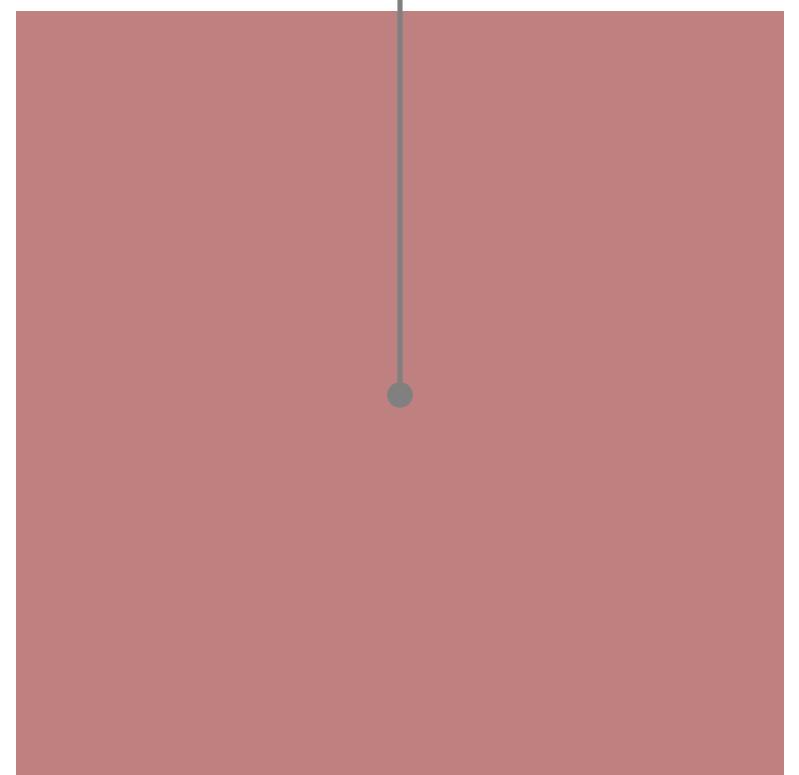
10%, 50%

Hello

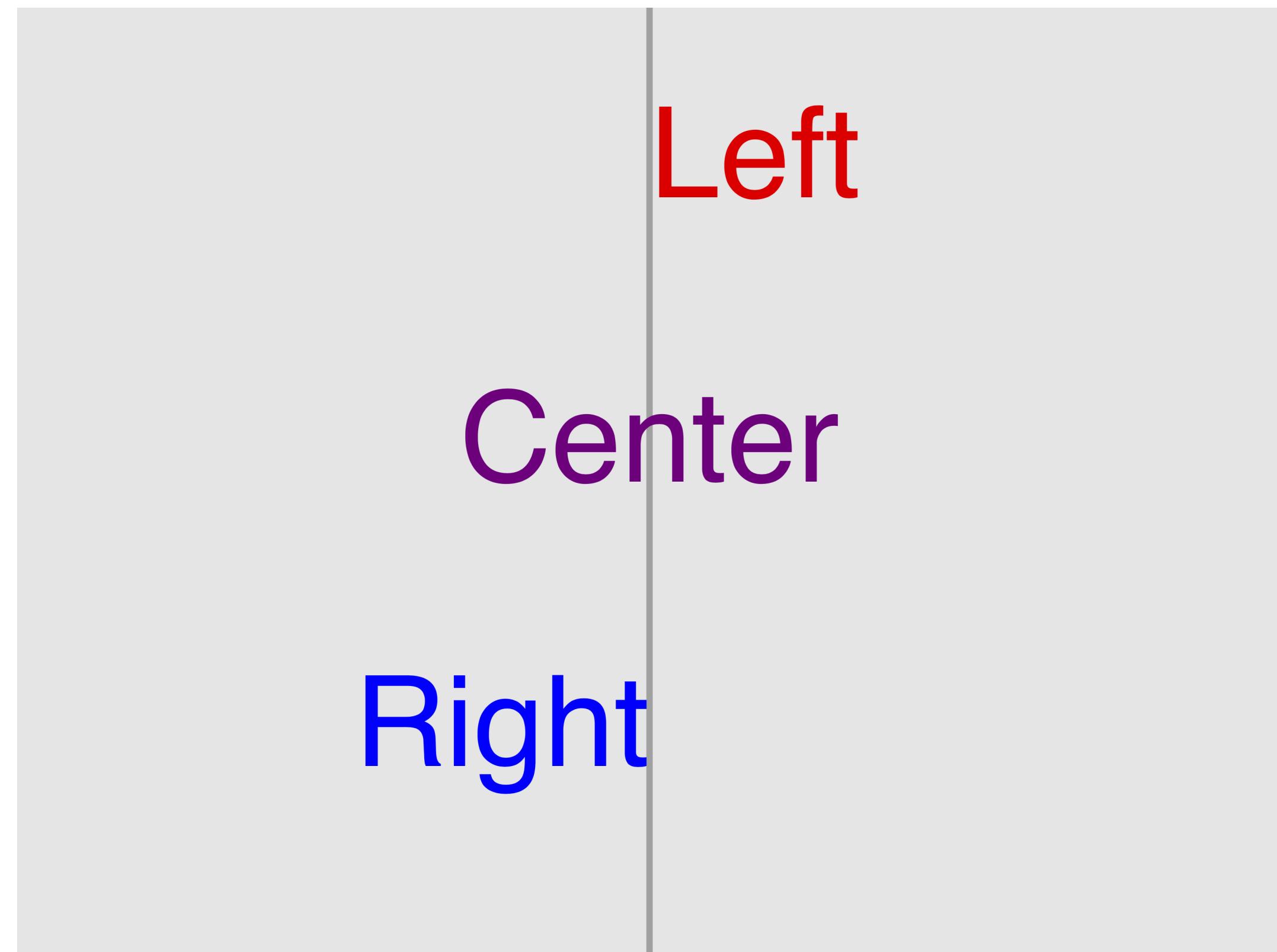
50%, 50%



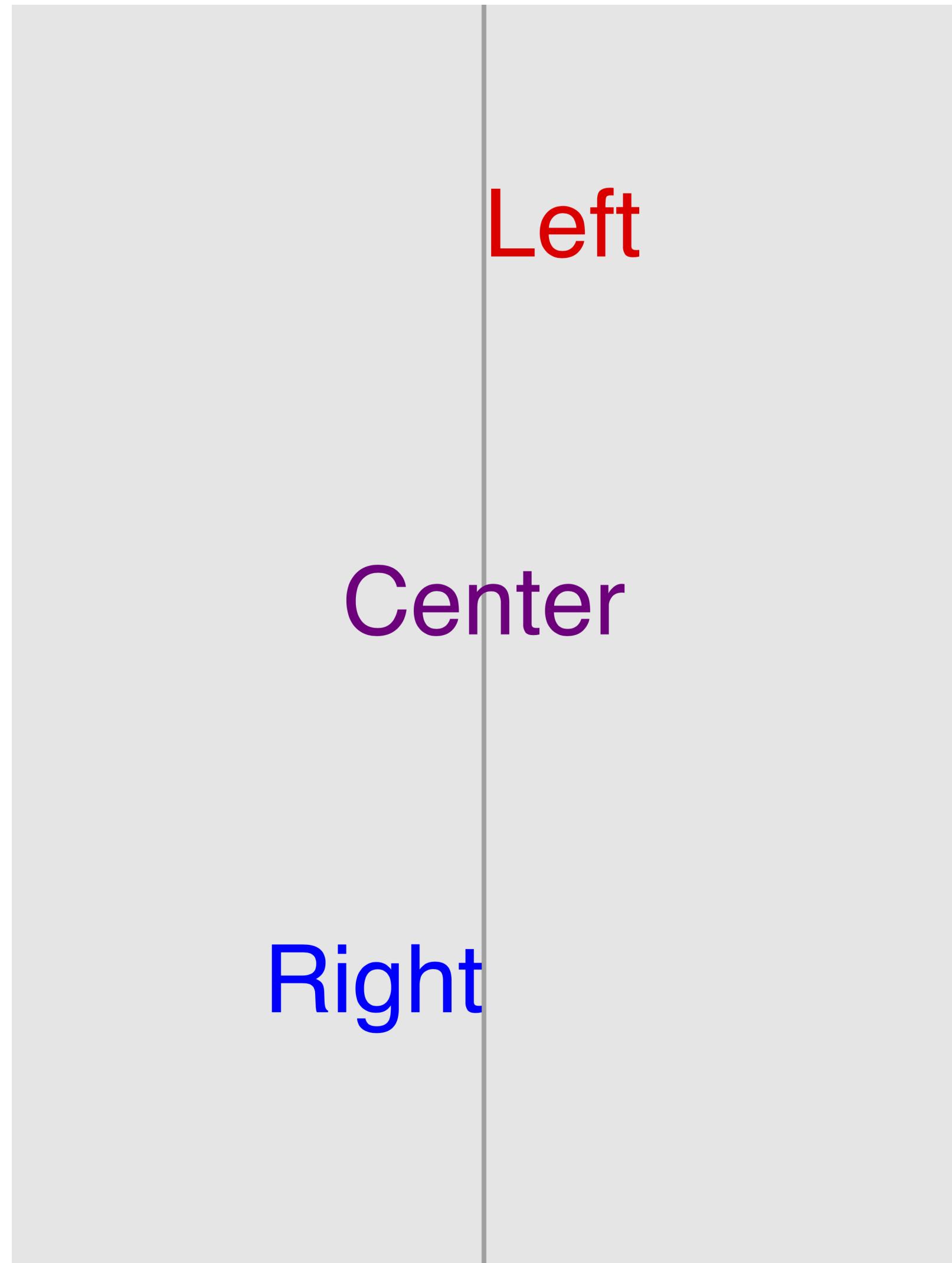
90%, 50%



Percentage-based layout



Landscape



Portrait

# Scaling the canvas

Data      Process

deck

PDF (pdfdeck)

PNG (pngdeck)

SVG (svgdeck)

OpenVG (vgdeck)

# deck/generate text and list functions

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

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

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

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

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

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

# deck/generate graphic functions

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

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

**Square** (x, y, w float64, color string, op ...float64)

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

**Circle** (x, y, w float64, color string, op ...float64)

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

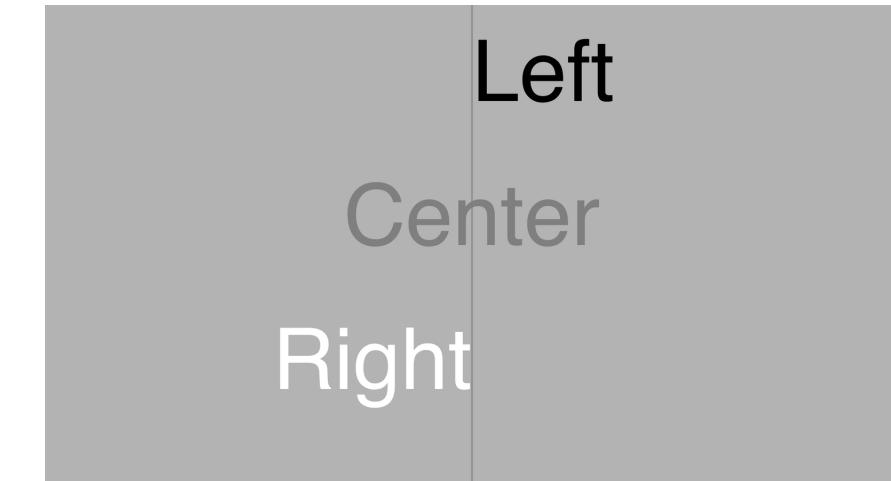
**Line** (x1, y1, x2, y2, size float64, color string, op ...float64)

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

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

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

```
deck.StartSlide("rgb(180,180,180)")
// ...
deck.EndSlide()
```



```
deck.StartSlide("blanchedalmond")
// ...
deck.EndSlide()
```

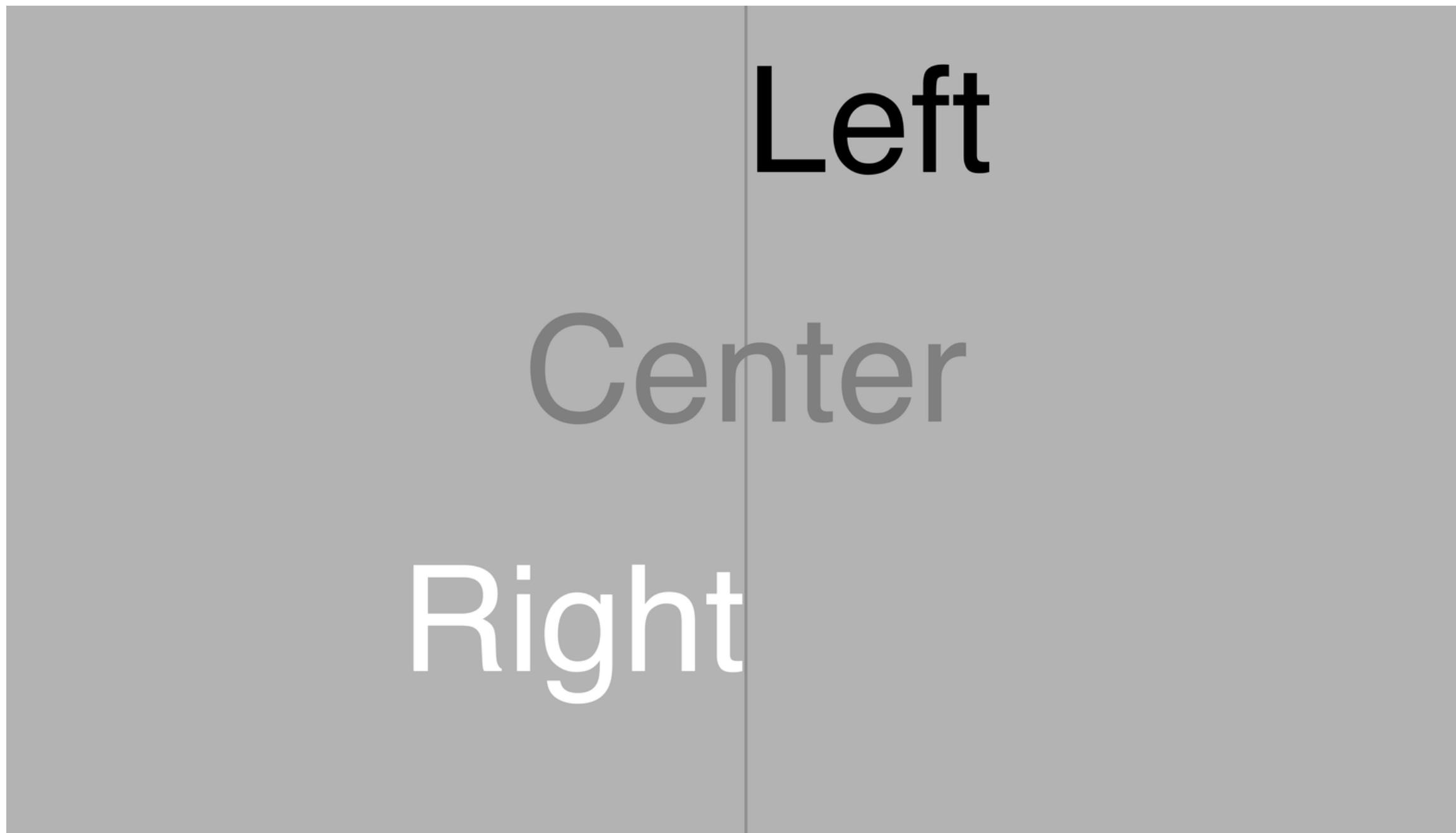


```
deck.StartSlide("black", "white")
// ...
deck.EndSlide()
```



```
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("blanchedalmond", "black")
deck.Text(10, 80, "Important Items", "sans", 5, "")
deck.List(10, 60, 4, 1.4, 50, items, "bullet", "sans", "red")
deck.EndSlide()
```

## Important Items

- First
- Second
- Third
- Fourth
- Fifth

```
// Picture with text annotation
quote := "Tony Stark was able to build this in a cave. With a box of scraps!"
deck.StartSlide("black", "white")
deck.Image(50, 50, 1600, 681, "cave.png", "https://youtu.be/g1fzMbAr1u0?t=253")
deck.Rect(70, 60, 40, 40, "black", 40)
deck.TextBlock(45, 70, quote, "sans", 5, 45, "")
deck.EndSlide()
```



# A View of User Experience: Designing for People

Anthony Starks / ajstarks@gmail.com / @ajstarks

Design



What works good is better than what looks good, because what works good lasts.

Ray Eames

# 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 last

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

```

deck
  canvas 1200 900
    slide "white" "black"
      ctext "Deck elements" 50 90 5
      cimage "follow.jpg" "Dreams" 70 60 640 480 50

    blist 10 70 3
      li "text, image, list"
      li "rect, ellipse, polygon"
      li "line, arc, curve"
    elist

    rect   15 20 8 6           "rgb(127,0,0)"
    ellipse 27.5 20 8 6        "rgb(0,127,0)"
    polygon "37 37 45" "17 23 20" "rgb(0,0,127)"
    line    50 20 60 20 0.2    "rgb(127,0,0)"
    arc     70 20 10 8 0 180 0.2 "rgb(0,127,0)"
    curve   80 20 95 30 90 20 0.2 "rgb(0,0,127)"

  eslide
edeck

```

## Deck elements

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



Dreams



# decksh

# Deck Web API

deckd -dir [start dir] -listen [address:port] -maxupload [bytes]

GET	/	List the API
GET	/deck/	List the content on the server
GET	/deck/?filter=[type]	List content filtered by deck, image, video
POST	/deck/content.xml?cmd=1s	Play a deck with the specified duration
POST	/deck/content.xml?cmd=stop	Stop playing a deck
POST	/deck/content.xml?slide=[num]	Play deck starting at a slide number
DELETE	/deck/content.xml	Remove content
POST	/upload/ Deck:content.xml	Upload content
POST	/table/ Deck:content.txt	Generate a table from a tab-separated list
POST	/table/?textsize=[size]	Specify the text size of the table
POST	/media/ Media:content.mov	Play the specified video



René Magritte

The False Mirror

LG



# 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

# 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.

The image contains two screenshots. The top screenshot shows a PDF viewer window titled "deck-fira-4x3.pdf (page 53 of 57)". It displays a quote in white text on a dark blue background: "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." Below the quote, it says "Less is exponentially more" and "Rob Pike". The bottom screenshot shows a web browser window titled "command center: Less is e...". The URL in the address bar is "commandcenter.blogspot.com/2012/06/less-is-exp...". The page content is identical to the quote in the PDF, along with some additional text and a "Read more" link.

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

What you're given is a set of powerful but easy to understand, easy to use building blocks from which you can assemble—compose—a solution to your problem. It might not end up quite as fast or as sophisticated or as ideologically motivated as the solution you'd write in some of those other languages, but it'll almost certainly be easier to write, easier to read, easier to understand, easier to maintain, and maybe safer.

