

decksh

a little language for decks



Anthony Starks

@ajstarks



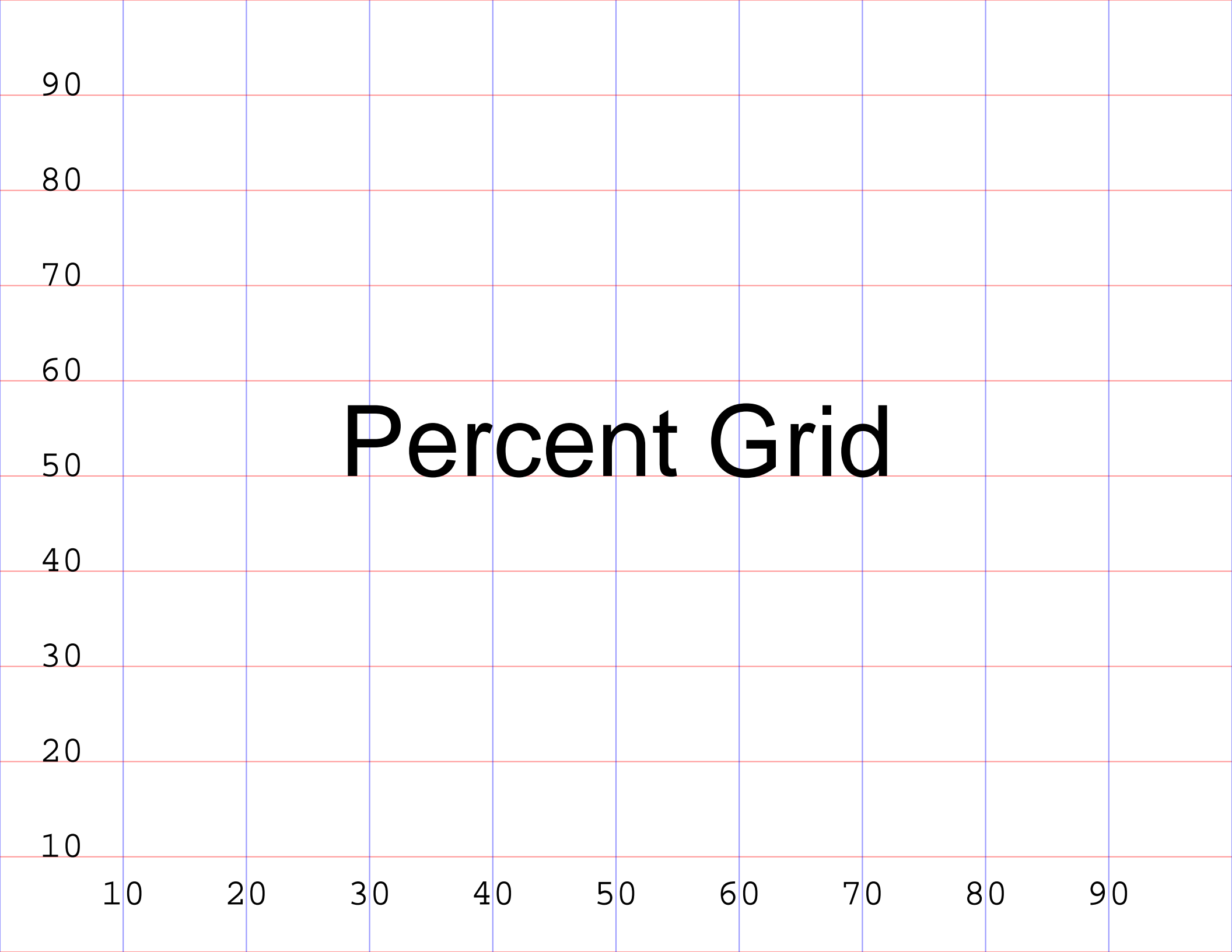
When you say “language,” most programmers think of the big ones, like FORTRAN or COBOL or Pascal. In fact, a language is any mechanism to express intent, and the input to many programs can be viewed profitably as statements in a language. This column is about those “little languages.”

