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

90

80

70

60

Percent Grid

50

40

30

20

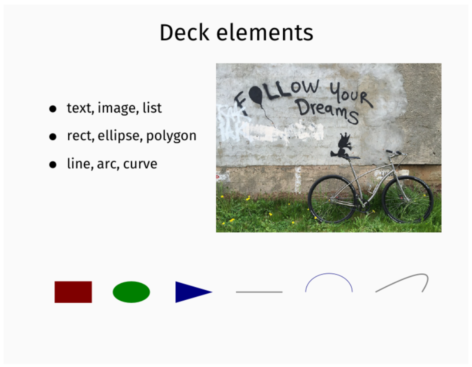
10

10 20 30 40 50 60 70 80 90

$$\begin{tikzcd}
 & & \text{SVG} \\
\text{decksh} \arrow[r] & \text{deck markup} \arrow[r] & \text{PDF} \\
 & & \text{PNG}
\end{tikzcd}$$

```
deck
    slide "rgb(250,250,250)" "black"
        ctext   "Deck elements" 50 90 5
        image   "follow.jpg"    70 60 640 480 60
        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)"
        line    50 20 60 20
        curve   80 20 95 30 90 20
        arc     70 20 10 8 0 180 0.1  "rgb(0,0,127)"
        polygon "37 37 45" "17 23 20" "rgb(0,0,127)"
    eslide
edeck
```

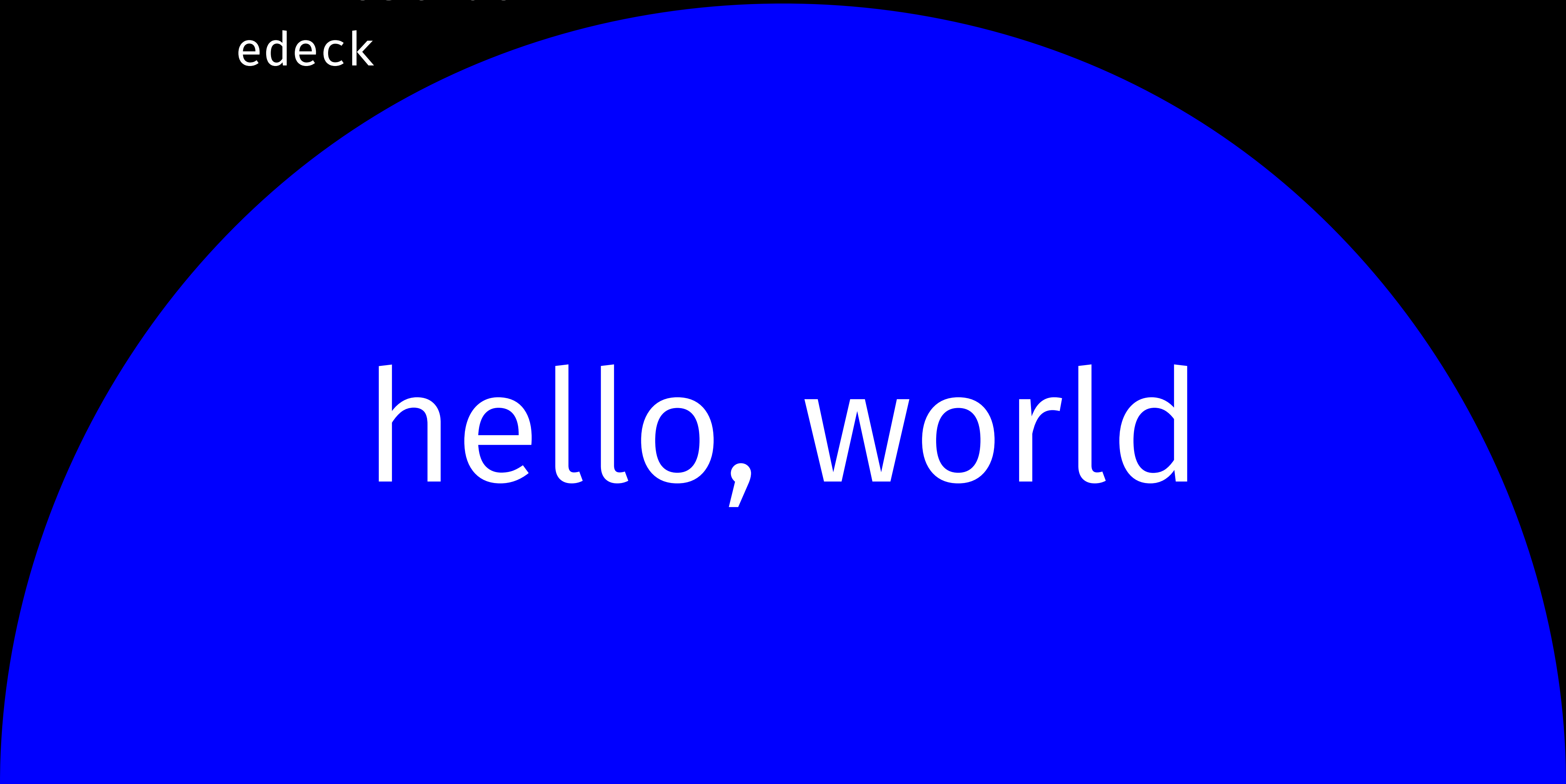
```
<deck>
<slide bg="rgb(250,250,250)" fg="black">
<text align="c" xp="50" yp="90" sp="5" >Deck elements</text>
<image name="follow.jpg" xp="70" yp="60" width="640" height="480" scale="60"/>
<list type="bullet" xp="10" yp="70" sp="3" >
<li>text, image, list</li>
<li>rect, ellipse, polygon</li>
<li>line, arc, curve</li>
</list>
<rect xp="15" yp="20" wp="8" hp="6" color="rgb(127,0,0)"/>
<ellipse xp="27.5" yp="20" wp="8" hp="6" color="rgb(0,127,0)"/>
<line xp1="50" yp1="20" xp2="60" yp2="20"/>
<curve xp1="80" yp1="20" xp2="95" yp2="30" xp3="90" yp3="20"/>
<arc xp="70" yp="20" wp="10" hp="8" a1="0" a2="180" sp="0.1" color="rgb(0,0,127)"/>
<polygon xc="37 37 45" yc="17 23 20" color="rgb(0,0,127)"/>
</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
```