To put it another way, oversimplifying of course:

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.

C++ programmers *don't* come to Go because they have fought hard to gain exquisite control of their programming domain, and don't want to surrender any of it. To them, software isn't just about getting the job done, it's about doing it a certain way.

The issue, then, is that Go's success would contradict their world view.

BOS



SFO

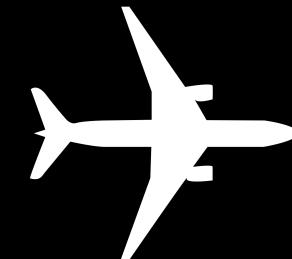
Virgin America 351

Gate B38

8:35am

On Time

JFK



IND

US Airways 1207

Gate C31C

5:35pm

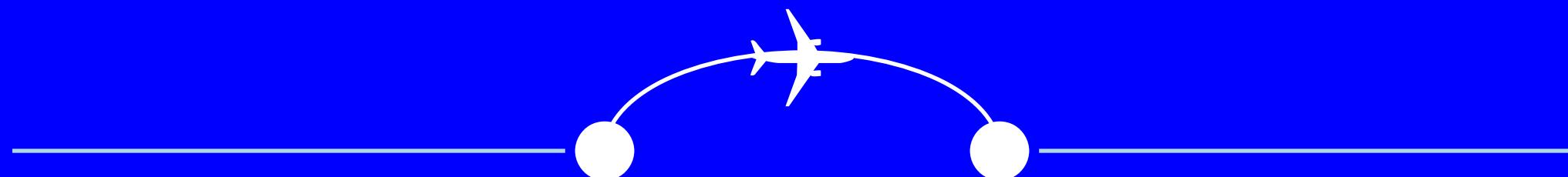
Delayed

# Flight Information

Los Angeles (LAX)



New York/Newark (EWR)



Distance Traveled

1,958 mi

3,151 km

Distance to Destination

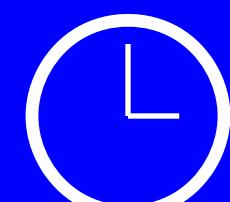
596 mi

798 km

Time to Destination

Estimated time of arrival

Local time of arrival



1:20

12:14 am

12:14 am

Ground speed



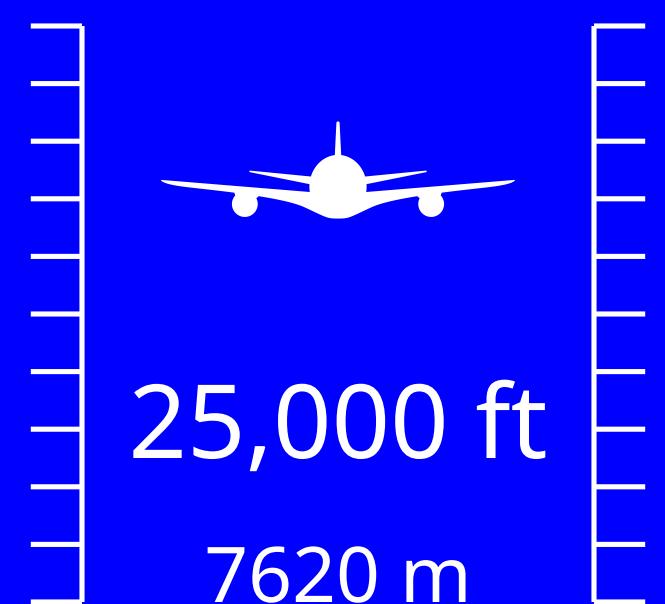
Headwind



Outside Temperature



Current Altitude



AAPL	179.98	3.04 (1.72%)
AMZN	1578.89	27.03 (1.74%)
FB	185.23	2.89 (1.58%)
GOOGL	1160.84	31.46 (2.79%)
MSFT	96.54	2.11 (2.23%)

AAPL	181.72	1.74 (0.97%)
AMZN	1598.39	19.50 (1.24%)
FB	184.76	-0.47 (-0.25%)
GOOGL	1165.93	5.09 (0.44%)
MSFT	96.77	0.23 (0.24%)

AAPL	175.30	-2.72 (-1.53%)
AMZN	1544.93	-26.75 (-1.70%)
FB	172.56	-12.53 (-6.77%)
GOOGL	1100.07	-34.35 (-3.03%)
MSFT	92.89	-1.71 (-1.81%)

```

func stockslide(deck *generate.Deck, symbols []string) {
    x := left
    y := top

    rintr := (top-bottom)/float64(len(symbols))
    size := rintr/2.5
    cintr := size*4

    var color string
    for _, s := range symbols {
        stock, err := stockapi(s)
        if err != nil || stock.Symbol == "" {
            continue
        }
        if stock.Change < 0 {
            color = negcolor
        } else {
            color = poscolor
        }
        deck.Text(x, y, stock.Symbol, "sans", size, "")
        x += cintr * 2
        deck.TextEnd(x, y, fmt.Sprintf("%.2f", stock.Price), "sans", size, "")
        x += cintr
        deck.TextEnd(x, y, fmt.Sprintf("%.2f", stock.Change), "sans", size, color)
        x += cintr
        deck.TextEnd(x, y, fmt.Sprintf("(%.2f%%)", stock.PctChange), "sans", size, color)
        x = left
        y -= rintr
    }
    deck.Text(footx, footy, time.Now().Format("2006-01-02 15:04:05"), "sans", 1.5, "yellow")
}

```

AAPL	181.72	1.74 (0.97%)
AMZN	1598.39	19.50 (1.24%)
FB	184.76	-0.47 (-0.25%)
GOOGL	1165.93	5.09 (0.44%)
MSFT	96.77	0.23 (0.24%)

2018-03-12 21:05:06

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

**what would Go do**

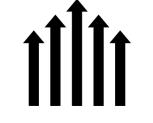
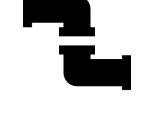
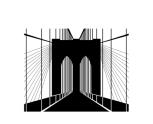
Andrew Mackenzie-Ross



# go

bug	start a bug report
build	compile packages and dependencies
clean	remove object files and cached files
doc	show documentation for package or symbol
env	print Go environment information
fix	update packages to use new APIs
fmt	gofmt (reformat) 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 or modules
mod	module maintenance
run	compile and run Go program
test	test packages
tool	run specified go tool
version	print Go version
vet	report likely mistakes in packages

# Go Proverbs

-  Don't communicate by sharing memory, share memory by communicating.
-  Concurrency is not parallelism.
-  Channels orchestrate; mutexes serialize.
-  The bigger the interface, the weaker the abstraction.
-  Make the zero value useful.
-  interface{} says nothing.
-  Gofmt's style is no one's favorite, yet gofmt is everyone's favorite.
-  A little copying is better than a little dependency.
-  Syscall must always be guarded with build tags.
-  Cgo must always be guarded with build tags.
-  Cgo is not Go.
-  With the unsafe package there are no guarantees.
-  Clear is better than clever.
-  Reflection is never clear.
-  Errors are values.
-  Don't just check errors, handle them gracefully.
-  Design the architecture, name the components, document the details.
-  Documentation is for users.
-  Don't panic.



Don't communicate  
by sharing memory,  
share memory by  
communicating.



The bigger the interface,  
the weaker the abstraction.



Make the zero  
value useful.

## Chapter V



## Advice from a Caterpillar

THE Caterpillar and Alice looked at each other for some time in silence: at last the Caterpillar took the hookah out of its mouth, and addressed her in a languid, sleepy voice. “Who are you?” said the Caterpillar. This was not an encouraging opening for a conversation. Alice replied, rather shyly, “I—I hardly know, sir, just at present—at least I know who I was when I got up this morning, but I think I must have been changed several times since then.”



"You promised to tell me your history, you know," said Alice, "and why it is you hate—C and D," she added in a whisper, half afraid that it would be offended again. "Mine is a long and a sad tale!" said the Mouse, turning to Alice, and sighing. "It is a long tail, certainly," said Alice, looking down with wonder at the Mouse's tail; "but why do you call it sad?" And she kept on puzzling about it while the Mouse was speaking, so that her idea of the tale was something like this:—

Fury said to a mouse, That he met in the house,  
"Let us both go to law:I will prosecute YOU—Come, I'll take no denial; We must have a trial:For really this morning I've nothing to do."  
Said the mouse to the cur, "Such a trial, dear Sir, With no jury or judge, would be wasting our breath."  
"I'll be judge, I'll be jury," Said cunning old Fury:  
"I'll try the whole cause, and condemn you to death."

## SVGo Examples



hello.go



SVGo



rl.go



concentric.go



concentric2.go



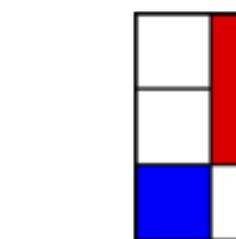
cgrid.go



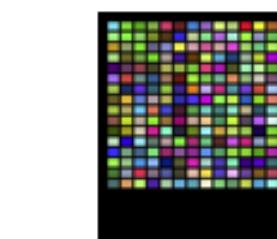
diag.go



shining.go



mondrian.go



richter.go



eclipse.go



pacman.go



pyramid.go



cloud.go



color-clouds.go



creepy.go



d4h.go



sunearth.go



conception.go



conception2.go



randbox.go



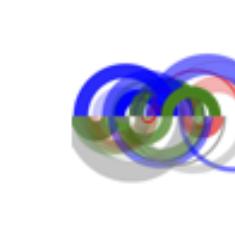
randspot.go



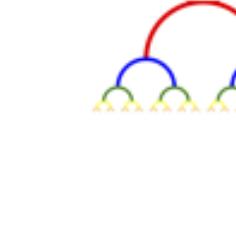
cube.go



schotter.go



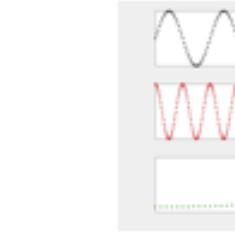
randarc.go



reurse.go



clock.go



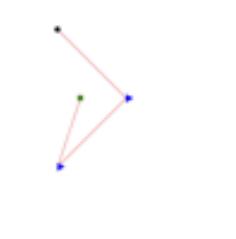
plotfunc.go



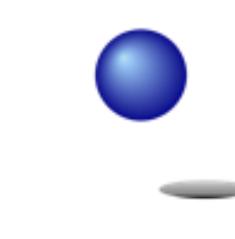
therm.go



pattern.go



marker.go



gradient.go



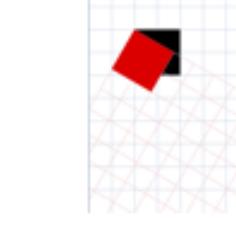
fontrange.go



uni.go



lorem.go



rotate.go



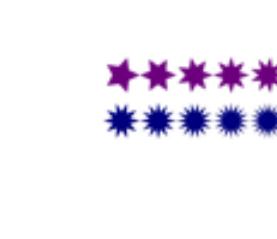
rottext.go



skewabc.go



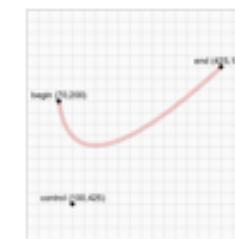
gear.go



star.go



starx.go



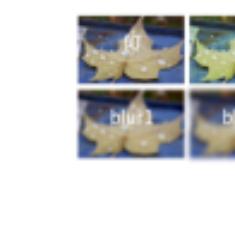
swoosh.go



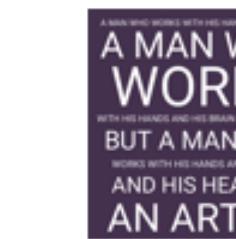
textpath.go



imfade.go



fe.go



artist.go



go.go

codepicdeck

```

package main

import (
    "fmt"
    "github.com/ajstarks/svggo"
    "os"
)

func main() {
    xp := []int{50, 70, 70, 50, 30, 30}
    yp := []int{40, 50, 75, 85, 75, 50}
    xl := []int{0, 0, 50, 100, 100}
    yl := []int{100, 40, 10, 40, 100}
    bgcolor := "rgb(227,78,25)"
    bkcolor := "rgb(153,29,40)"
    stcolor := "rgb(65,52,44)"
    stwidth := 12
    stylefmt := "stroke:%s;stroke-width:%d;fill:%s"
    canvas := svg.New(os.Stdout)
    width, height := 500, 500
    canvas.Start(width, height)
    canvas.Def()
    canvas.Gid("unit")
    canvas.Polyline(xl, yl, "fill:none")
    canvas.Polygon(xp, yp)
    canvas.Gend()
    canvas.Gid("runit")
    canvas.TranslateRotate(150, 180, 180)
    canvas.Use(0, 0, "#unit")
    canvas.Gend()
    canvas.Gend()
    canvas.DefEnd()
    canvas.Rect(0, 0, width, height, "fill:"+bgcolor)
    canvas.Gstyle(fmt.Sprintf(stylefmt, stcolor, stwidth, bkcolor))
    for y := 0; y < height; y += 130 {
        for x := -50; x < width; x += 100 {
            canvas.Use(x, y, "#unit")
            canvas.Use(x, y, "#runit")
        }
    }
    canvas.Gend()
    canvas.End()
}

```



shining.go [8]

```

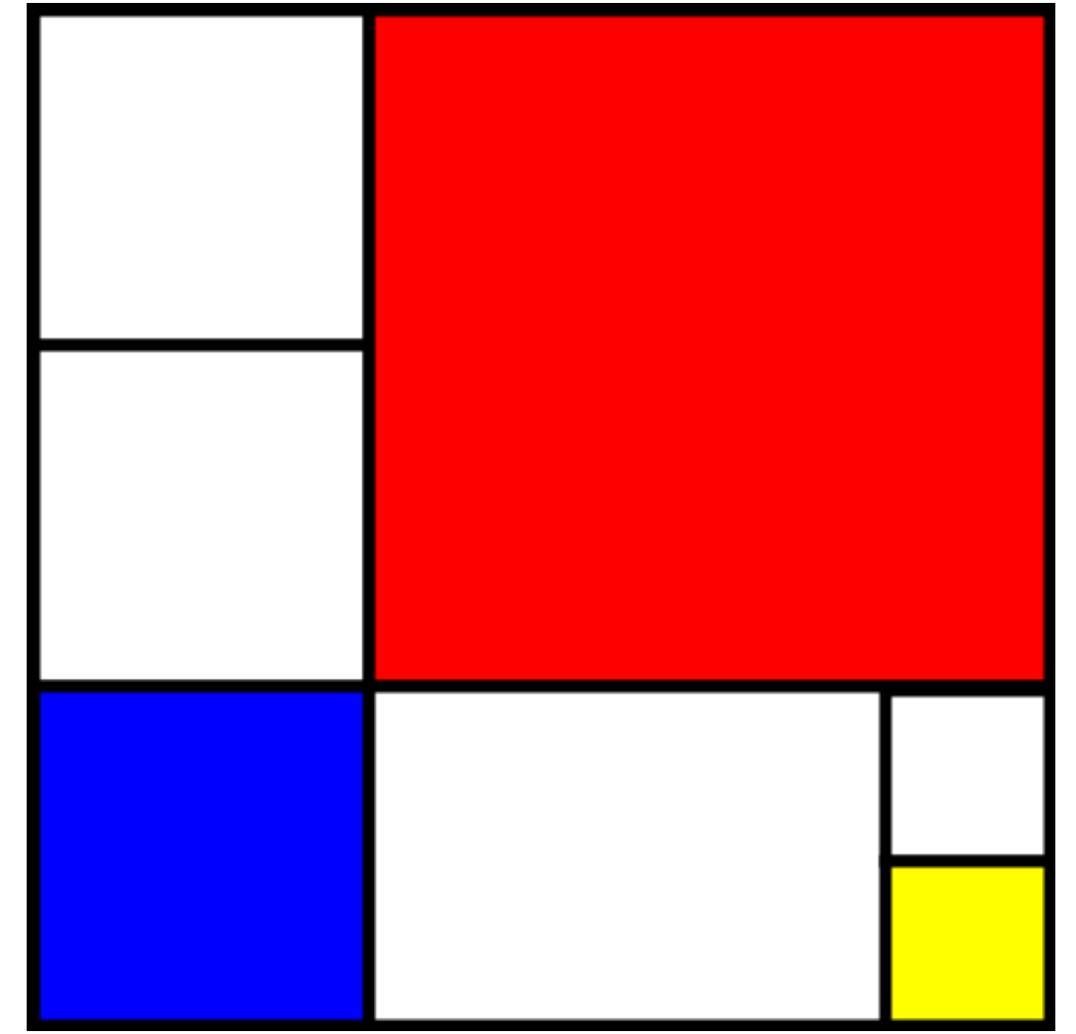
package main

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

var (
    canvas = svg.New(os.Stdout)
    width  = 500
    height = 500
)

// Piet Mondrian - Composition in Red, Blue, and Yellow
func main() {
    w3 := width / 3
    w6 := w3 / 2
    w23 := w3 * 2
    canvas.Start(width, height)
    canvas.Gstyle("stroke:black;stroke-width:6")
    canvas.Rect(0, 0, w3, w3, "fill:white")
    canvas.Rect(0, w3, w3, w3, "fill:white")
    canvas.Rect(0, w23, w3, w3, "fill:blue")
    canvas.Rect(w3, 0, w23, w23, "fill:red")
    canvas.Rect(w3, w23, w23, w3, "fill:white")
    canvas.Rect(width-w6, height-w3, w3-w6, w6, "fill:white")
    canvas.Rect(width-w6, height-w6, w3-w6, w6, "fill:yellow")
    canvas.Gend()
    canvas.Rect(0, 0, width, height, "fill:none;stroke:black;stroke-width:12")
    canvas.End()
}

```



mondrian.go [9]

```

package main

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

var (
    canvas = svg.New(os.Stdout)
    width  = 500
    height = 500
)

func tro() {
    canvas.Rect(100, 100, 100, 100)
    canvas.TranslateRotate(100, 100, 30)
    canvas.Grid(0, 0, width, height, 50, "stroke:lightsteelblue")
    canvas.Rect(0, 0, 100, 100, "fill:red")
    canvas.Gend()
}

func main() {
    canvas.Start(width, height)
    canvas.Grid(0, 0, width, height, 50, "stroke:lightsteelblue")
    tro()
    canvas.End()
}