Jon Bentley, ACM Programming Pearls, Little Languages, 1986

Deck



a Go package for presentations



decksh



deck markup



SVG

PDF

PNG

```
deck
  slide "rgb(250,250,250)" "black"
    ctext "Deck elements" 50 90 5
    image "follow.jpg" 70 50 640 480 50
    blist 10 75 3
      li "text, image, list"
      li "rect, ellipse, polygon"
      li "line, arc, curve"
    elist

    gy=10
    rect 15 gy 8 6 "rgb(127,0,0)"
    ellipse 27.5 gy 8 6 "rgb(0,127,0)"
    line 50 gy 60 gy
    curve 80 gy 95 30 90 gy
    arc 70 gy 10 8 0 180 0.1 "rgb(0,0,127)"
    polygon "37 37 45" "13 7 10" "rgb(0,0,127)"

    opts="-fulldeck=f -textsize 1 -xlabel=2 -barwidth 1.5"
    dchart -left 10 -right 42 -top 42 -bottom 25 opts AAPL.d
  eslide
edek
```

```
<deck>
<slide bg="rgb(250,250,250)" fg="black">
<text align="c" xp="50" yp="90" sp="5">Deck elements</text>
<image name="follow.jpg" xp="70" yp="50" width="640" height="480" scale="50" />
<list type="bullet" xp="10" yp="75" sp="3">
<li>text, image, list</li>
<li>rect, ellipse, polygon</li>
<li>line, arc, curve</li>
</list>
<rect xp="15" yp="10" wp="8" hp="6" color="rgb(127,0,0)" />
<ellipse xp="27.5" yp="10" wp="8" hp="6" color="rgb(0,127,0)" />
<line xpl="50" ypl="10" xp2="60" yp2="10" />
<curve xpl="80" ypl="10" xp2="95" yp2="30" xp3="90" yp3="10" />
<arc xp="70" yp="10" wp="10" hp="8" a1="0" a2="180" sp="0.1" color="rgb(0,0,127)" />
<polygon xc="37 37 45" yc="13 7 10" color="rgb(0,0,127)" />
<text xp="26.00" yp="45.60" sp="1.50" align="center" wp="0.00" font="sans" opacity="100.00"
color="black" type="">AAPL Volume</text>
<line xpl="10.00" ypl="25.00" xp2="10.00" yp2="37.46" sp="1.50" opacity="100.00"
color="lightsteelblue" />
<text xp="10.00" yp="38.46" sp="0.75" align="center" wp="0.00" font="sans" opacity="100.00"
color="rgb(127,0,0)" type="">679.9</text>
<text xp="10.00" yp="23.00" sp="0.80" align="center" wp="0.00" font="sans" opacity="100.00"
color="rgb(75,75,75)" type="">2017-09-01</text>
<line xpl="12.91" ypl="25.00" xp2="12.91" yp2="34.24" sp="1.50" opacity="100.00"
color="lightsteelblue" />
<text xp="12.91" yp="35.24" sp="0.75" align="center" wp="0.00" font="sans" opacity="100.00"
color="rgb(127,0,0)" type="">504.3</text>
...
</slide>
</deck>
```

Deck elements

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



```
// hello world
deck
  slide "black" "white"
    ctext "hello, world" 50 25 10
    circle 50 0 100 "blue"
  eslide
edeck
```



hello, world

Running decksh

decksh

read from stdin, write to stdout

decksh in.dsh

read from file, write to stdout

decksh -o out.xml

read from stdin, write to file

decksh -o out.xml in.dsh

read from file, write to file

chmod +x in.dsh; ./in.dsh

executable deck

```
#!/path/to/decksh
deck
    slide
    ...
    eslide
edeck
```

Keywords and arguments

```
text "string...." x y n [font][color][op]
```

```
text "hello, world" 80 50 2
```

hello, world

```
text "hello, world" 80 40 2 "serif"
```

hello, world

```
text "hello, world" 80 30 2 "serif" "red"
```

hello, world

```
text "hello, world" 80 20 2 "serif" "red" 50
```

hello, world

Keywords

Structure

deck
edeck
slide
eslide
canvas

Loop

for
efor

Text

text
ctext
etext
textblock
textfile
textcode

Lists

list
blist
nlist
li
elist

Graphics

rect
ellipse
square
circle
polygon
arc
curve
line
hline
vline

Arrows

arrow
crarrow
clarrow
cuarrow
cdarrow

Images

image
cimage

Charts

dchart
legend

Assignments

```
// decksh assignments
x=10                                // number assignment
y=20
factor=2
what="hello world"                 // string assignment

size=x/factor                      // assignment with binop
text what x y size                 // text "hello world" 10 20 5

y-=10                              // assignment operation
size+=factor                       // assignment op, substitute
text what x y size                 // text "hello world" 10 10 7

for v=0 100 5                      // loop from 0 to 100 by 5
    line 100 v 0 v 0.1 "blue"      // blue horizontal lines
    line v 100 v 0 0.1 "red"       // red vertical lines
efor
```

Text

.hello world

text

x y size [font] [color] [op] [link]

The quick brown
fox jump over the
lazy dog

textblock

"text" x y width size [font] [color] [op] [link]

hello .world

ctext

x y size [font] [color] [op] [link]

This is the contents
of a file

textfile

"file" x y size [font] [color] [op] [sp]

hello world.