Running decksh

`decksh`	*read from stdin, write to stdout*
`decksh mydeck`	*read from file, write to stdout*
`decksh -o out.xml`	*read from stdin, write to file*
`decksh -o out.xml mydeck`	*read from file, write to file*
`chmod +x mydeck; ./mydeck`	*executable deck*

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

`keyword args [optionals]`

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

rarrow
larrow
uarrow
darrow
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

hello world

hello world

text

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

ctext

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

etext

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]

This is the contents of a file

textfile

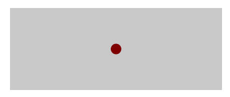
"file" x y size [font] [color] [op] [sp]

```
package main

import "fmt"

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

textcode

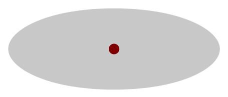
"filename" x y width size [color]

Graphics



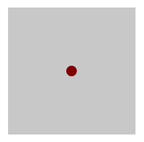
`rect`

x y w h [color] [op]



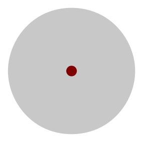
`ellipse`

x y w h [color] [op]



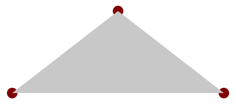
`square`

x y w [color] [opacity]



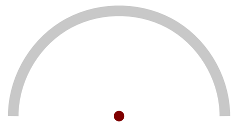
`circle`

x y w [color] [op]



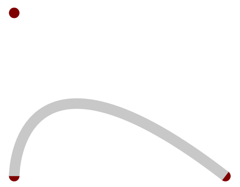
`polygon`

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



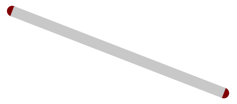
`arc`

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