```



rotate.go [36]

```

package main

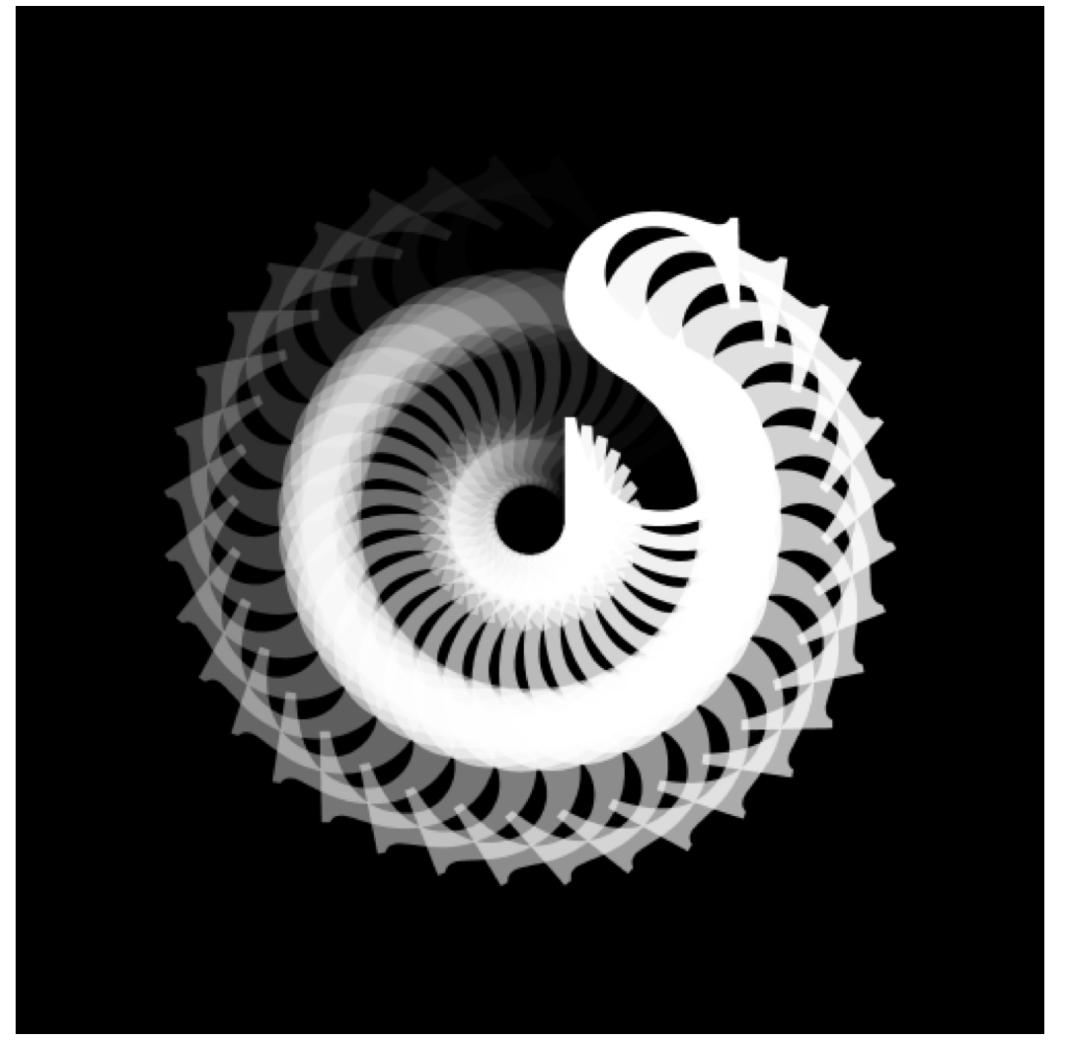
import (
    "os"
    "github.com/ajstarks/svg"
)

var (
    canvas = svg.New(os.Stdout)
    width  = 500
    height = 500
)

func main() {
    a := 1.0
    ai := 0.03
    ti := 10.0

    canvas.Start(width, height)
    canvas.Rect(0, 0, width, height)
    canvas.Gstyle("font-family:serif;font-size:244pt")
    for t := 0.0; t <= 360.0; t += ti {
        canvas.TranslateRotate(width/2, height/2, t)
        canvas.Text(0, 0, "s", canvas.RGBA(255, 255, 255, a))
        canvas.Gend()
        a -= ai
    }
    canvas.Gend()
    canvas.End()
}

```



rottext.go [37]

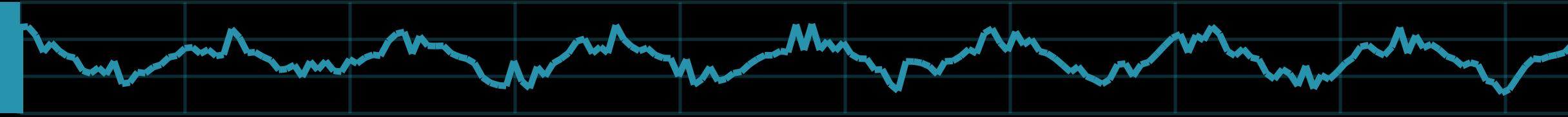
The new masters of our universe are people who are essentially only half-educated.

They have had no exposure to the humanities or the social sciences, the academic disciplines that aim to provide some understanding of how society works, of history and of the roles that beliefs, philosophies, laws, norms, religion and customs play in the evolution of human culture

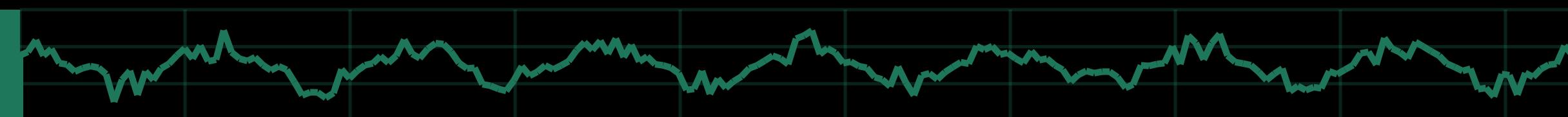
John Naughton, The Guardian, 19 November, 2017