etext

x y size [font] [color] [op] [link]

```
package main

import "fmt"

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

textcode

"filename" x y width size [color]

Lists

One

Two

Three

Four

- One

- Two

- Three

- Four

1. One

2. Two

3. Three

4. Four

`list`

x y size [font] [color] [opacity] [spacing]

`blist`

x y size [font] [color] [opacity] [spacing]

`nlist`

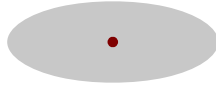
x y size [font] [color] [opacity] [spacing]

Graphics



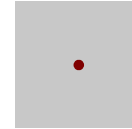
rect

x y w h [color] [op]



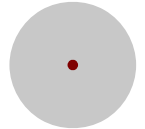
ellipse

x y w h [color] [op]



square

x y w [color] [opacity]



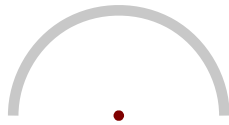
circle

x y w [color] [op]



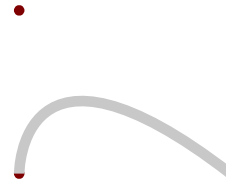
polygon

"xc" "yc" [color] [op]



arc

x y w h a1 a2 [lw] [color] [op]



curve

x1 y2 x2 y2 x3 y3 [color] [op]



line

x1 y2 x2 y2 [lw] [color] [op]



hline

x y len [lw] [color] [op]



vline

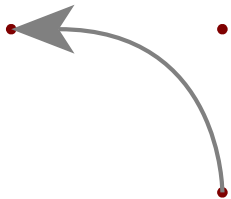
x y len [lw] [color] [op]

Arrows



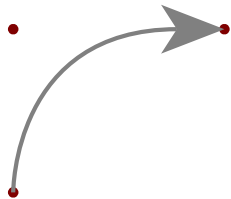
arrow

x1 y1 x2 y2 [aw] [ah] [lw] [color] [op]



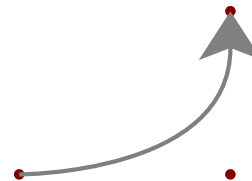
lcarrow

x1 y1 x2 y2 x3 y3 [lw] [aw] [ah] [color] [op]



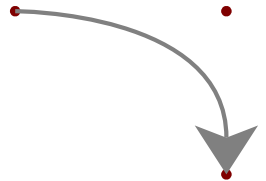
rcarrow

...



ucarrow

...



dcarrow

...

Images



image

"file" x y w h [scale] [link]

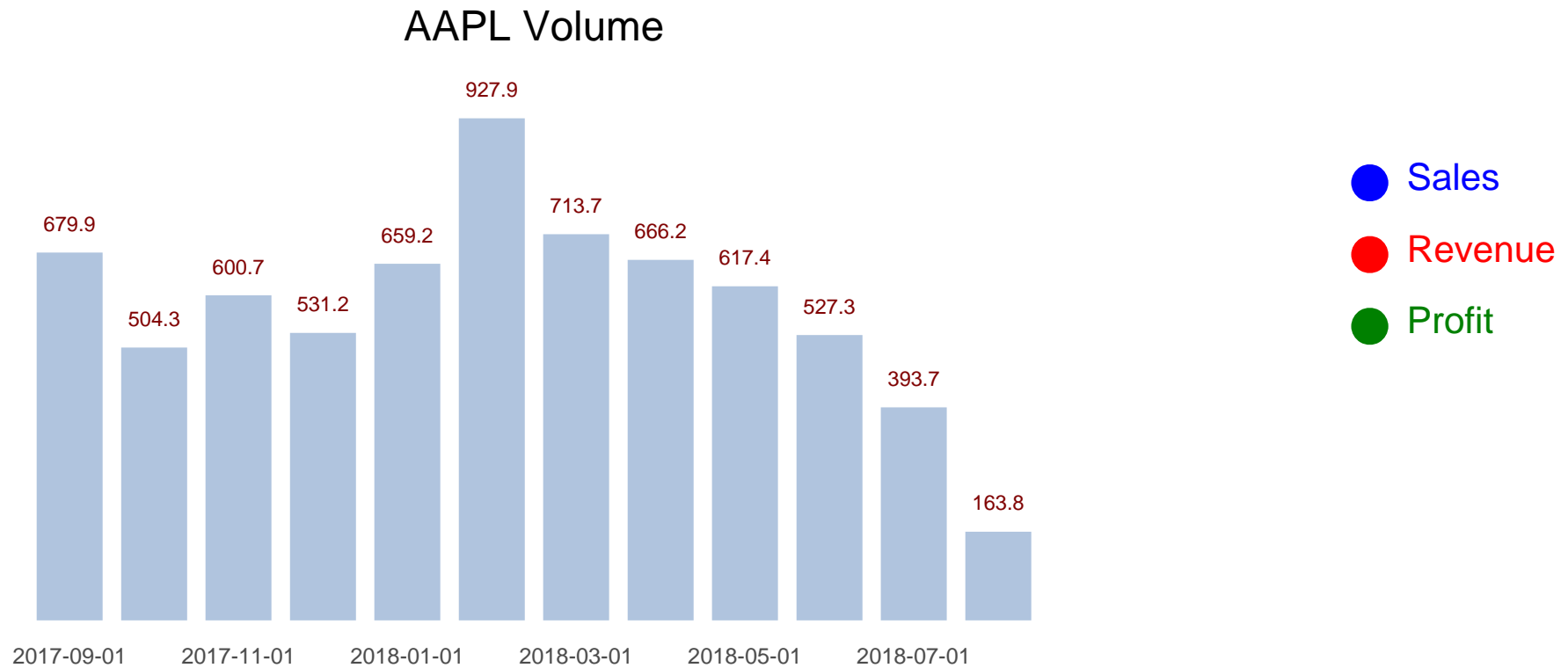


Up in the clouds

cimage

"file" "caption" x y w h [scale] [link]

Charts



dchart

[args]

legend

x y size [font] [color]

decksh example.dsh | pdf

deck