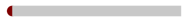
`curve`

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



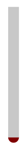
`line`

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



`hline`

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



`vline`

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

Images

Up in the clouds

`image`

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

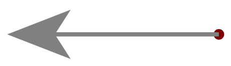
`cimage`

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

Lists

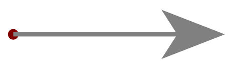
One	● One	1. One
Two	● Two	2. Two
Three	● Three	3. Three
Four	● Four	4. Four
Five	● Five	5. Five
`list`	`blist`	`nlist`
x y size [font] [color] [opacity] [spacing]	*x y size [font] [color] [opacity] [spacing]*	*x y size [font] [color] [opacity] [spacing]*

Arrows



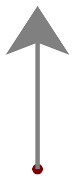
`larrow`

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



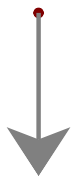
`rarrow`

...



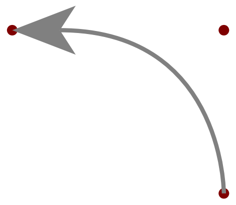
`uarrow`

...



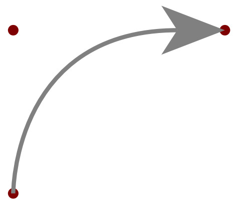
`darrow`

...



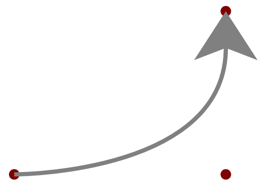
`lcarrow`

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



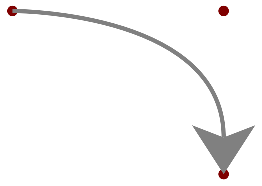
`rcarrow`

...



`ucarrow`

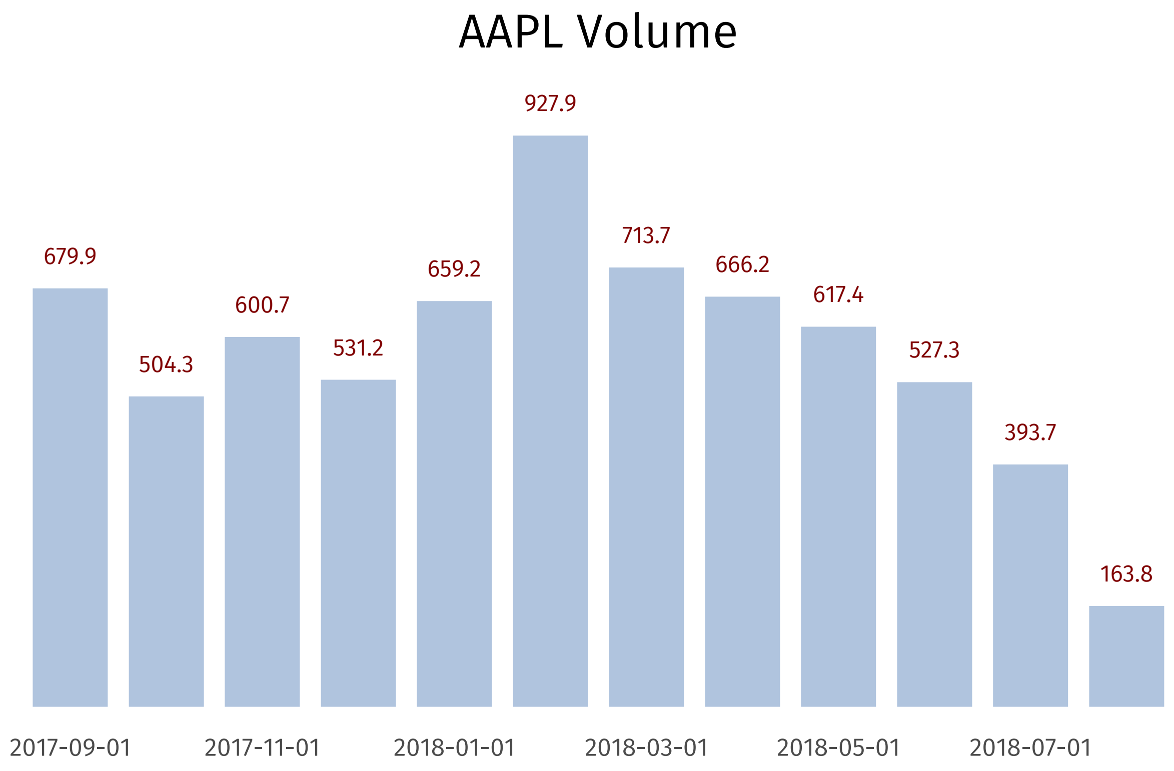
...

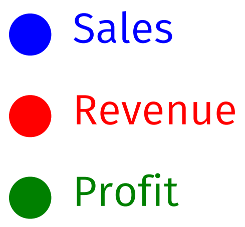


`dcarrow`

...

Charts





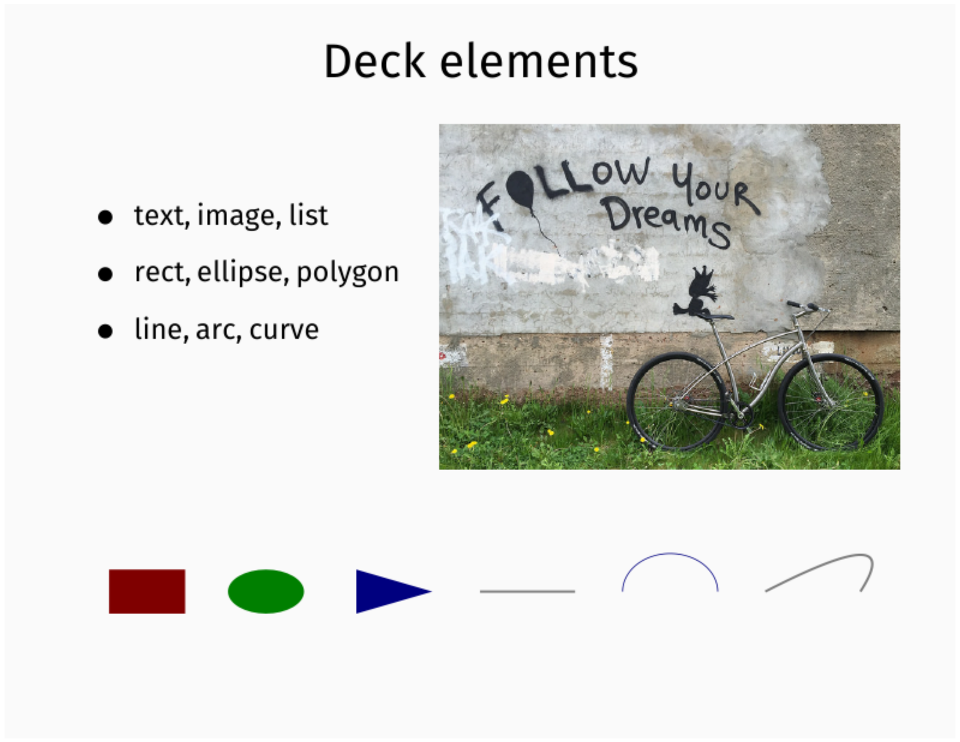
`dchart`

[args]

`legend`

x y size [font] [color]