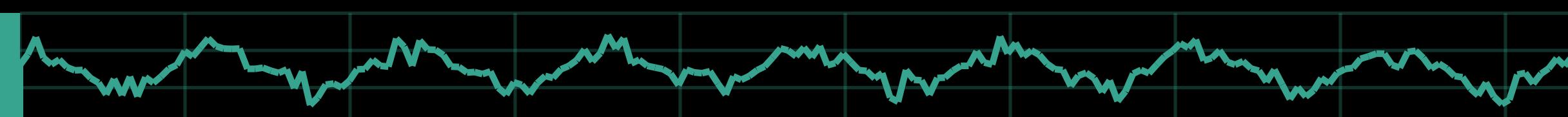
CARDIO  
VASCULAR



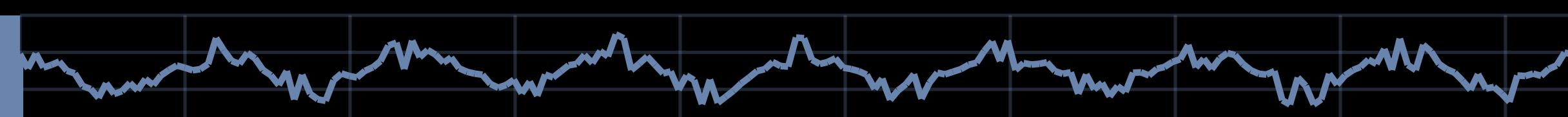
METABOLIC  
LEVELS



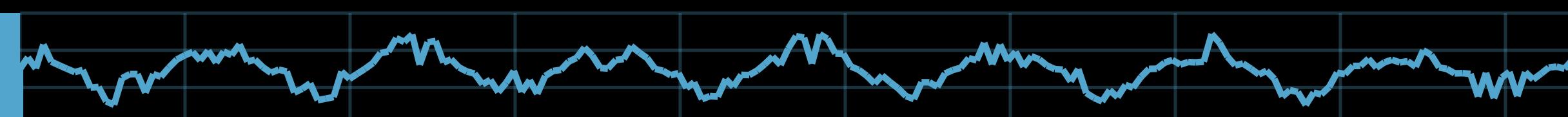
CENTRAL  
NERV. SYSTEM



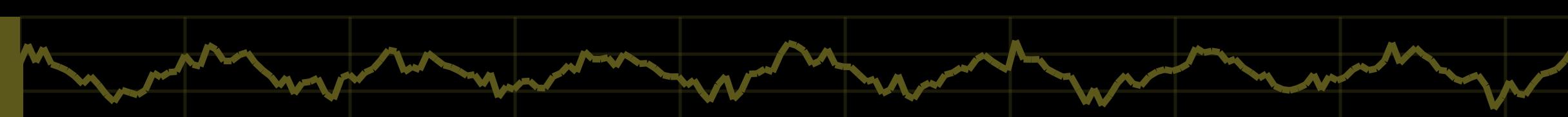
PULMONARY  
FUNCTION



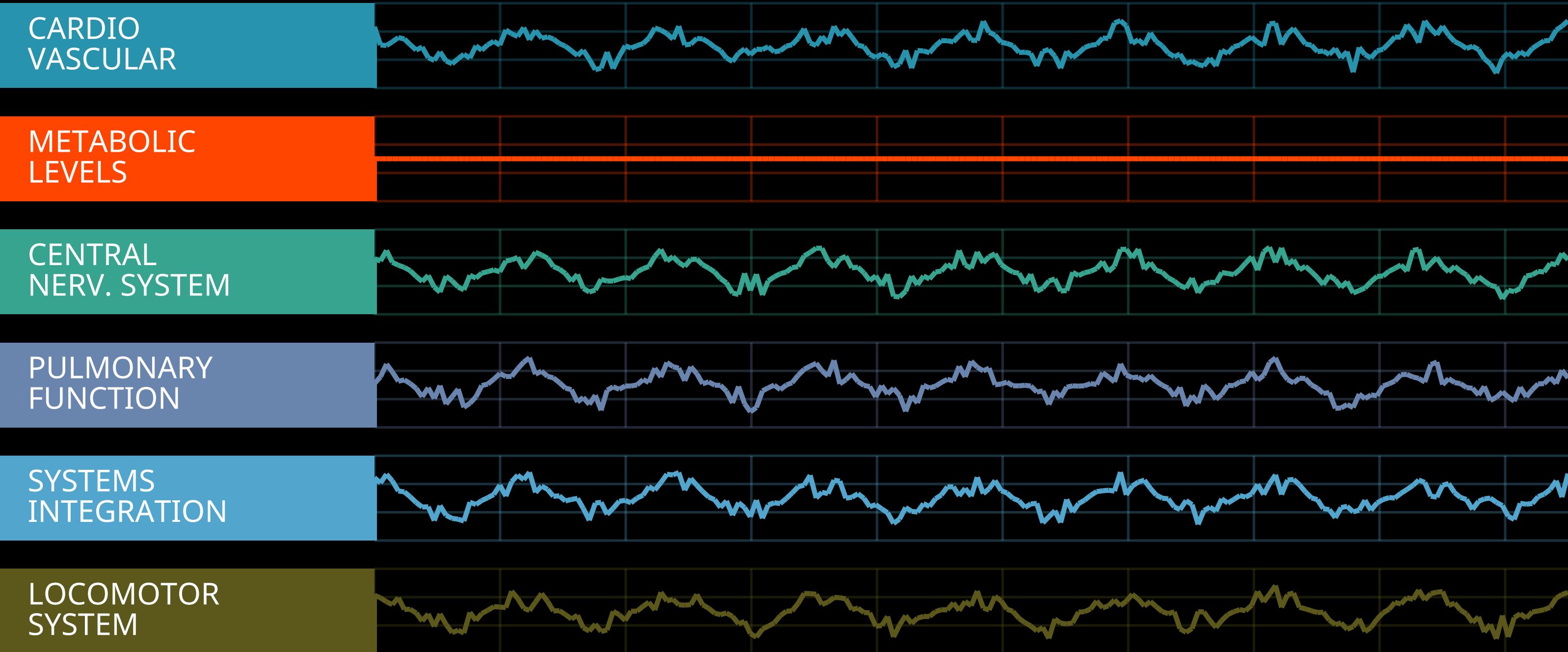
SYSTEMS  
INTEGRATION

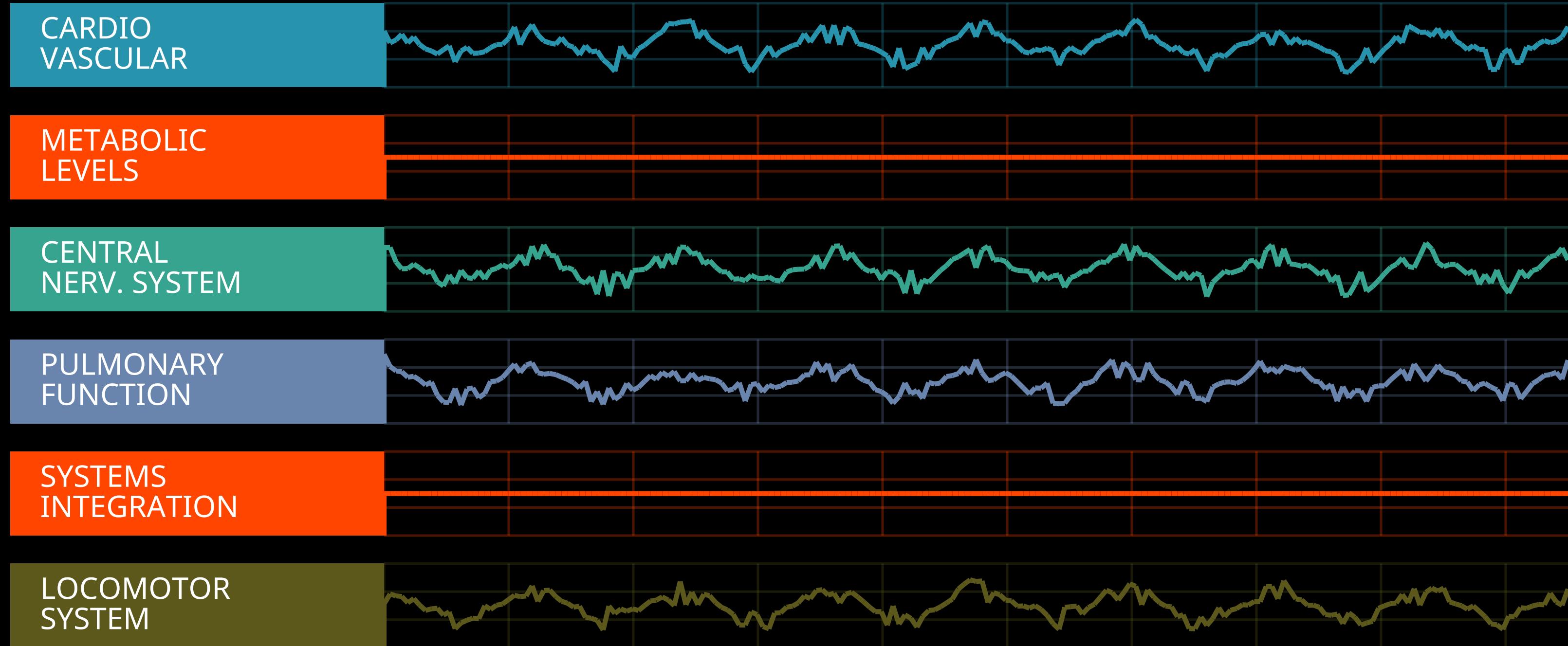


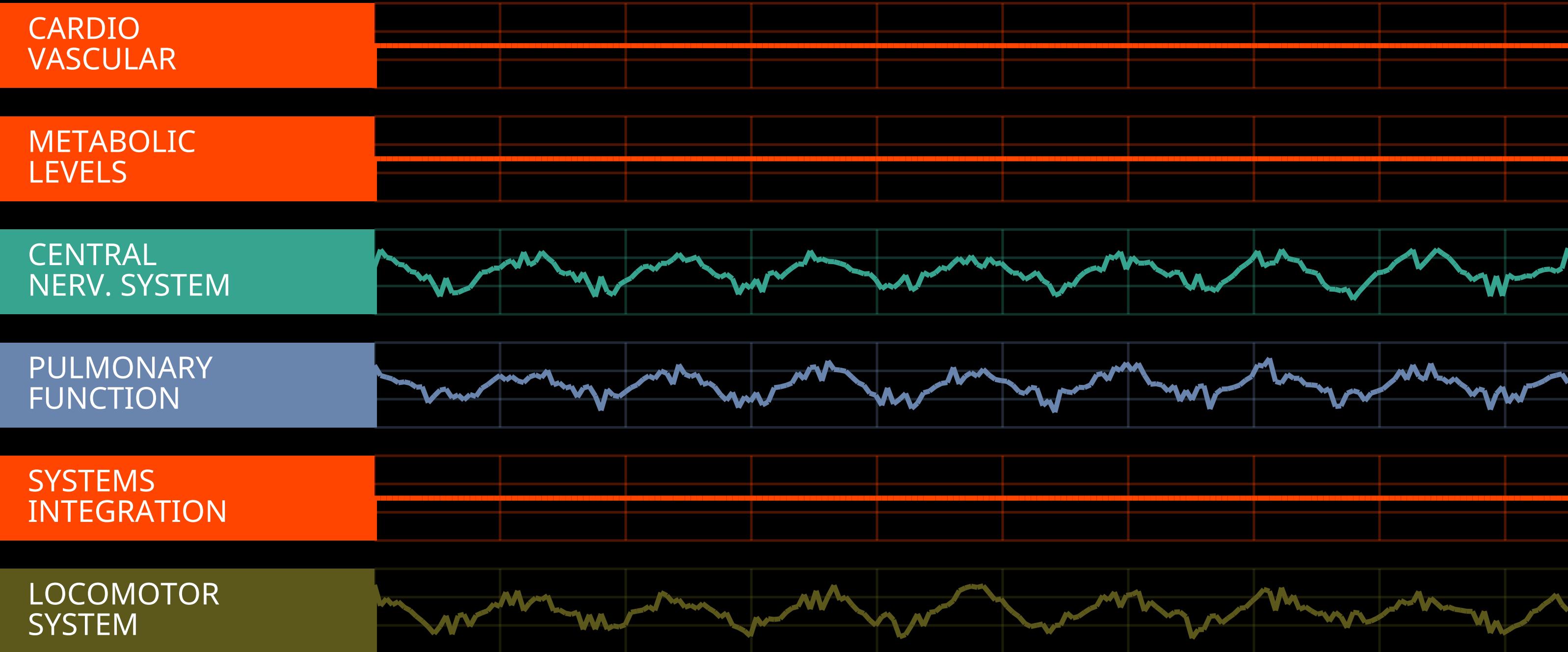
LOCOMOTOR  
SYSTEM



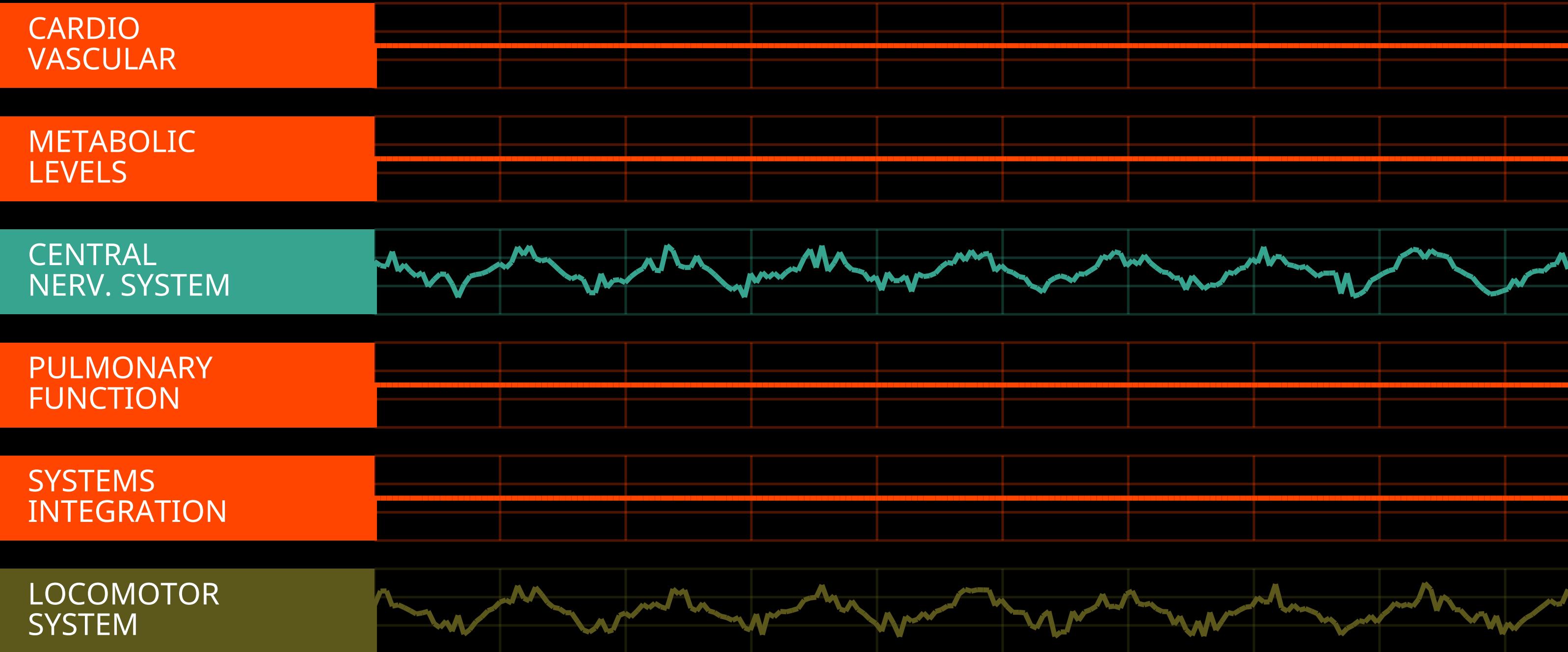
COMPUTER  
MALFUNCTION







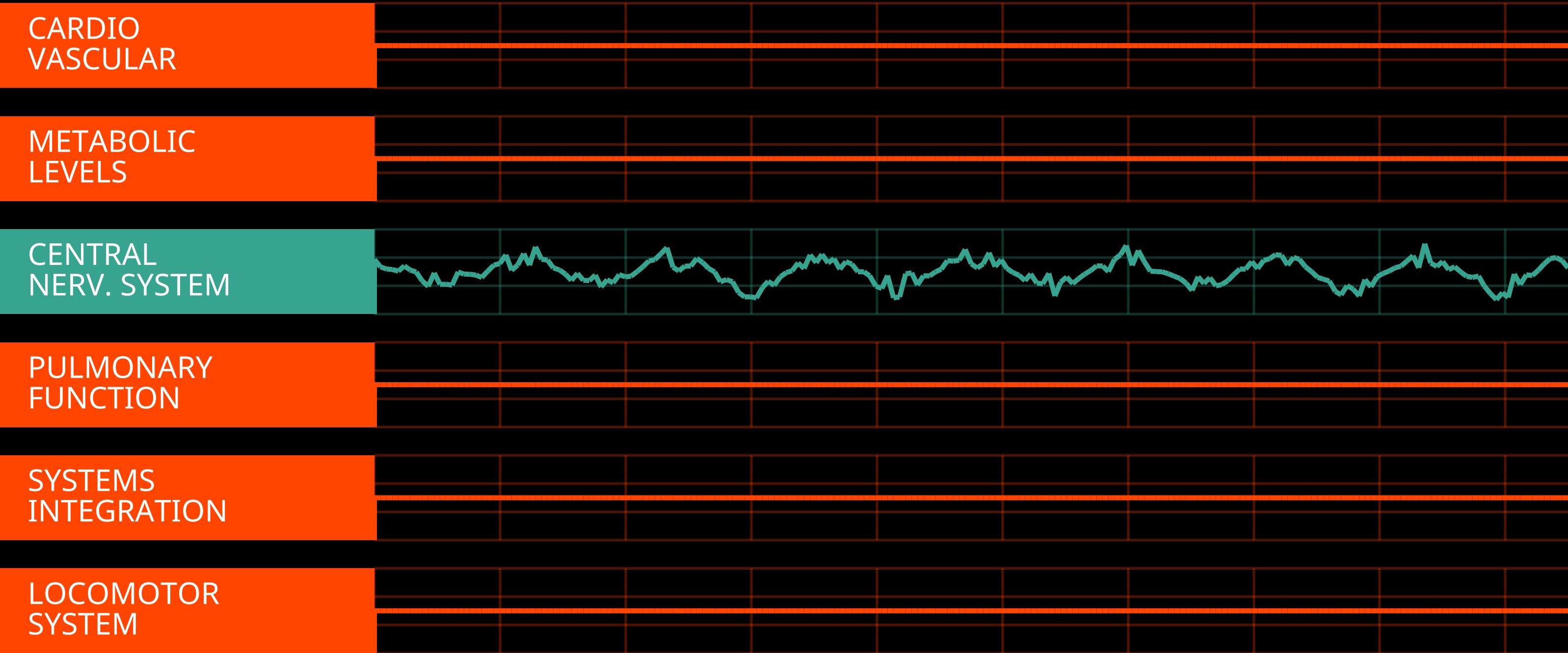
COMPUTER  
MALFUNCTION



LIFE FUNCTIONS  
CRITICAL

LIFE FUNCTIONS  
CRITICAL

LIFE FUNCTIONS  
CRITICAL



# CARDIO VASCULAR



# METABOLIC LEVELS



# CENTRAL NERV. SYSTEM



# PULMONARY FUNCTION



# SYSTEMS INTEGRATION



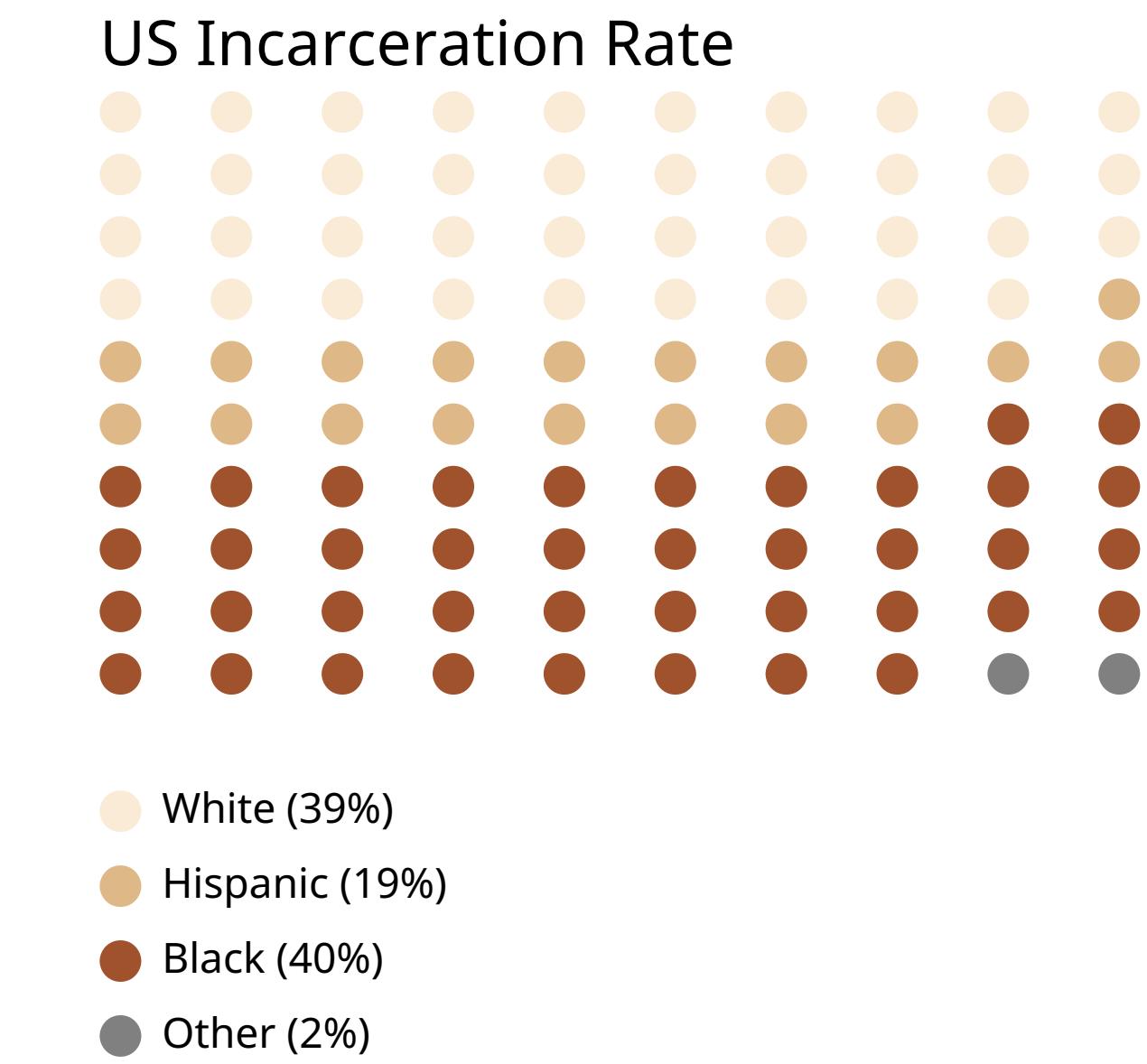
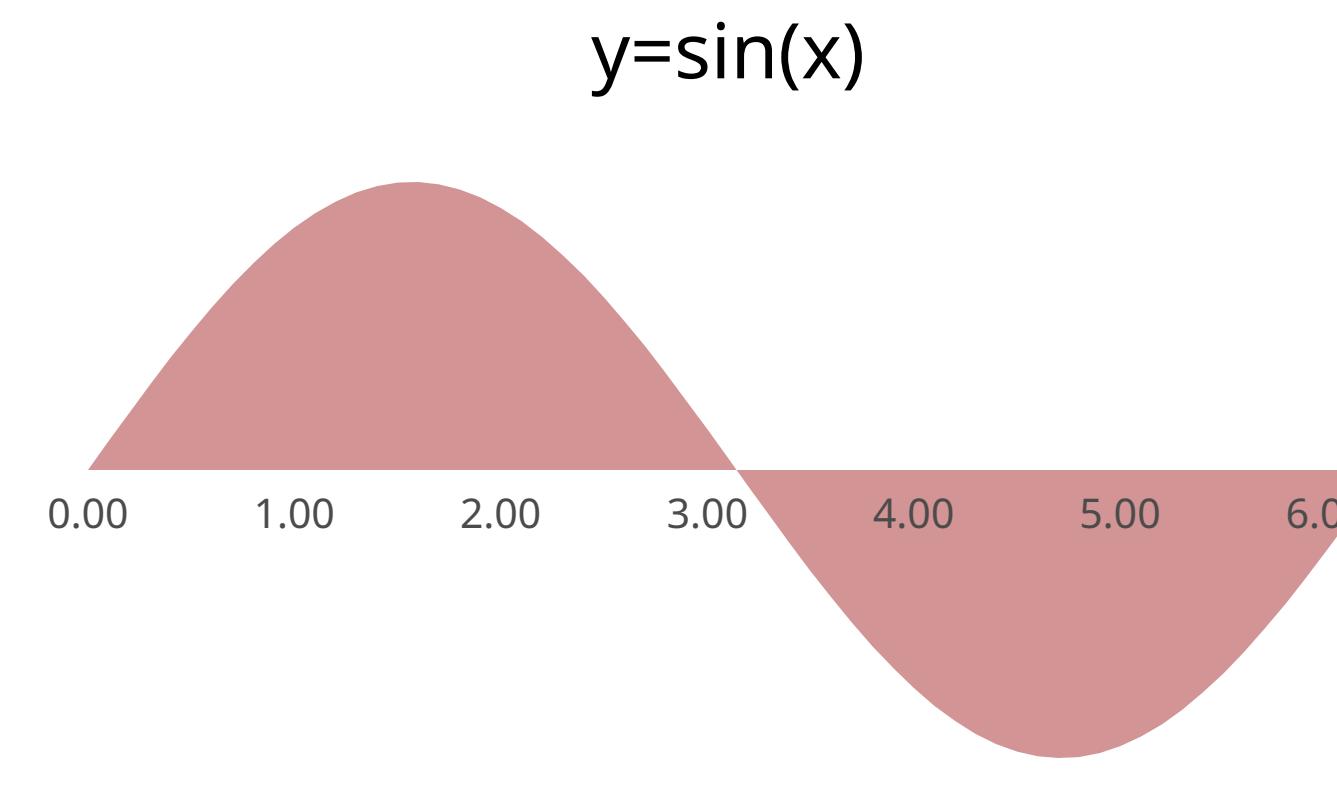
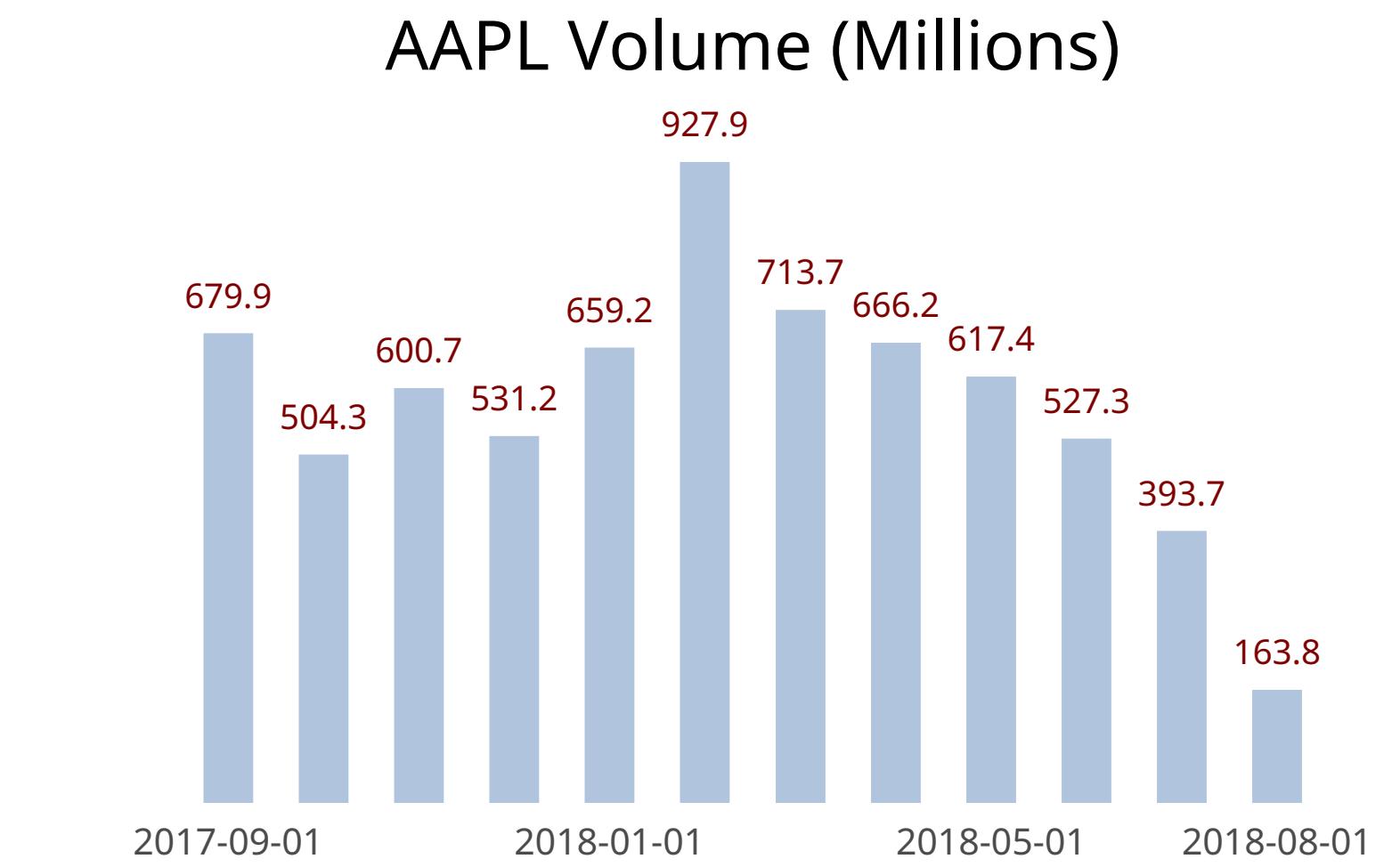
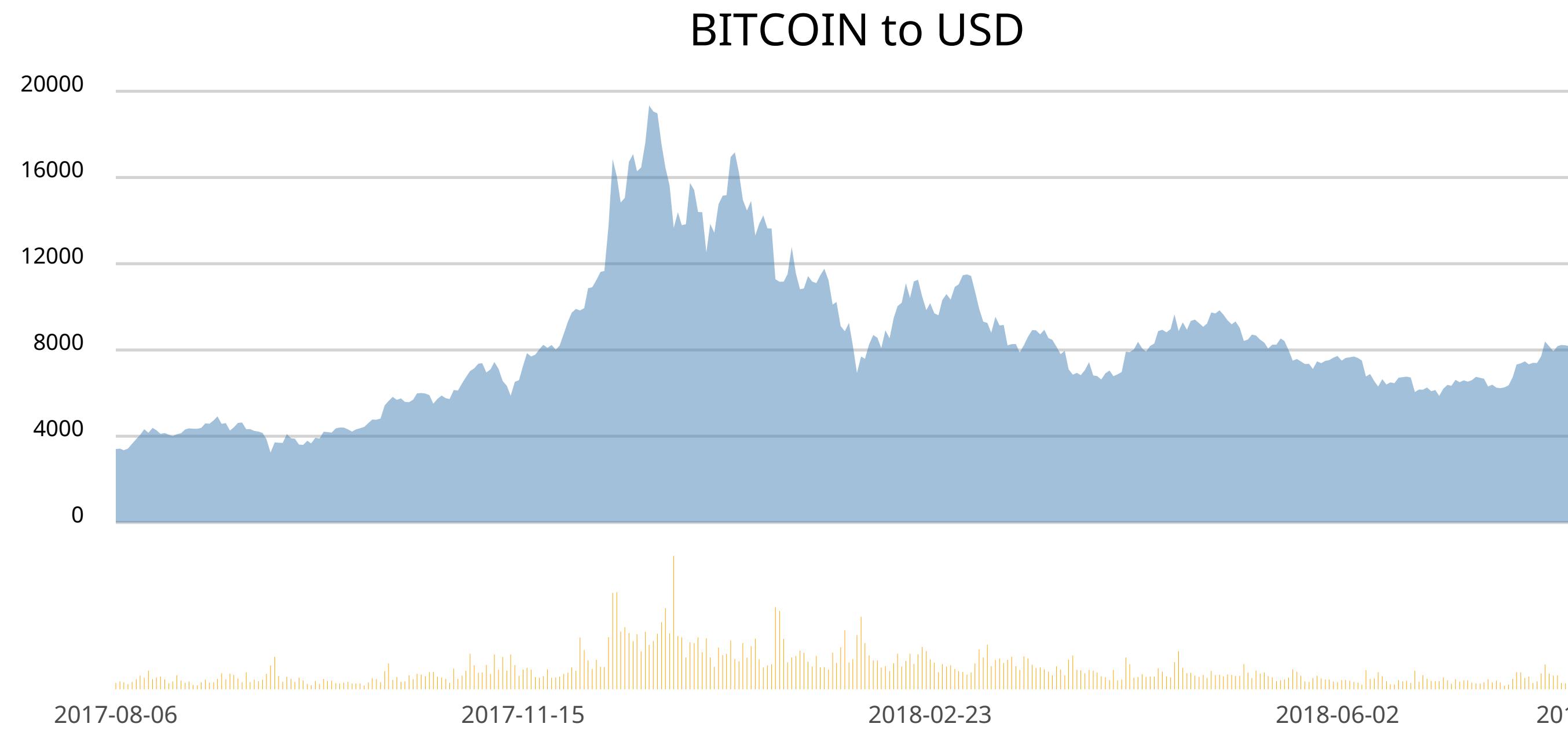
# LOCOMOTOR SYSTEM