```
slide "rgb(250,250,250)" "black"
  ctext  "Deck elements" 50 90 5
  image  "follow.jpg"    70 50 640 480 50
  blist  10 75 3
    li "text, image, list"
    li "rect, ellipse, polygon"
    li "line, arc, curve"
  elist

  gy=10
  rect  15 gy 8 6 "rgb(127,0,0)"
  ellipse 27.5 gy 8 6 "rgb(0,127,0)"
  line  50 gy 60 gy
  curve 80 gy 95 30 90 gy
  arc  70 gy 10 8 0 180 0.1 "rgb(0,0,127)"
  polygon "37 37 45" "13 7 10" "rgb(0,0,127)"

  opts="-fulldeck=f -textsize 1 -xlabel=2 -barwidth 1.5"
  dchart -left 10 -right 42 -top 42 -bottom 25 opts AAPL.d
eslide
edeck
```

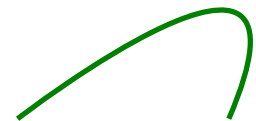
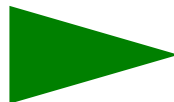
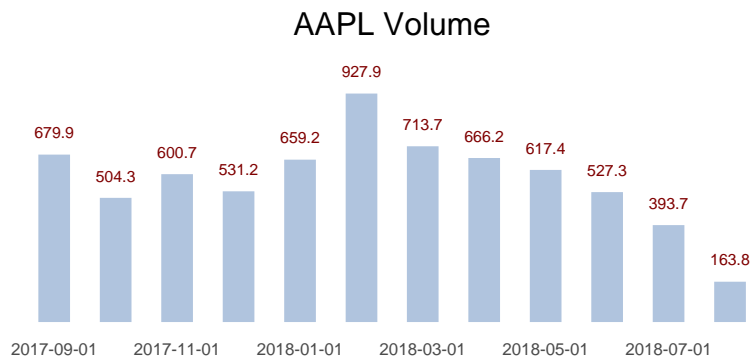
Deck elements

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



Deck elements

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



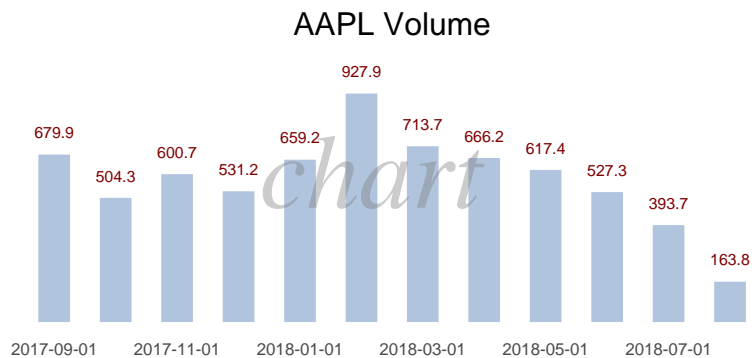