```
deck
    slide "rgb(250,250,250)" "black"
        ctext   "Deck elements" 50 90 5
        image   "follow.jpg"    70 60 640 480 60
        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)"
        line    50 20 60 20
        curve   80 20 95 30 90 20
        arc     70 20 10 8 0 180 0.1  "rgb(0,0,127)"
        polygon "37 37 45" "17 23 20" "rgb(0,0,127)"
    eslide
edeck
```

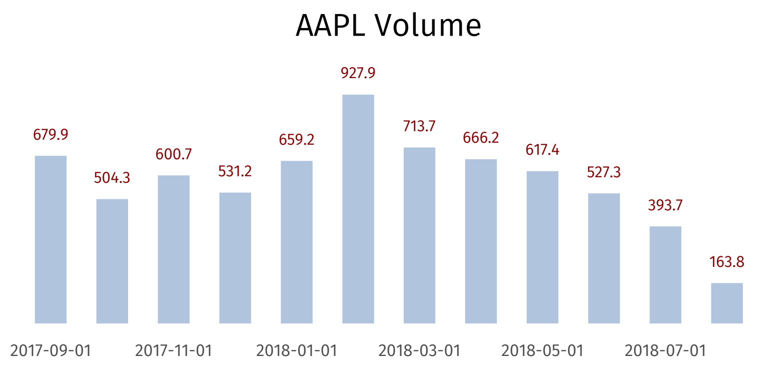

Deck elements

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

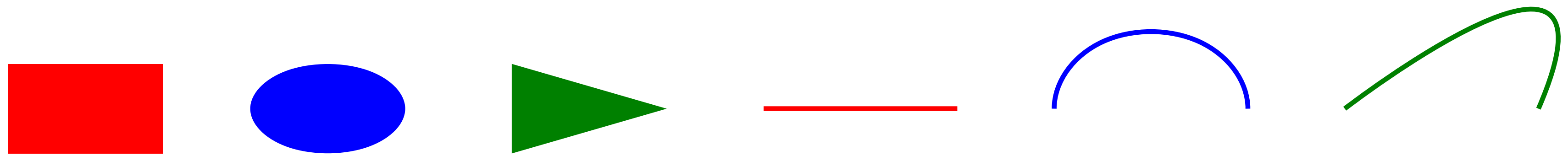
```
decksh example.dsh | pdf
```

Deck elements

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



Dreams



text

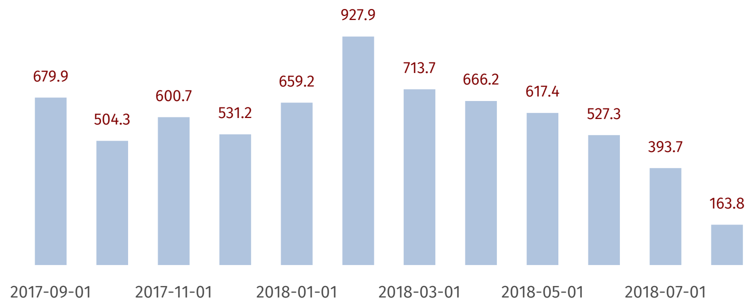
Deck elements

list

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

chart

AAPL Volume