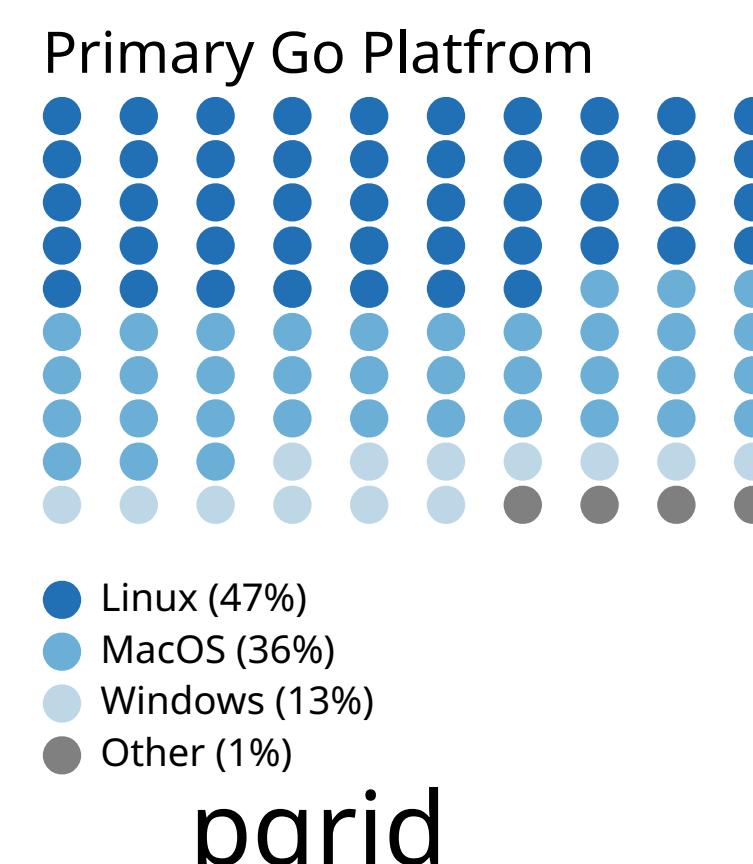
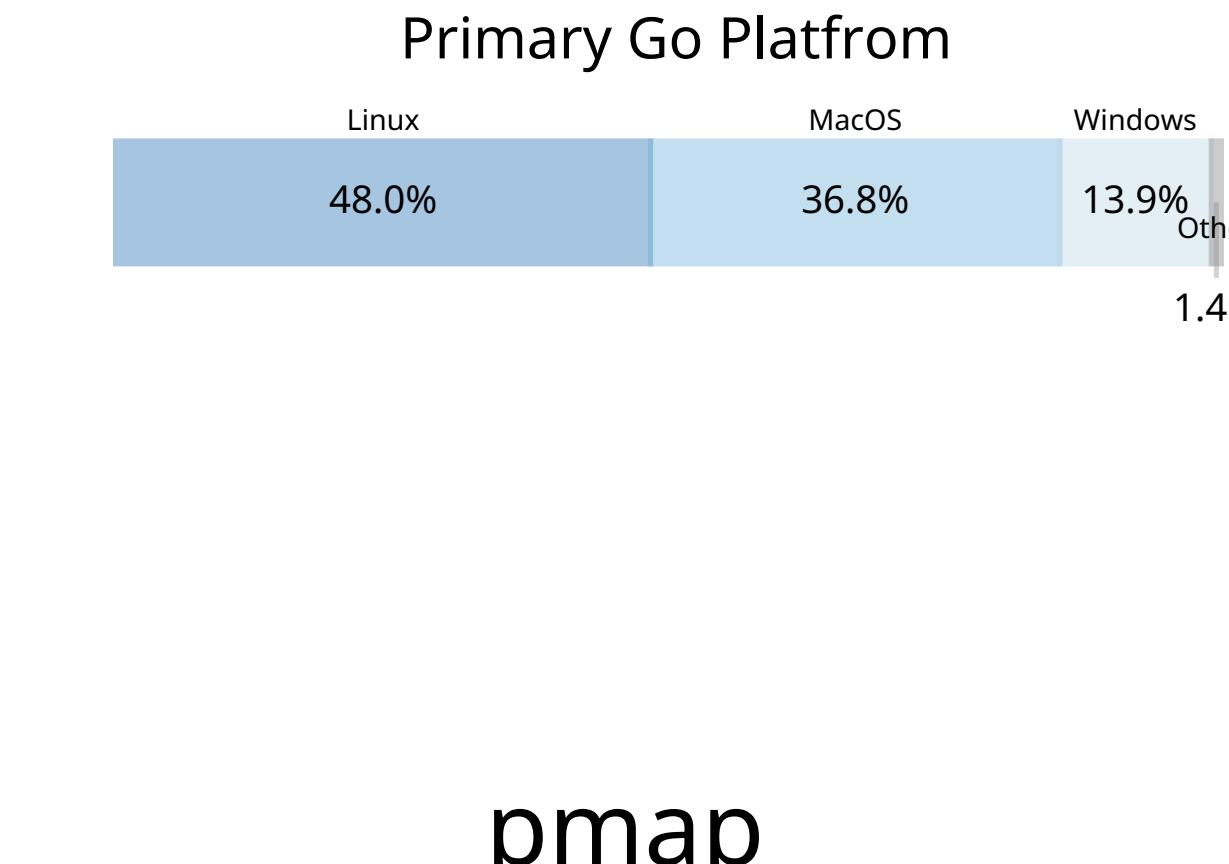
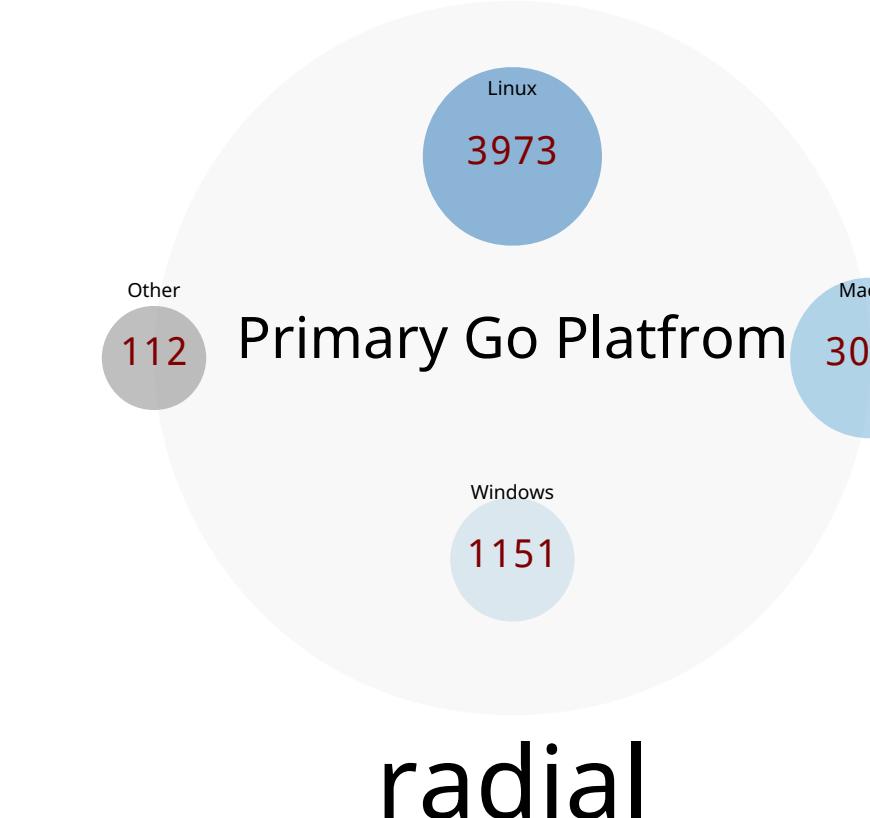
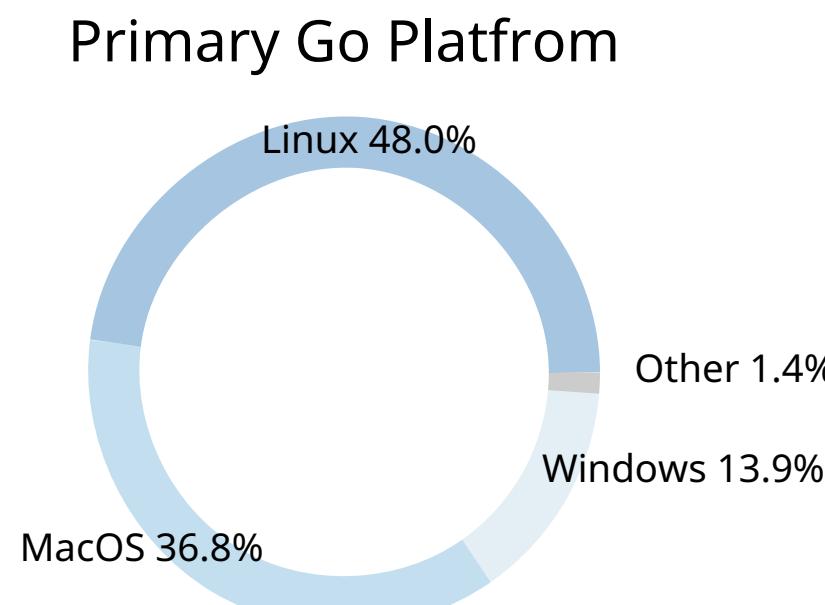
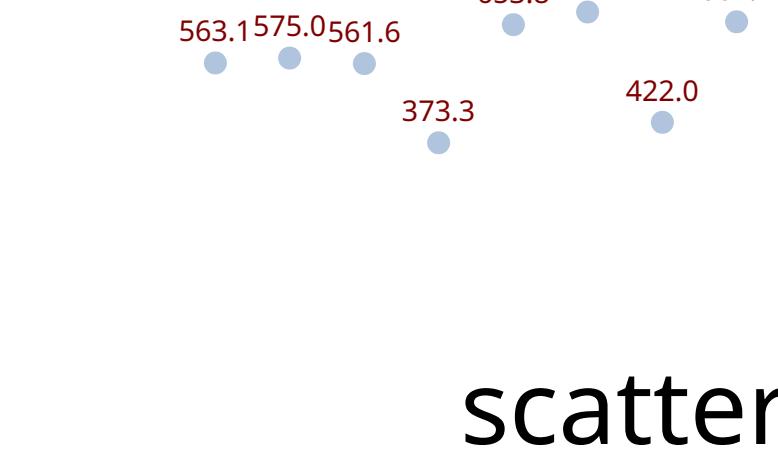
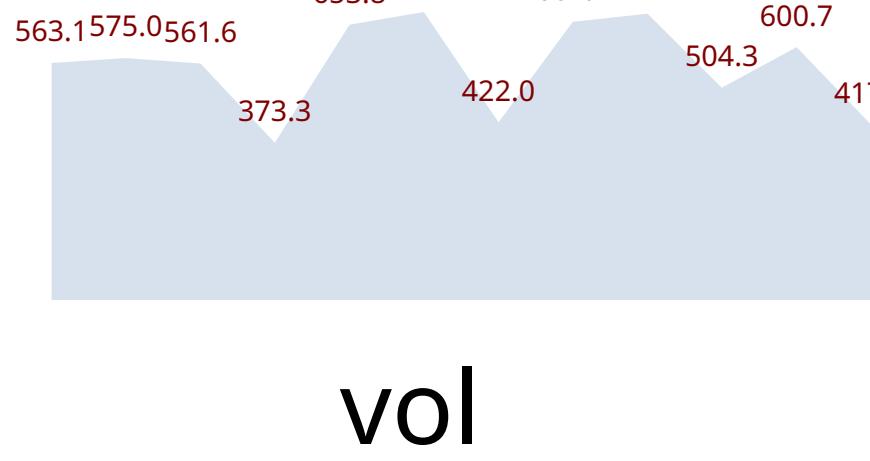
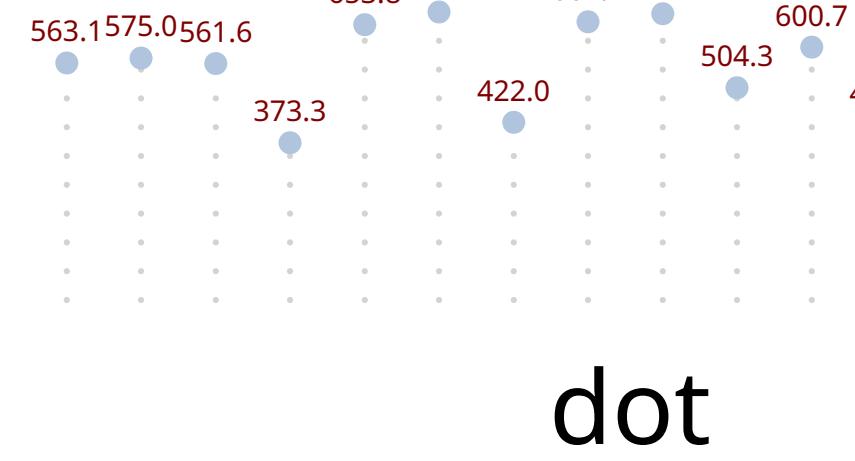
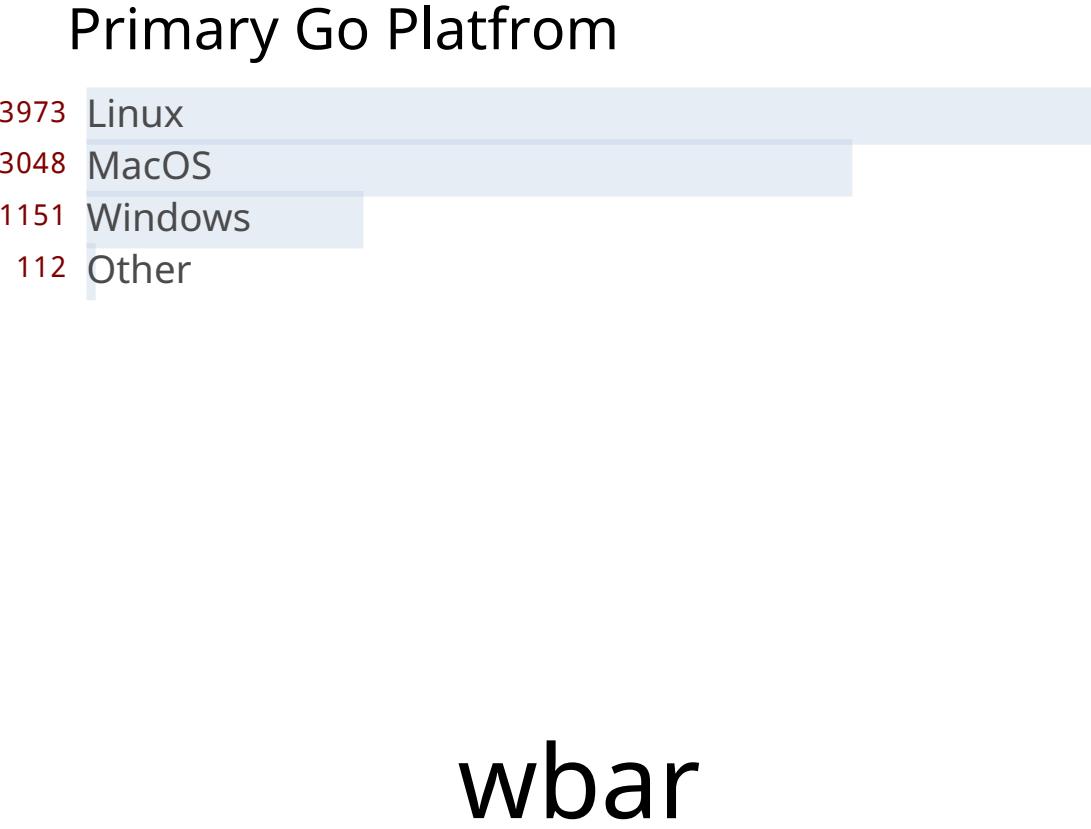
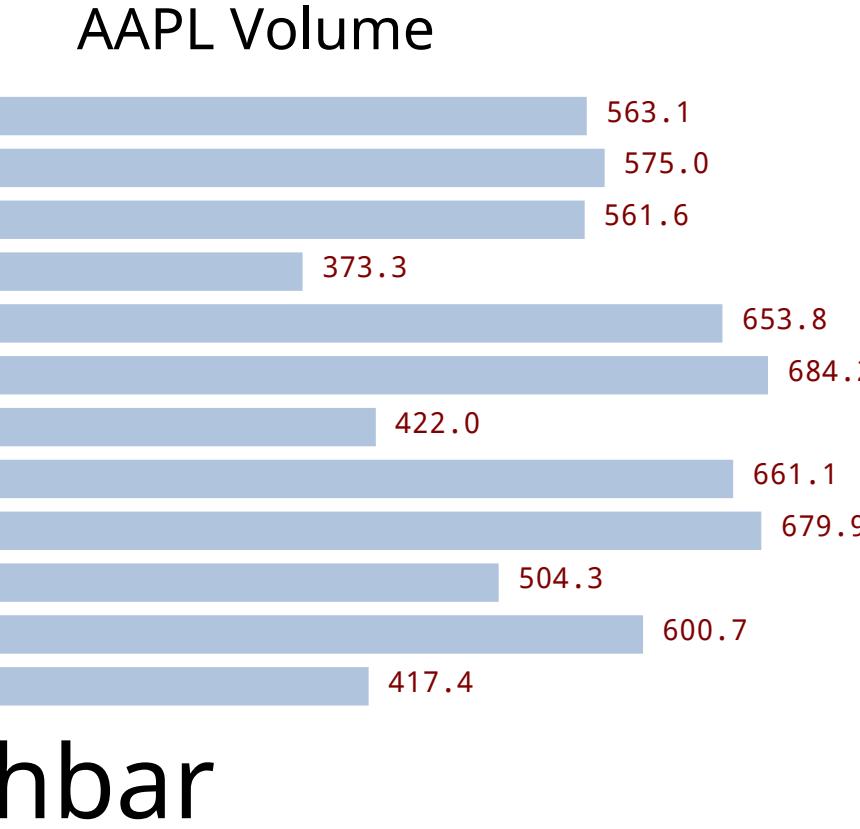
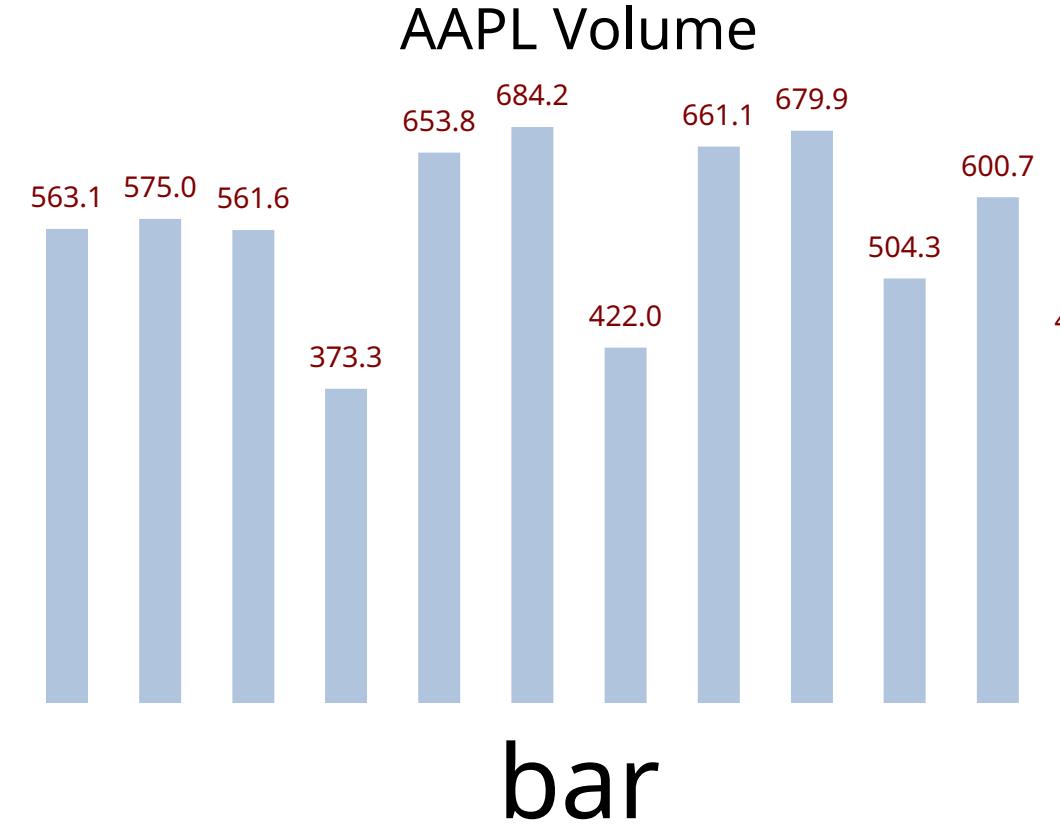
LIFE FUNCTIONS  
TERMINATED