text

Deck elements

list

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

image



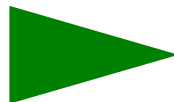
rect



ellipse



polygon



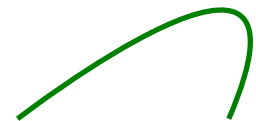
line



arc

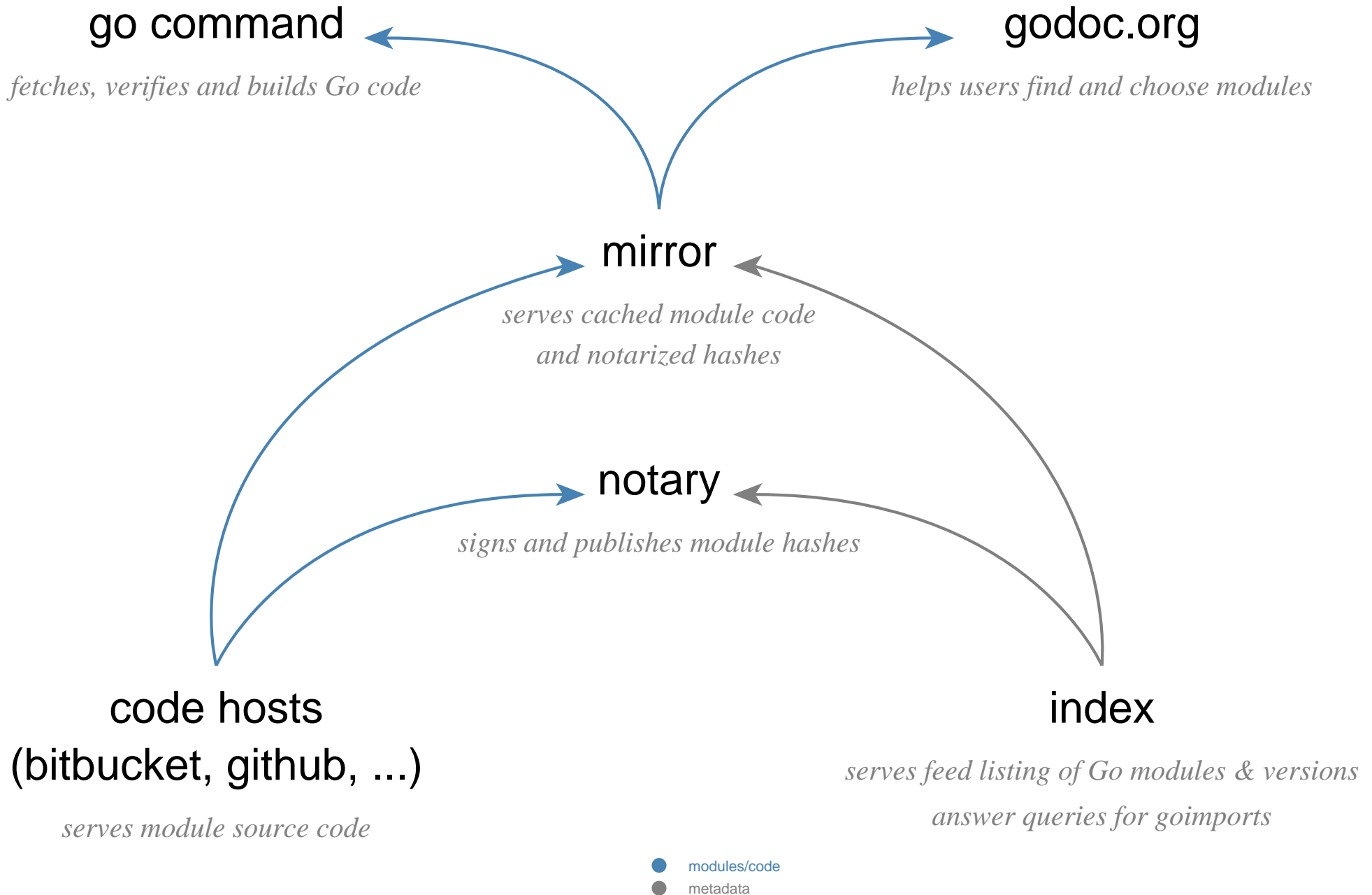


curve



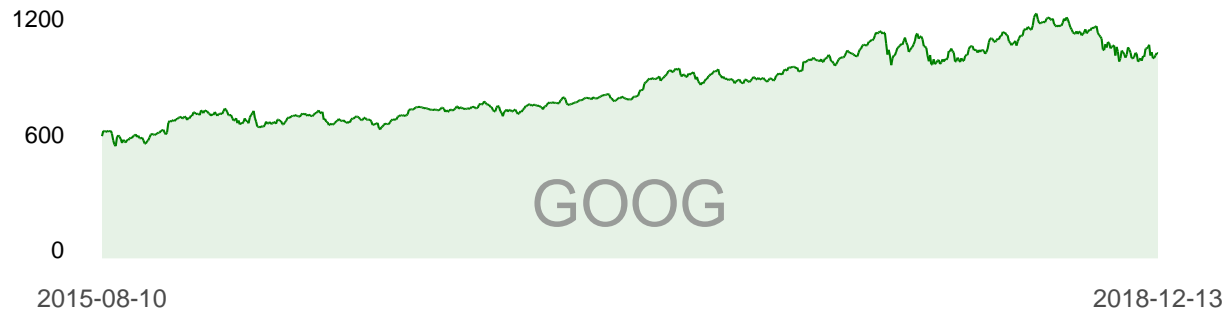
Examples

Go Module Information Flows





Pichai



+38.19%



Nadella



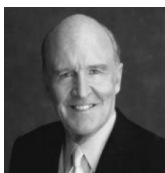