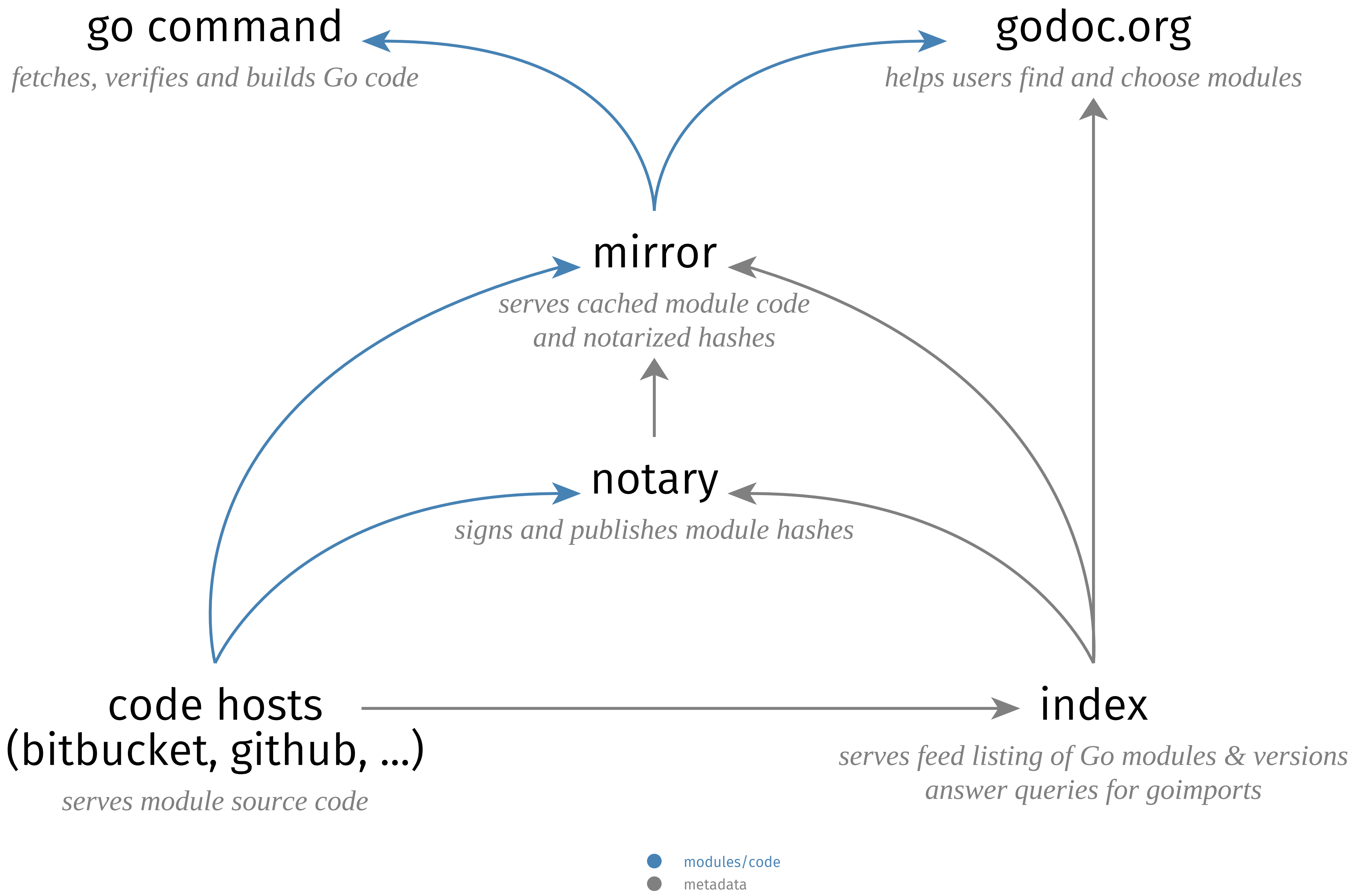
image

Dreams

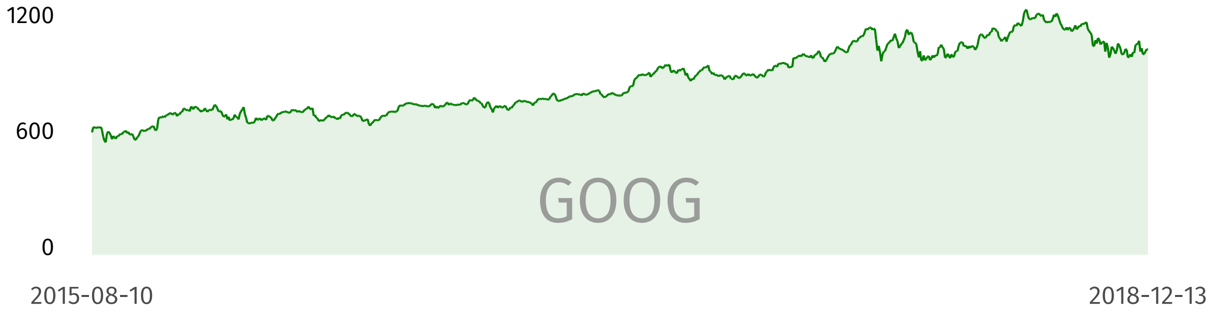
rect *ellipse* *polygon* *line* *arc* *curve*

Examples

Go Module Information Flows

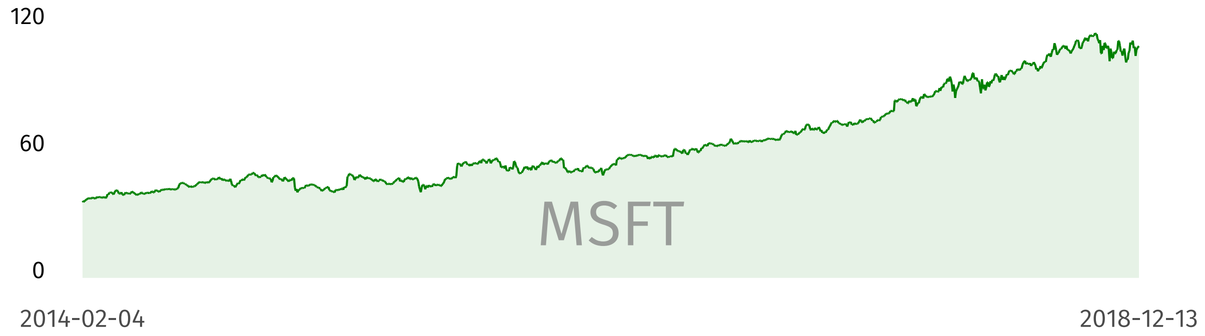

go command
fetches, verifies and builds Go code
godoc.org
helps users find and choose modules
mirror
serves cached module code
and notarized hashes
notary
signs and publishes module hashes