LIFE FUNCTIONS  
TERMINATED

LIFE FUNCTIONS  
TERMINATED

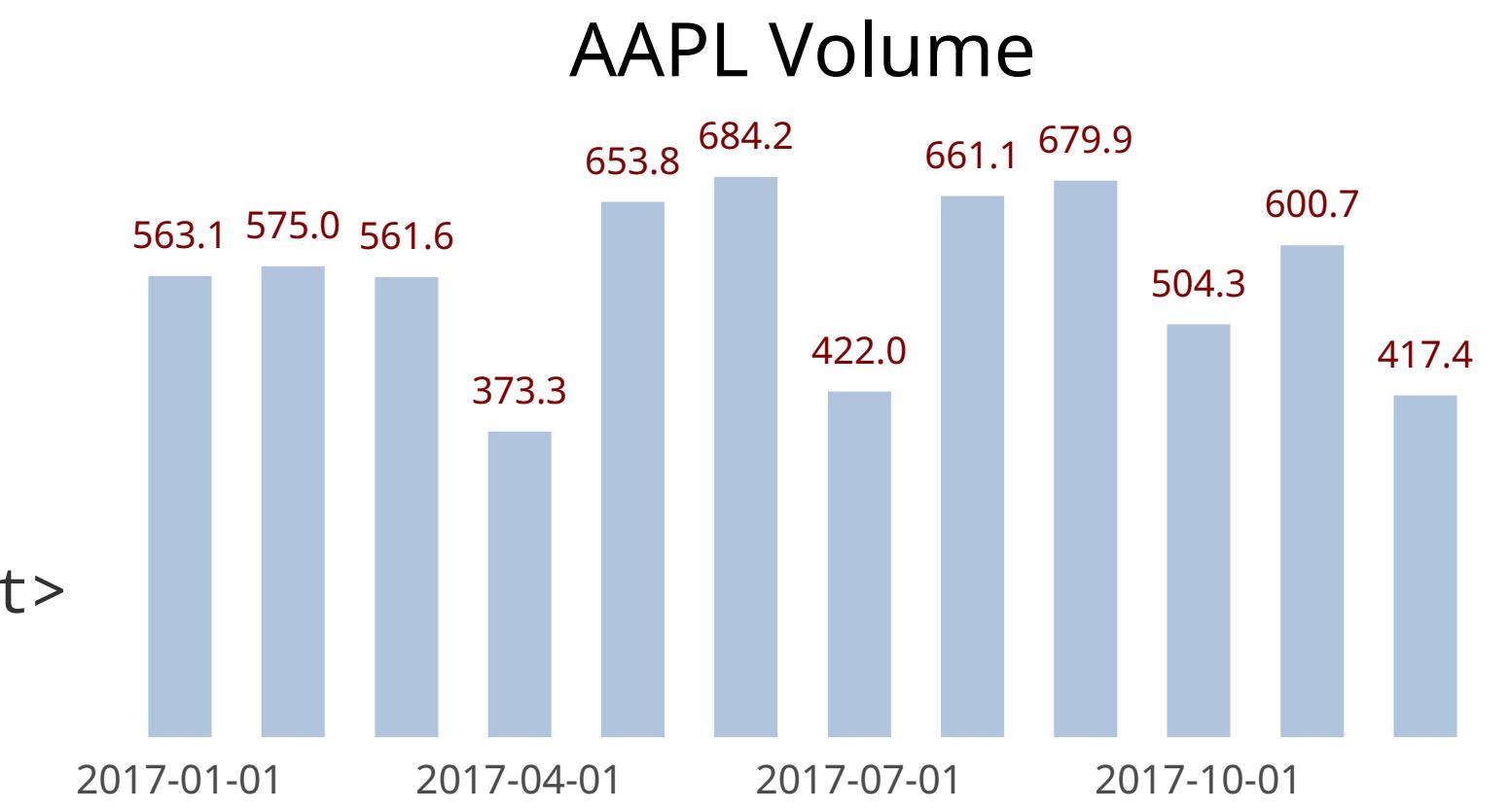
# dchart: charts for deck





```
# AAPL Volume
2017-09-01    679.879
2017-10-01    504.291
2017-11-01    600.663
2017-12-01    531.184
2018-01-01    659.181
2018-02-01    927.894
2018-03-01    713.728
2018-04-01    666.154
2018-05-01    617.408
2018-06-01    527.298
2018-07-01    393.691
2018-08-01    163.768
```

```
<deck>
  <slide>
    <text ...>AAPL Volume</text>
    <line ... color="lightsteelblue"/>
    <text ... color="rgb(127,0,0)">563.1</text>
    <text ... color="rgb(75,75,75)">2017-01-01</text>
  </slide>
</deck>
```



data | dchart | pdf

# Chart Types

-bar	bar chart (default true)
-wbar	word bar chart (default false)
-hbar	horizontal bar chart (default false)
-scatter	scatter chart (default false)
-dot	dot plot (default false)
-line	line chart (default false)
-vol	volume plot (default false)
-pgrid	proportional grid (default false)
-pmap	proportional map (default false)
-donut	donut chart (default false)
-radial	radial chart (default false)

# Chart Elements

-grid	show gridlines on the y axis (default false)
-val	show values (default true)
-valpos	value position (t=top, b=bottom, m=middle) (default "t")
-pct	show computed percentage with the values (default false)
-yaxis	show a y axis (default true)
-yrange	specify the y axis labels (min,max,step)
-fulldeck	generate full deck markup (default true)
-title	show the title (default true)
-charttitle	specify the title (overriding title in the data)
-xlabel	x axis label interval (default 1, 0 to suppress all labels)
-xlast	show the last x label
-hline	horizontal line at value with label

# Position and Scaling

-top	top of the plot (default 80)
-bottom	bottom of the plot (default 30)
-left	left margin (default 20)
-right	right margin (default 80)
-min	set the minimum value
-max	set the maximum value
-dmin	data minimum (default false, min=0)

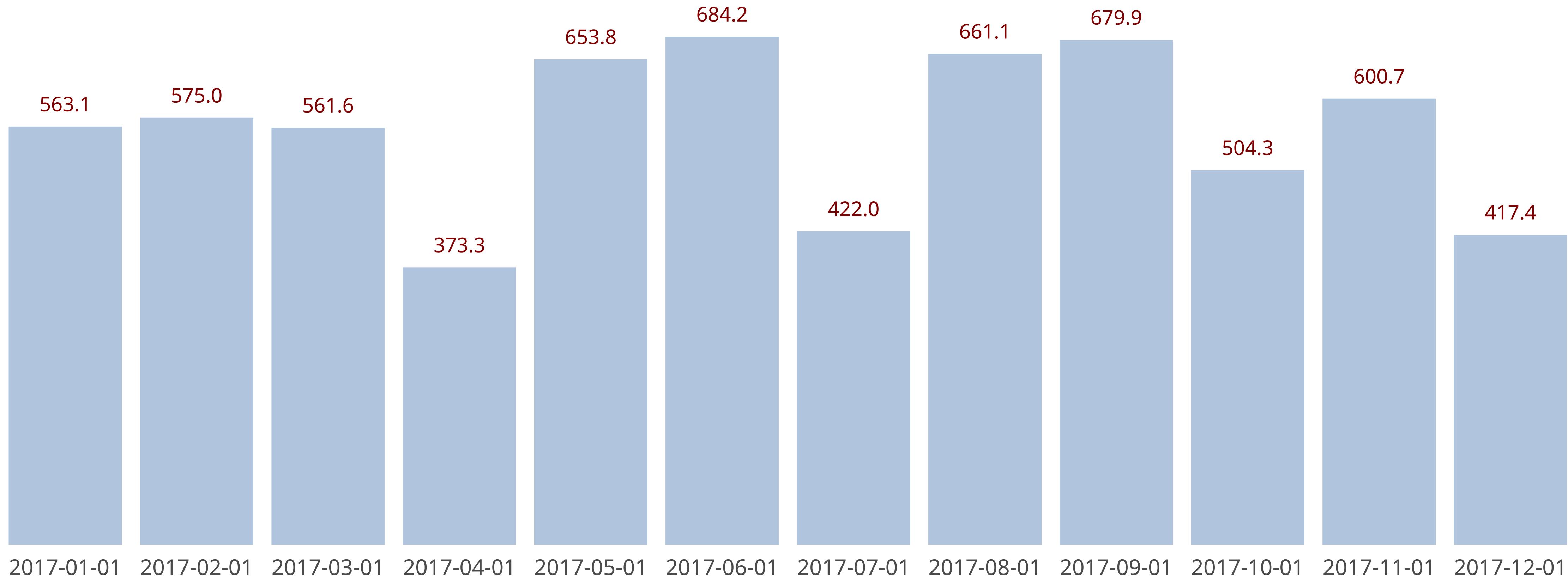
# Measures and Attributes

-barwidth	barwidth (default computed from data size)
-linewidth	linewidth (default 0.2)
-ls	linespacing (default 2.4)
-textsize	text size (default 1.5)
-color	data color (default "lightsteelblue")
-vcolor	value color (default "rgb(127,0,0)")
-volop	volume opacity (default 50)
-datafmt	data format for values (default "%.1f")
-psize	diameter of the donut (default 30)
-pwidth	width of the donut or proportional map (default 3)
-spokes	show spokes on radial charts (default false)

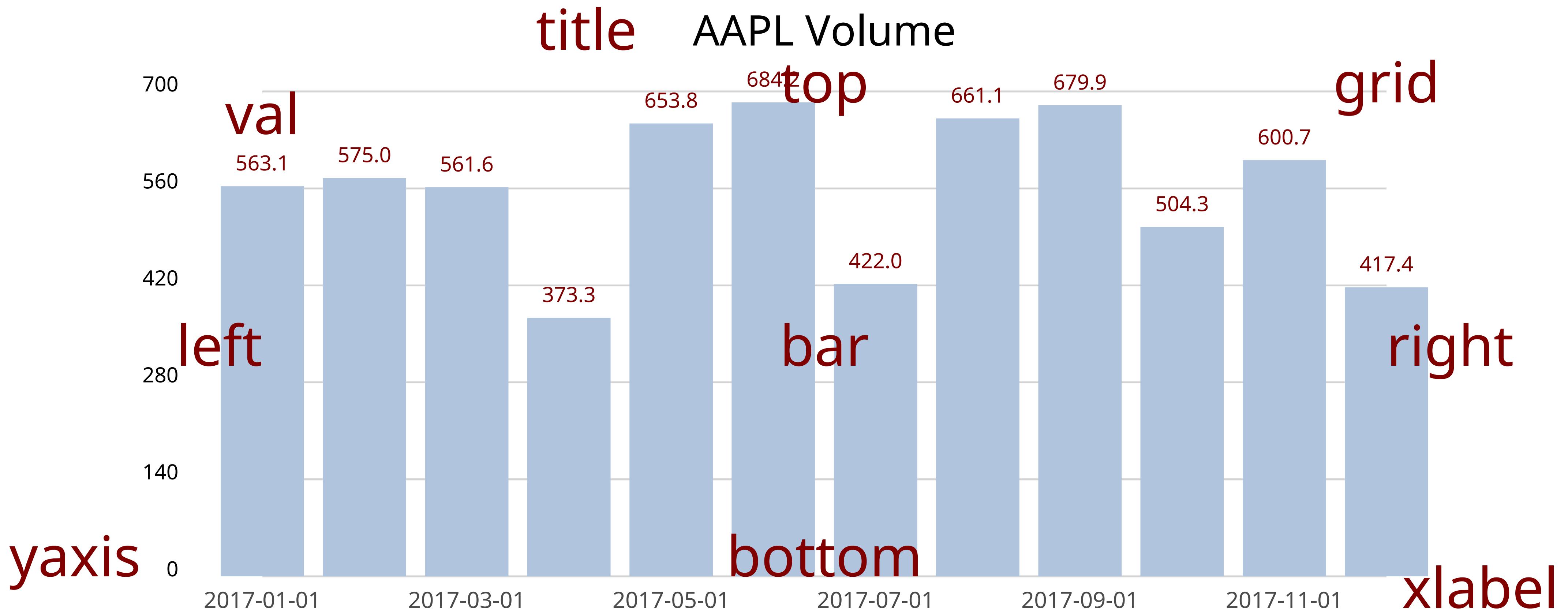
# CSV

-csv	read CSV files (default false)
-csvcol	specify the columns to use for label,value

## AAPL Volume



dchart AAPL.d



dchart -left=20 -right=80 -top=75 -yaxis -xlabel=2 -grid AAPL.d

```

package main

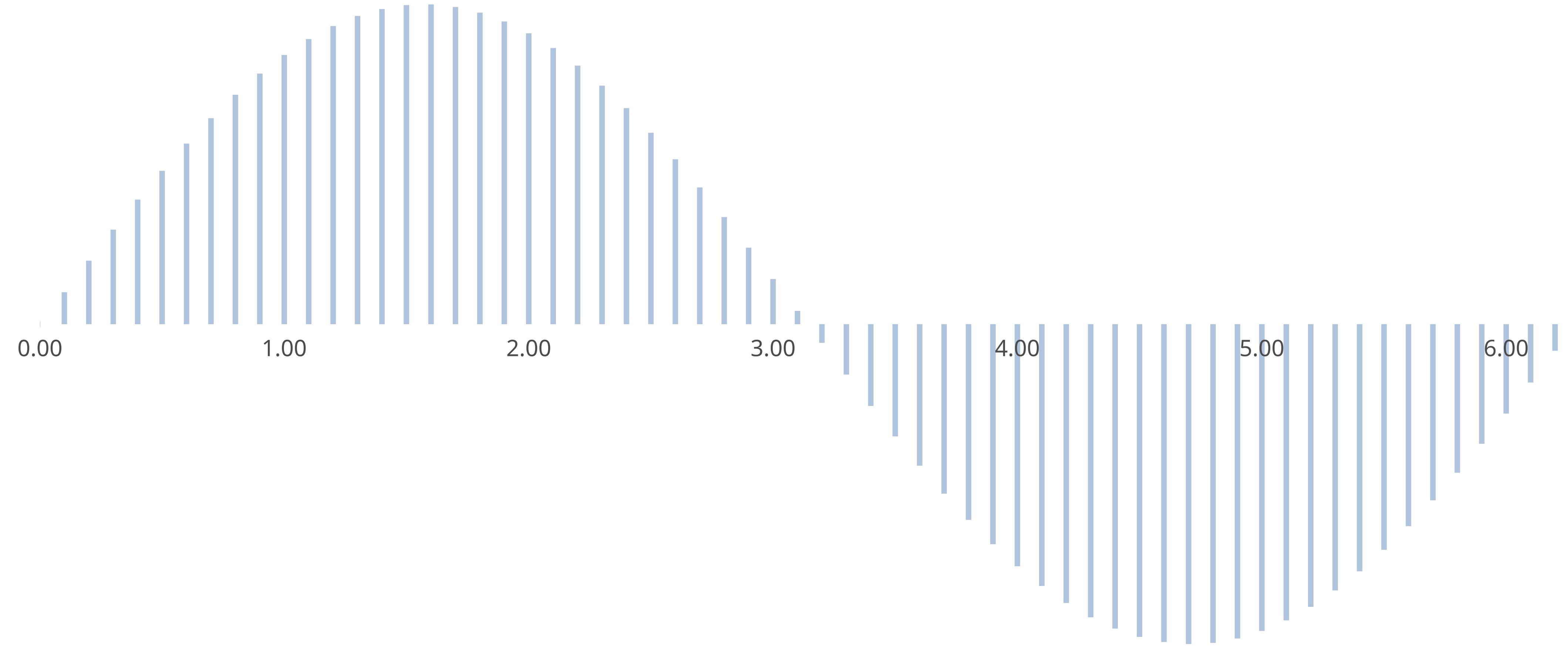
import (
    "fmt"
    "math"
)

func main() {
    fmt.Println("# y=sin(x)")
    for x := 0.0; x < math.Pi*2; x += 0.1 {
        fmt.Printf("%.2f\t%.4f\n", x, math.Sin(x))
    }
}