+66.79%



Cook



+68.56%

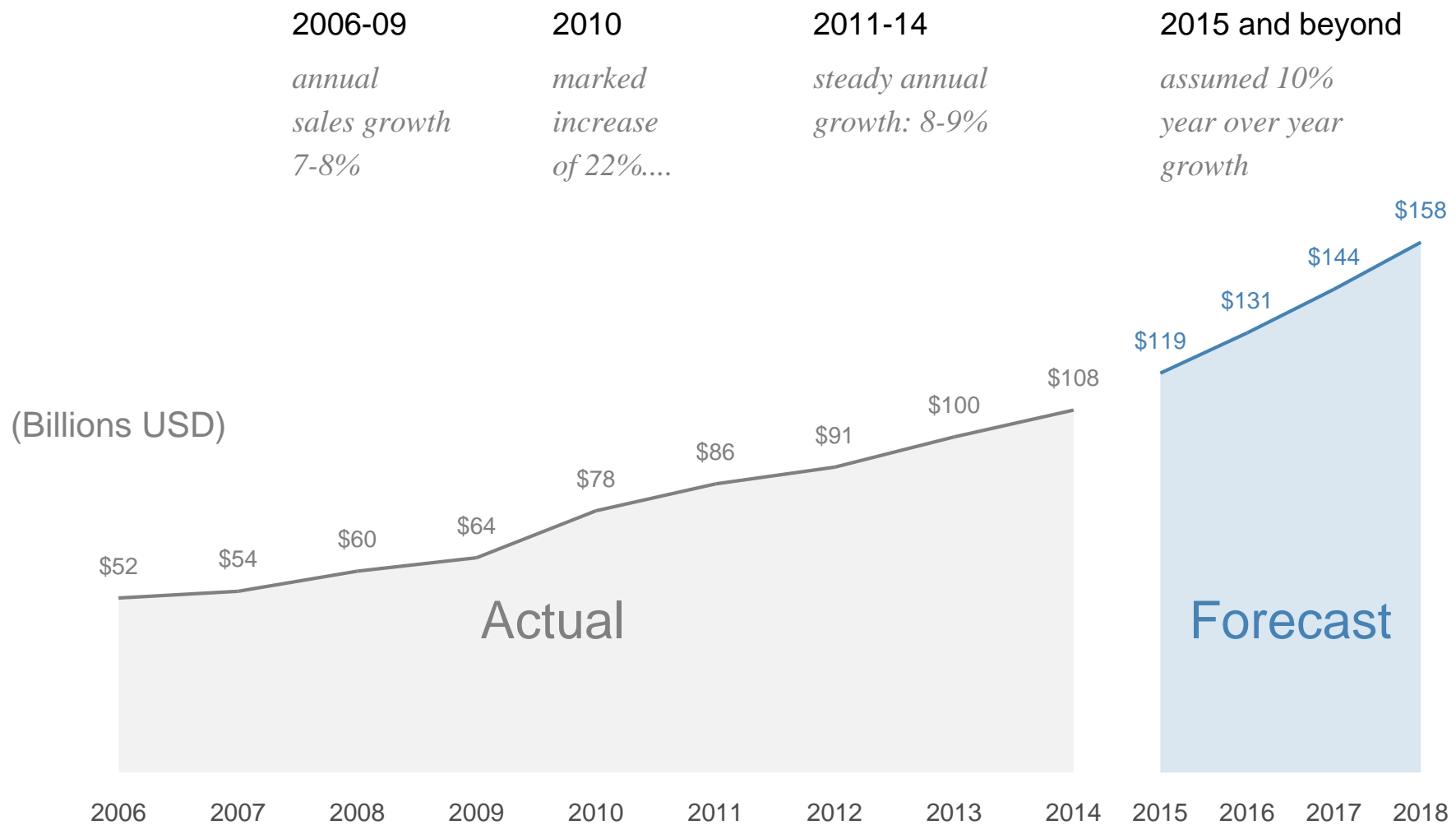


Welch

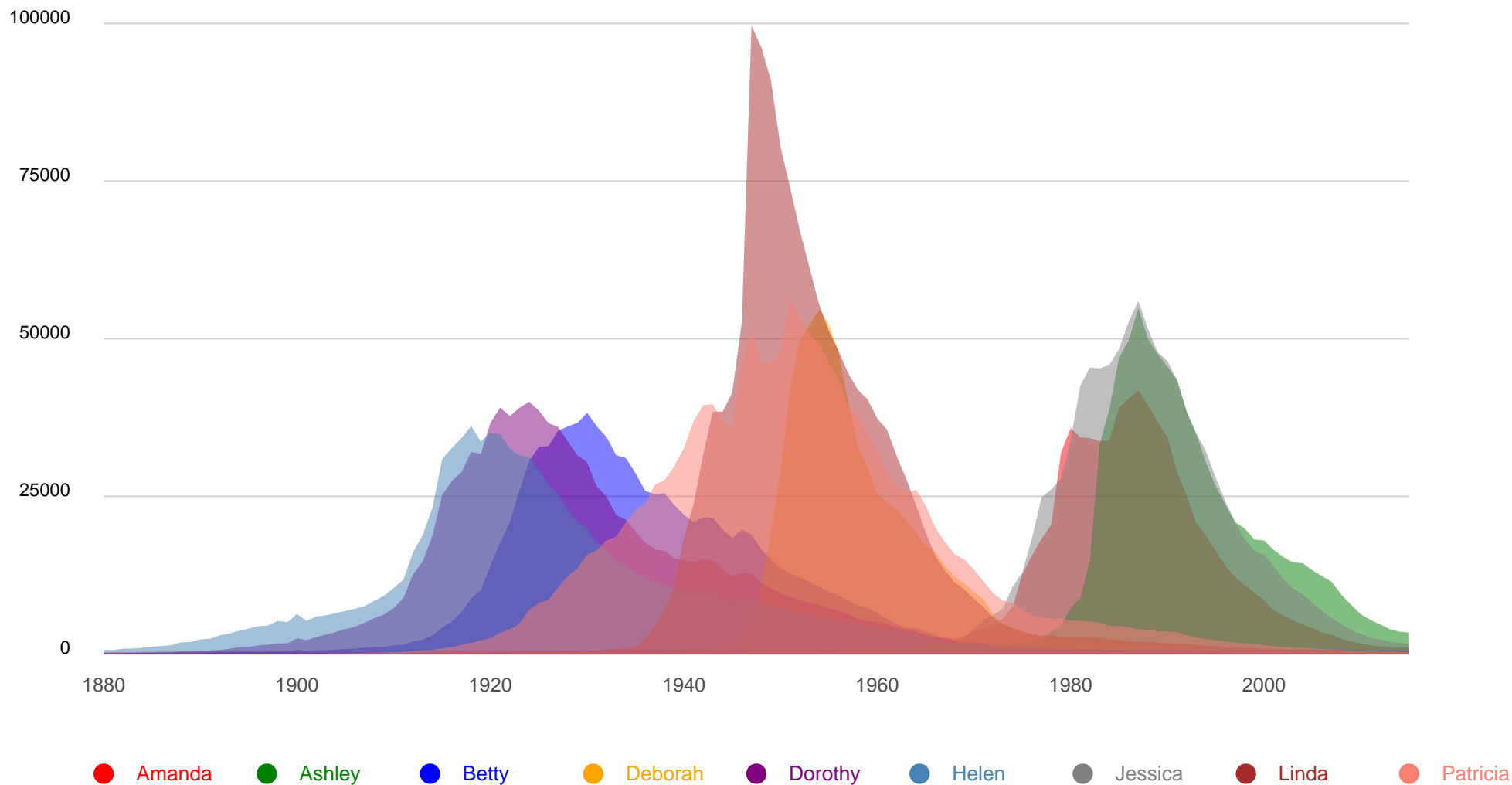


+96.56%

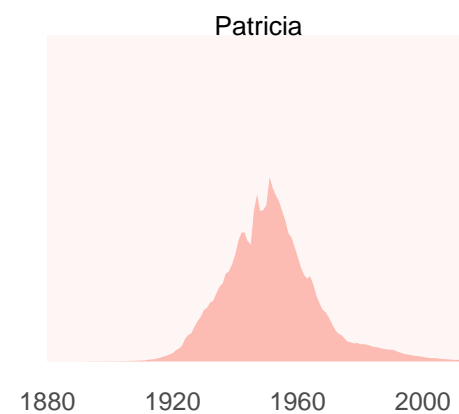
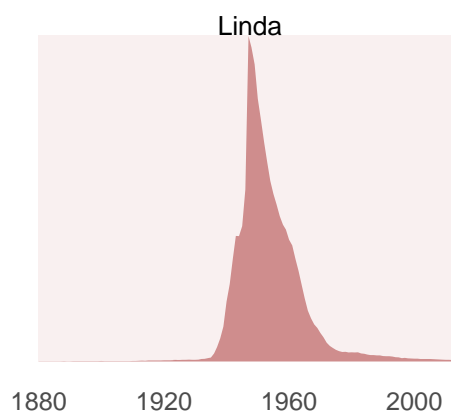
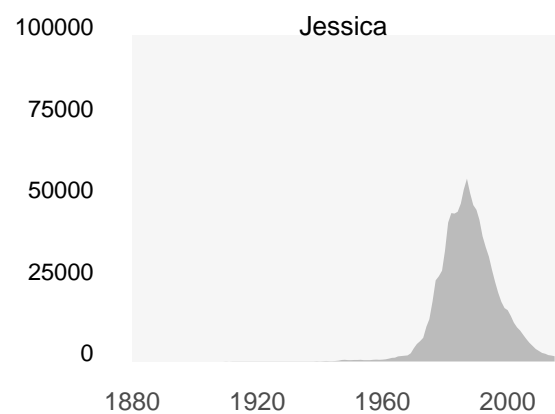
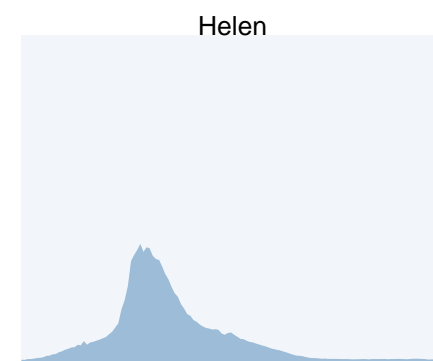
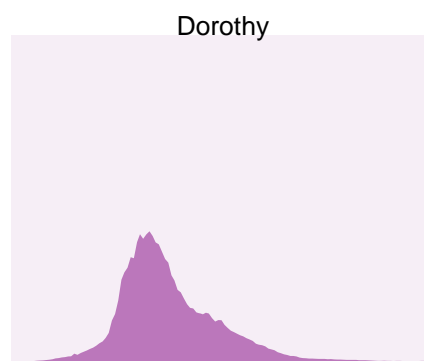
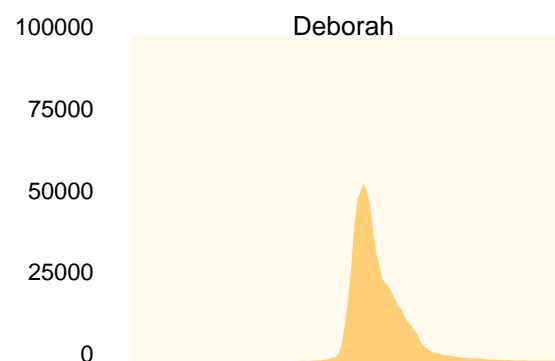
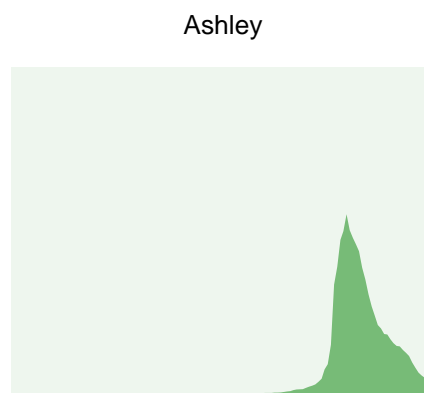
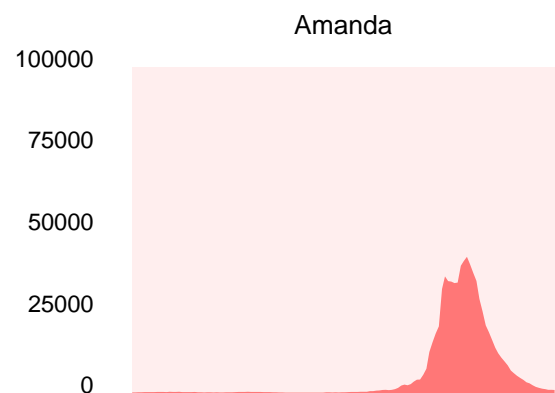