```

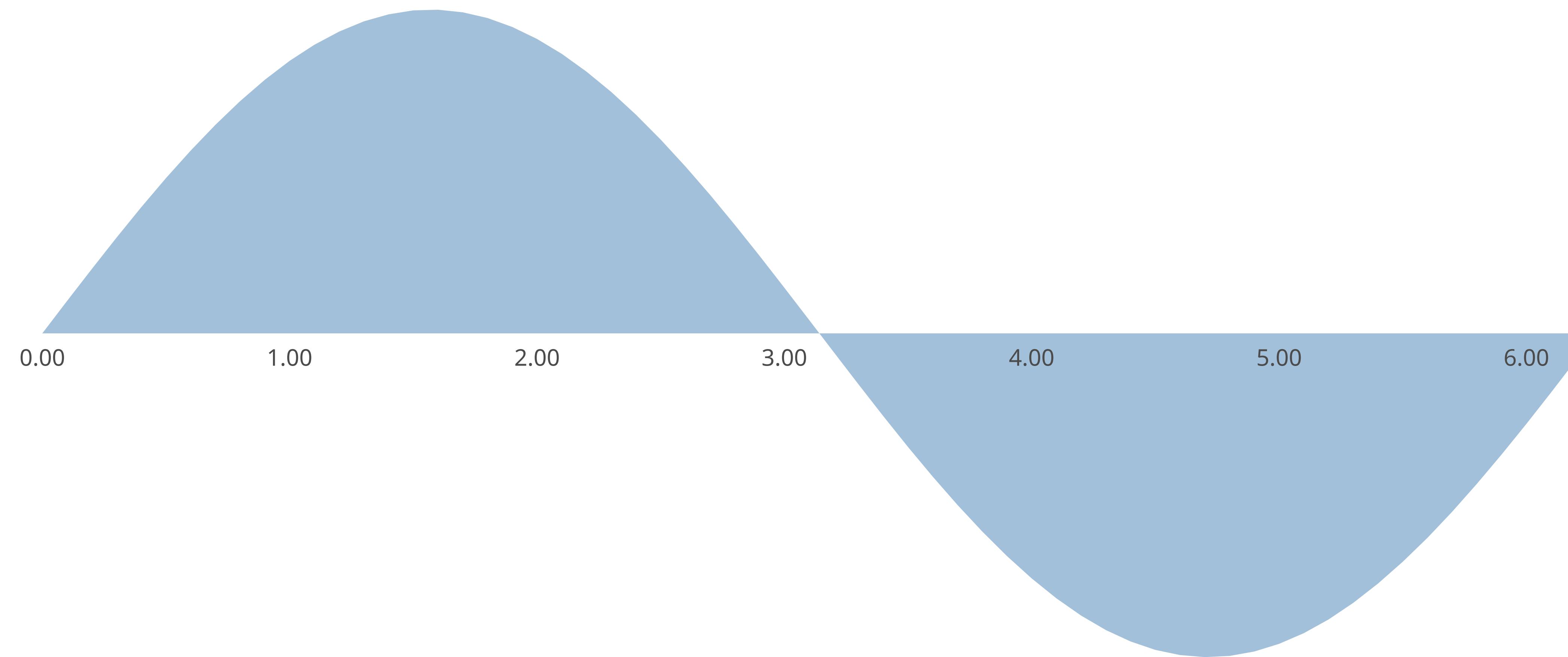
#	y=sin(x)
0.00	0.0000
0.10	0.0998
0.20	0.1987
0.30	0.2955
0.40	0.3894
0.50	0.4794
0.60	0.5646
0.70	0.6442
0.80	0.7174
...	
5.80	-0.4646
5.90	-0.3739
6.00	-0.2794
6.10	-0.1822
6.20	-0.0831

$y=\sin(x)$

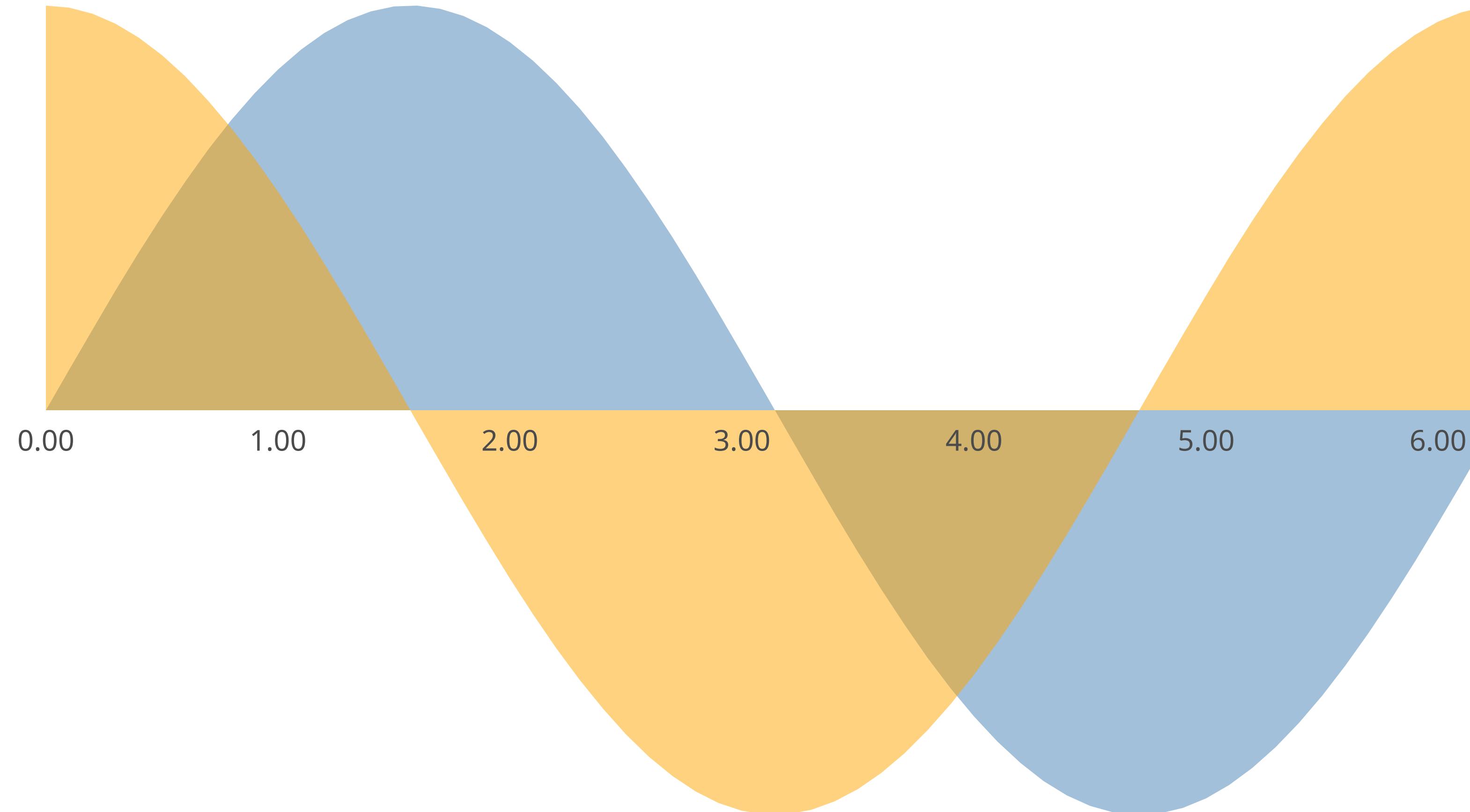


sine | dchart -val=f -bottom=50 -xlabel=10

$y=\sin(x)$



sine | dchart -val=f -bottom=50 -xlabel=10 -bar=f -color=steelblue -vol



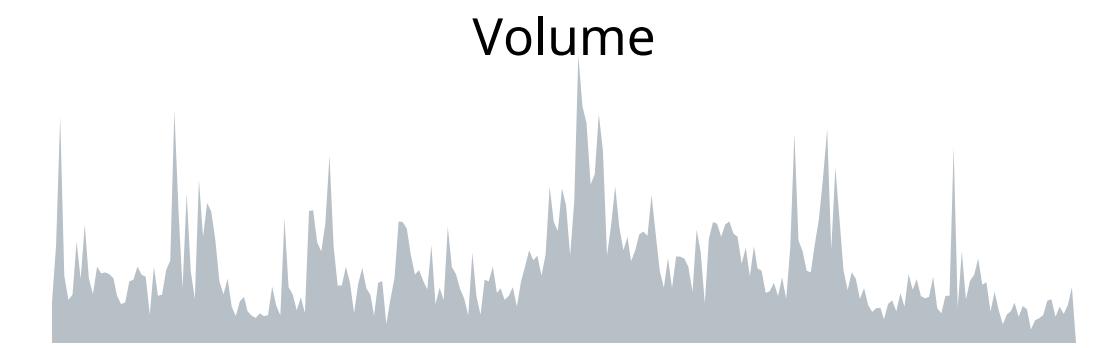
```
#!/bin/sh
mopts="-fulldeck=f -bar=f -vol -top=90 -bottom 60 -left=20 -right=80 -val=f -title=f"
mfunc -f sine | dchart $mopts -color=steelblue -xlabel=10
mfunc -f cosine | dchart $mopts -xlabel=0 -color=orange
```

# Tech Stock Performance

July 2017 - July 2018

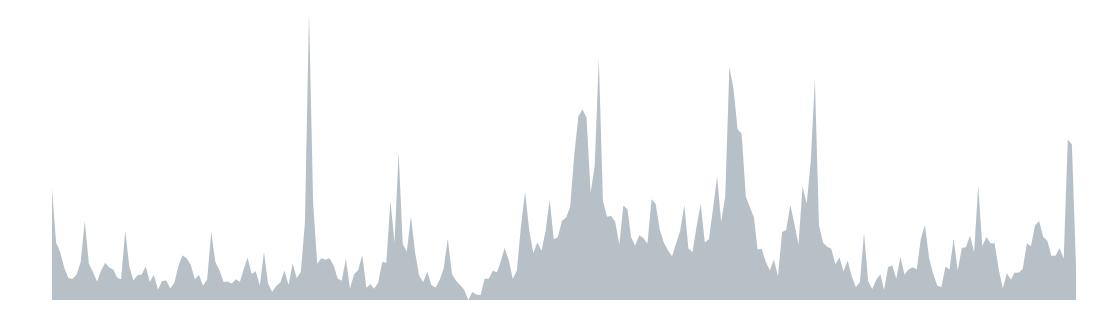
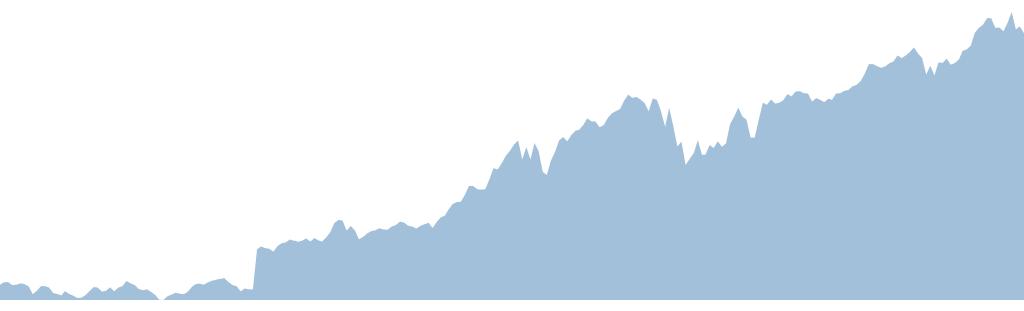
Apple

AAPL



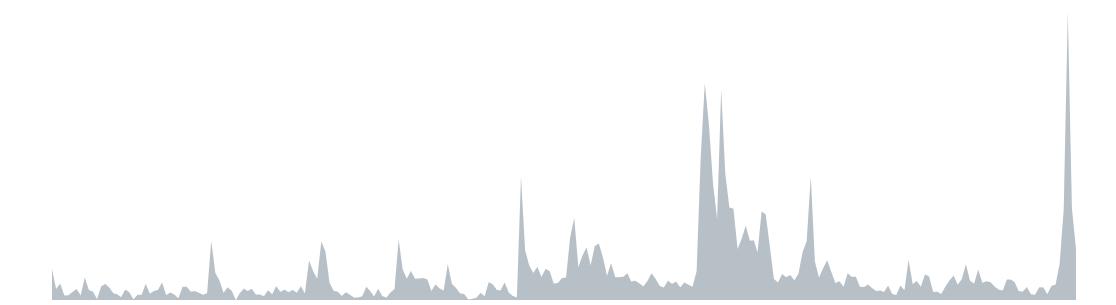
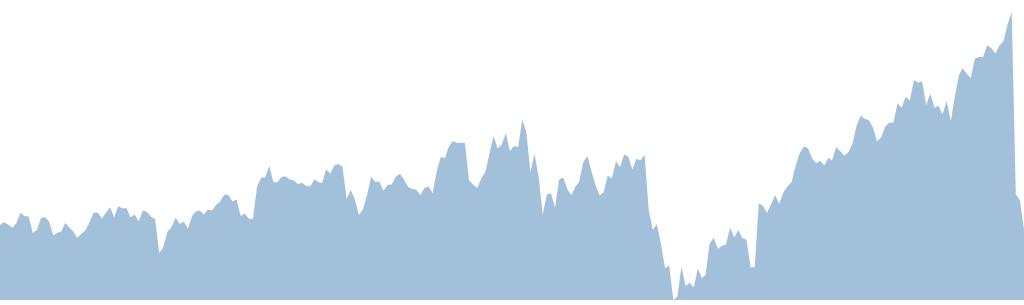
Amazon.com

AMZN



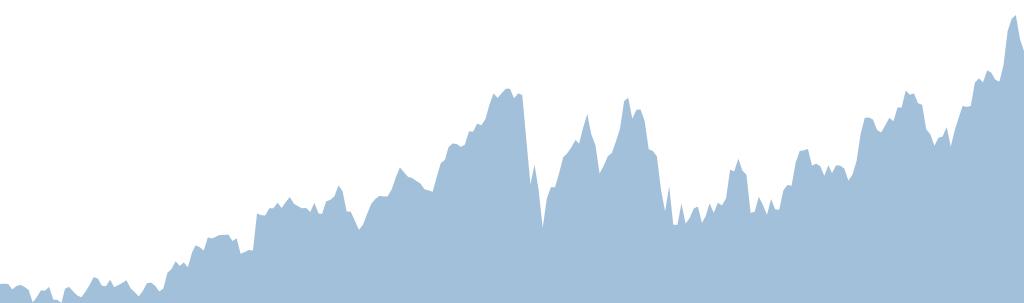
Facebook

FB



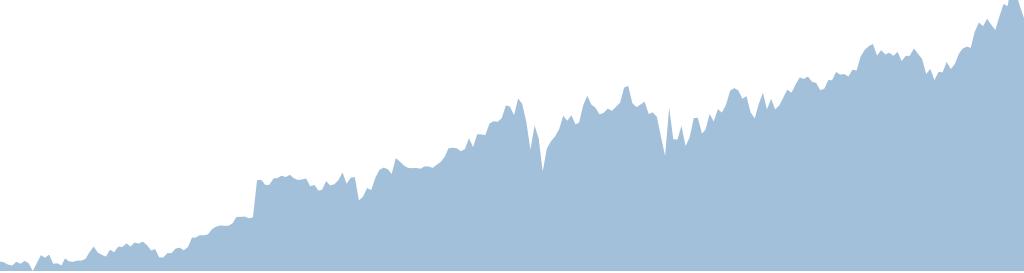
Alphabet

GOOG



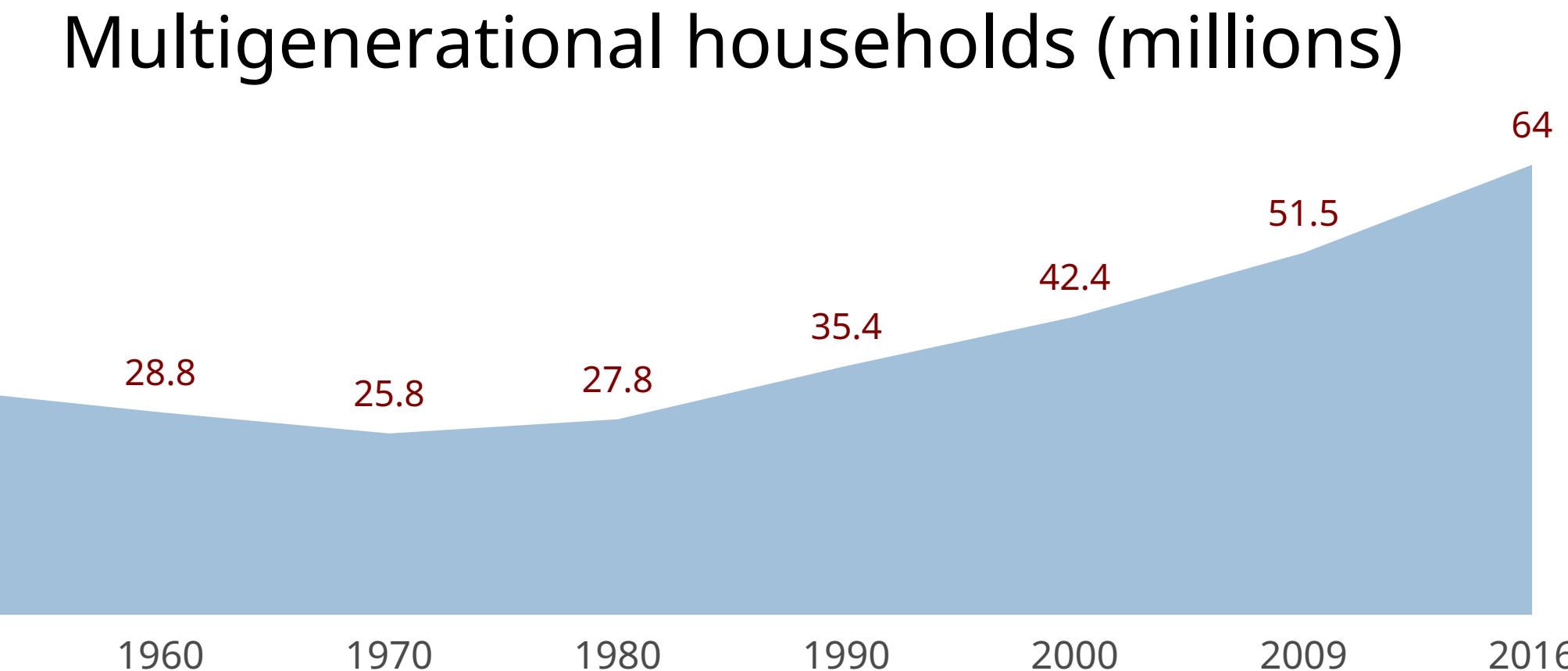
Microsoft

MSFT

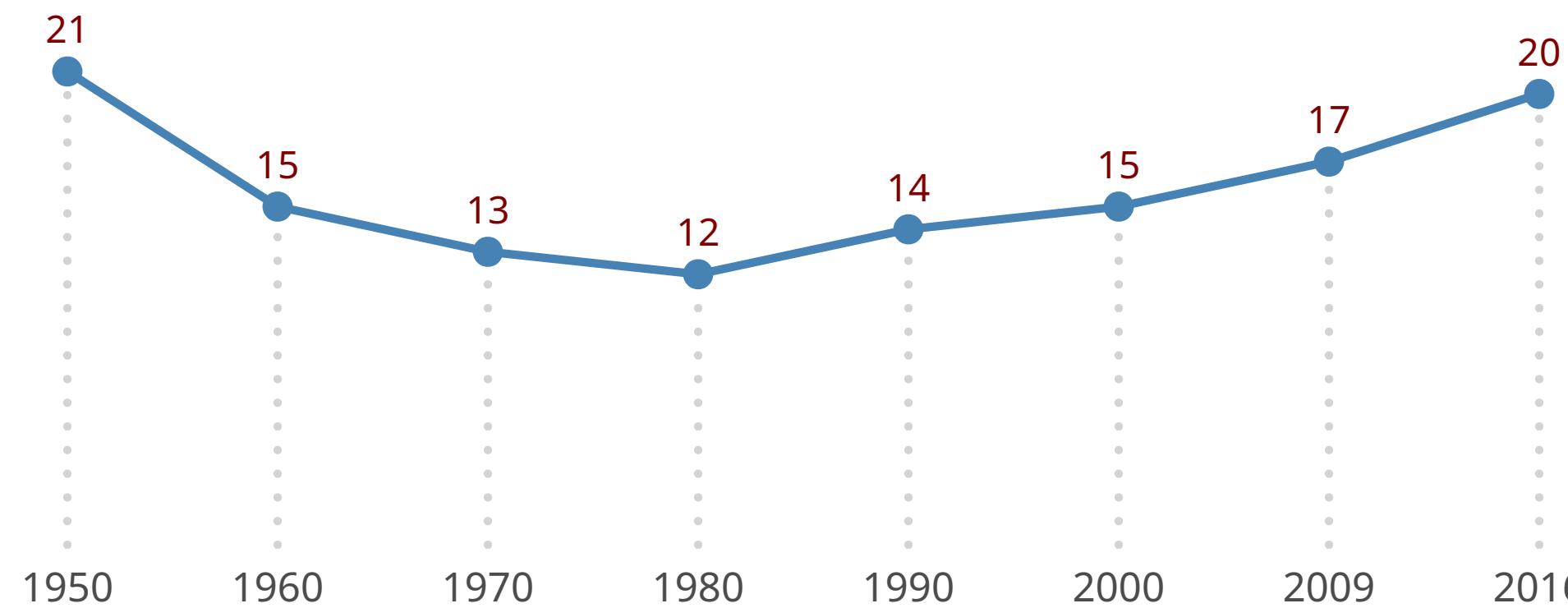


# A record 64 million Americans live in multigenerational households

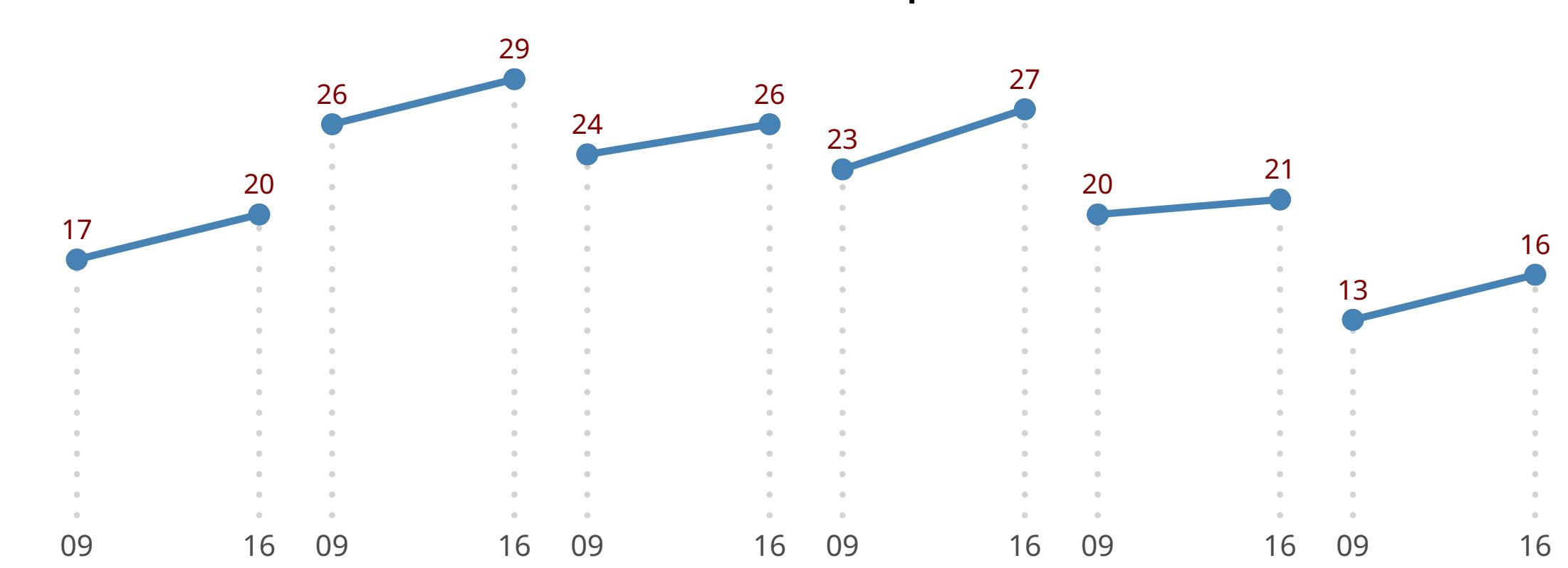
The number and share of Americans living in multi-generational family households have continued to rise, despite improvements in the U.S. economy since the Great Recession. In 2016, a record 64 million people, or 20% of the U.S. population, lived with multiple generations under one roof, according to a new Pew Research Center analysis of census data.



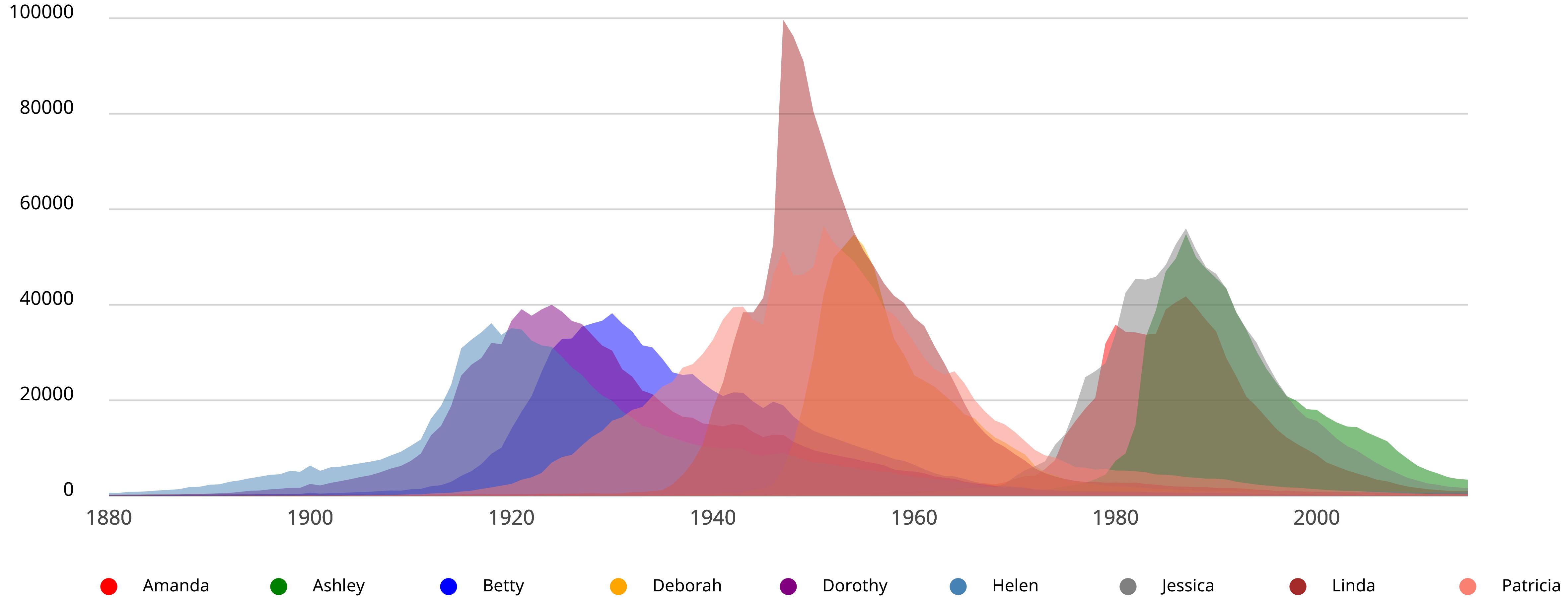
% of Americans in multigenerational households



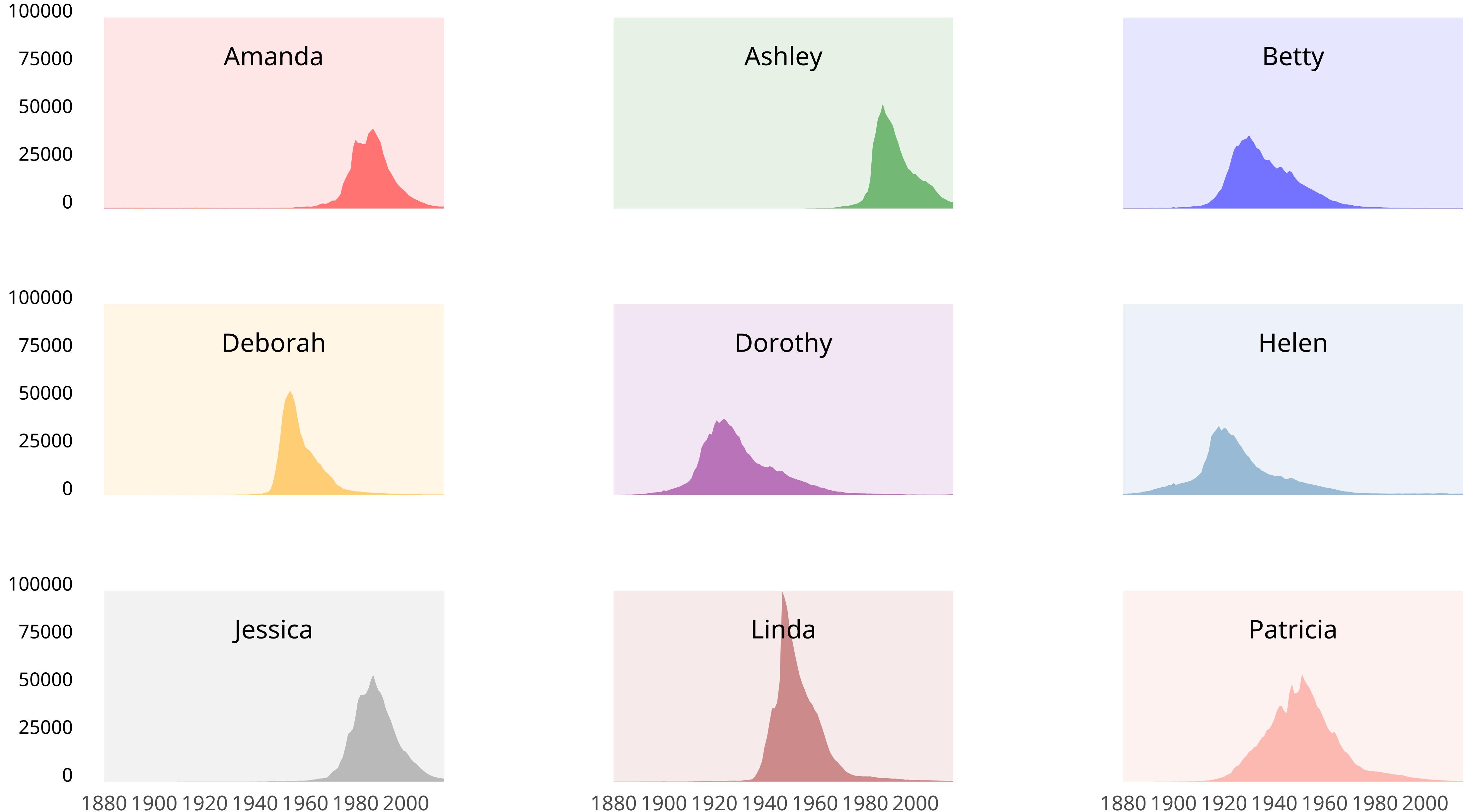
Total Asian Black Hispanic Other White



# Evolution of Baby Names in the US: 1880-2015



# Evolution of Baby Names in the US: 1880-2015



# What is it about Go?

fmt

func

io.Writer

io.Reader

net/http

encoding/xml

encoding/json

encoding/csv

cgo

# The Community

From: Russ Cox  
Subject: Re: [go-nuts] Visualizing Random Number Generators...  
Date: March 5, 2010 1:14:44 EST  
To: ajstarks <ajstarks@gmail.com>

---

are you going to share the library  
or just tease us with pictures? ;-)

Thompson wanted to create a comfortable computing environment constructed according to his own design, using whatever means were available.

Dennis M. Ritchie, “The Development of the C Language”

Go is not the product of a Whiggish development process. We were just trying to get something that worked for us.

Rob Pike, "Origin of Go's interface design", golang-nuts

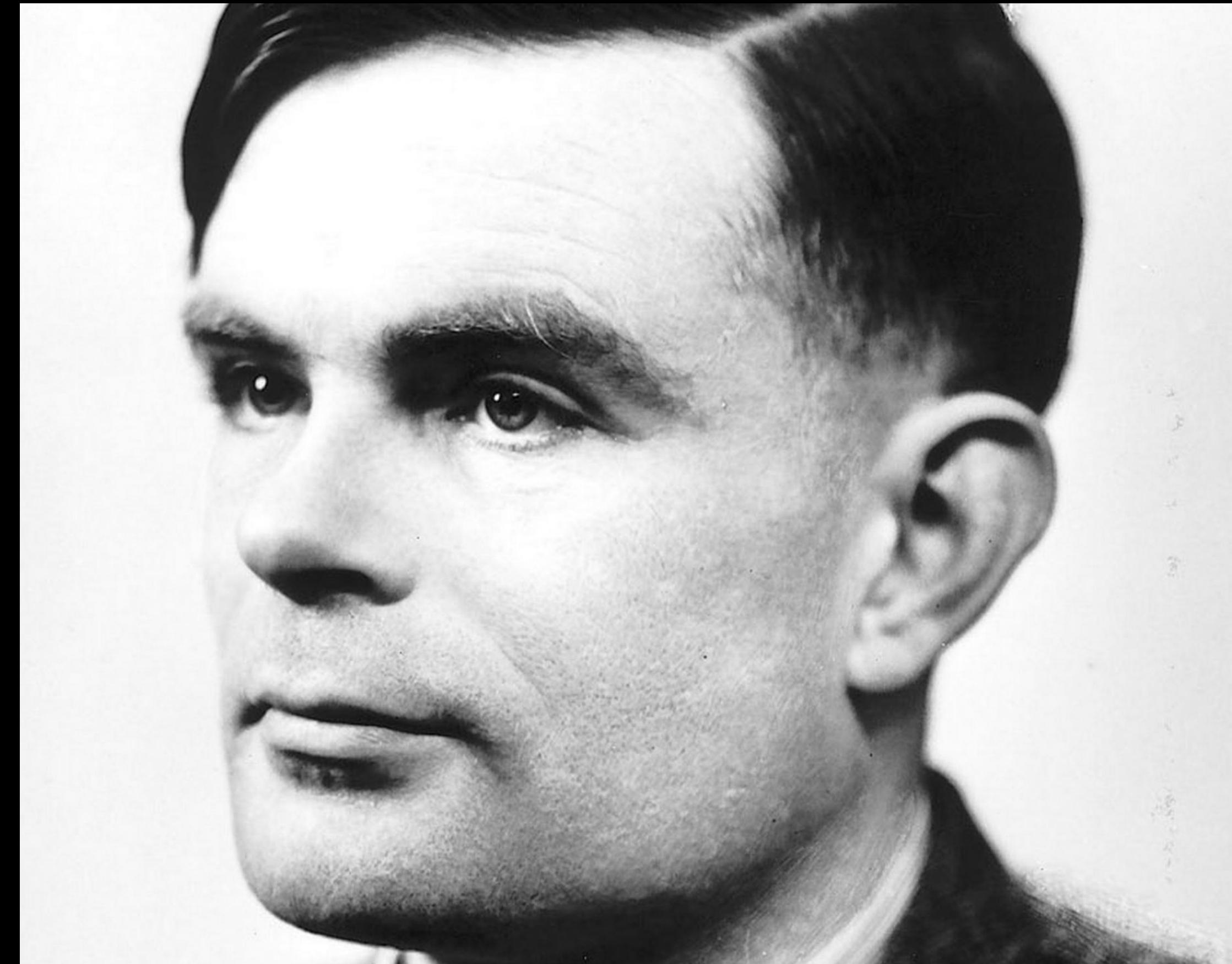
# Making Tools

# Fun

Reducing the distance from  
the idea to the picture



Picasso



Turing

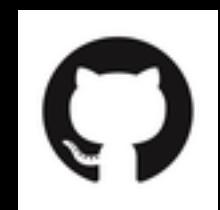
# Thank you



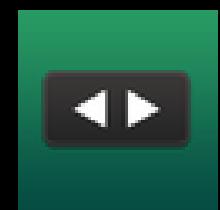
@ajstarks



ajstarks@gmail.com



github.com/ajstarks



speakerdeck.com/ajstarks



flickr.com/photos/ajstarks