Sales over time



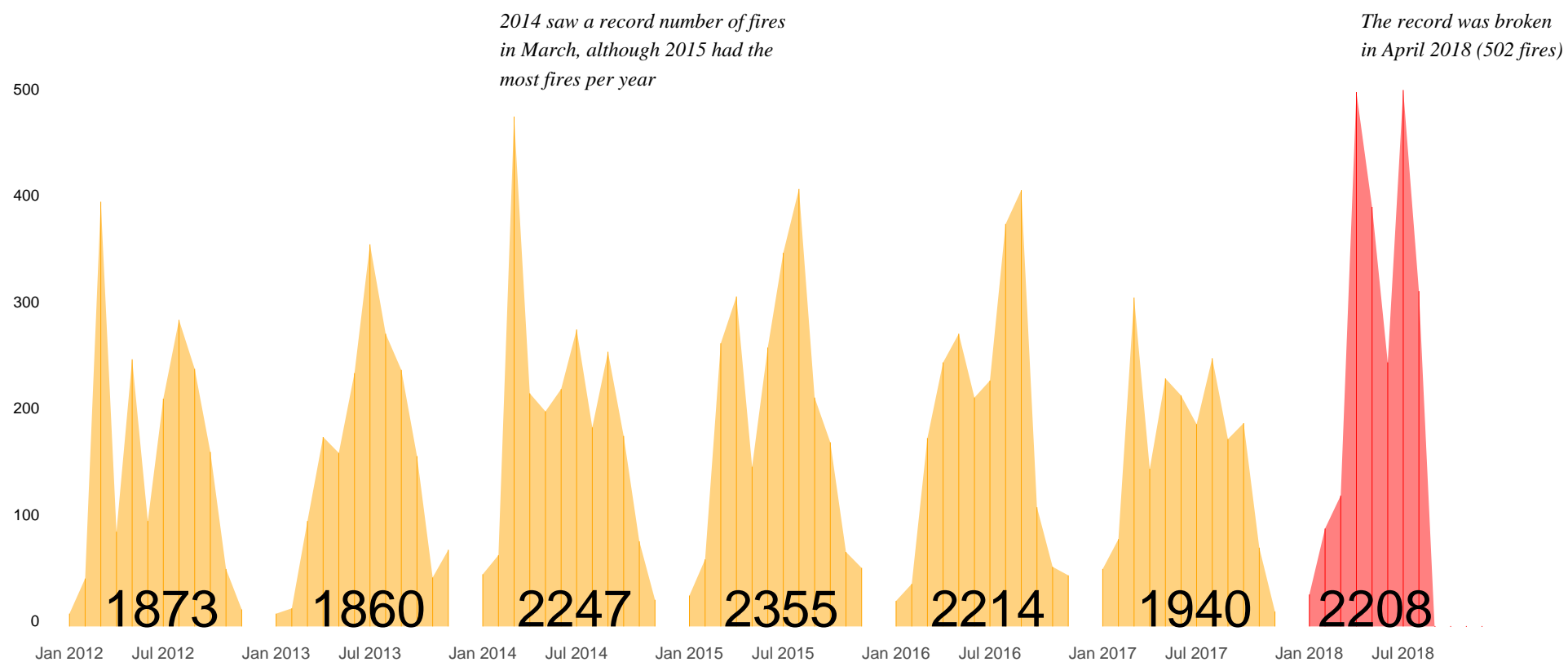
Evolution of Baby Names in the US: 1880-2015



Evolution of Baby Names in the US: 1880-2015



German Wildfires 2012-2018



go get it

deck

`github.com/ajstarks/deck`

decksh

`github.com/ajstarks/deck/cmd/decksh`

pdfdeck

`github.com/ajstarks/deck/cmd/pdfdeck`

dchart

`github.com/ajstarks/deck/cmd/dchart`

deck fonts

`github.com/ajstarks/deckfonts`