code hosts
(bitbucket, github, ...)
serves module source code
index
serves feed listing of Go modules & versions
answer queries for goimports
● modules/code
● metadata

Pichai



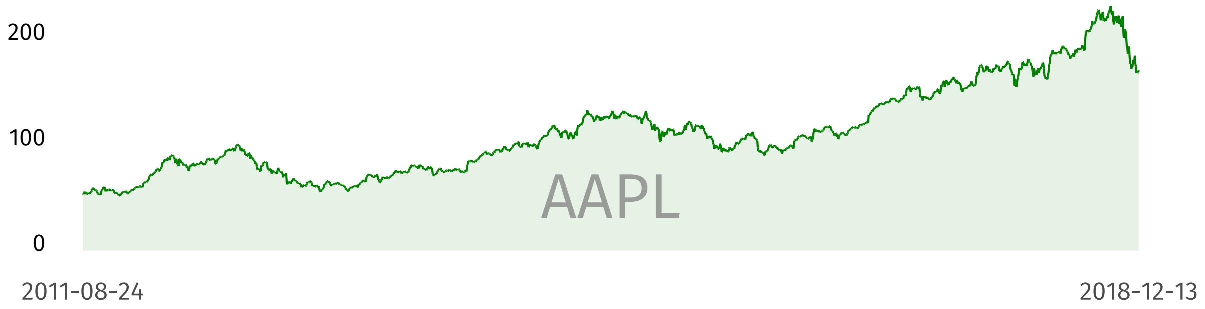
+38.19%

Nadella

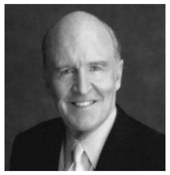


+66.79%

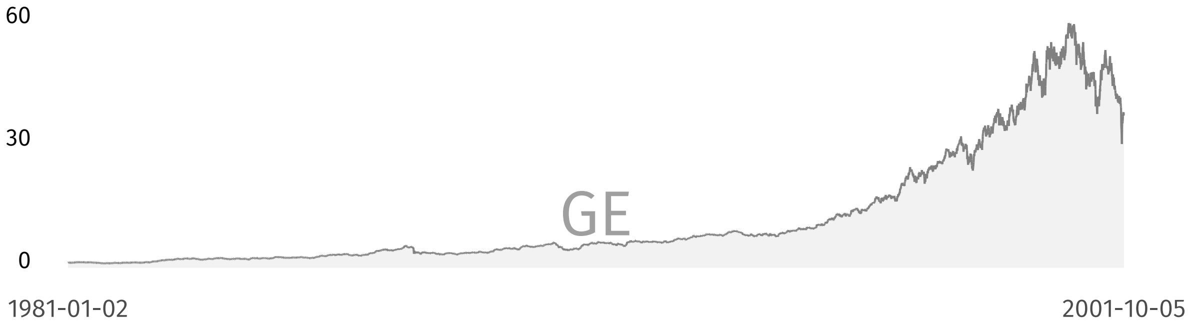
Cook



+68.56%

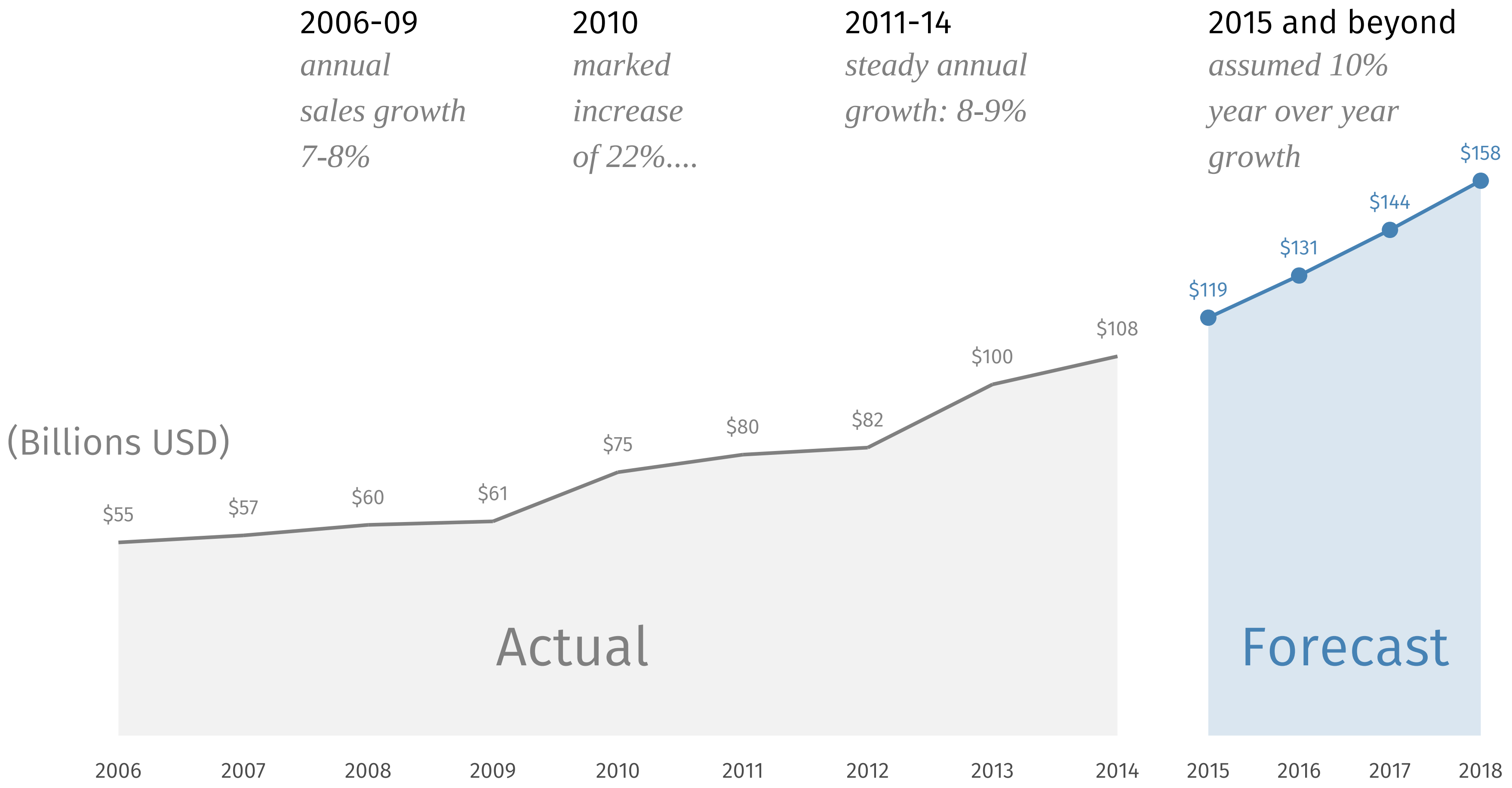


Welch



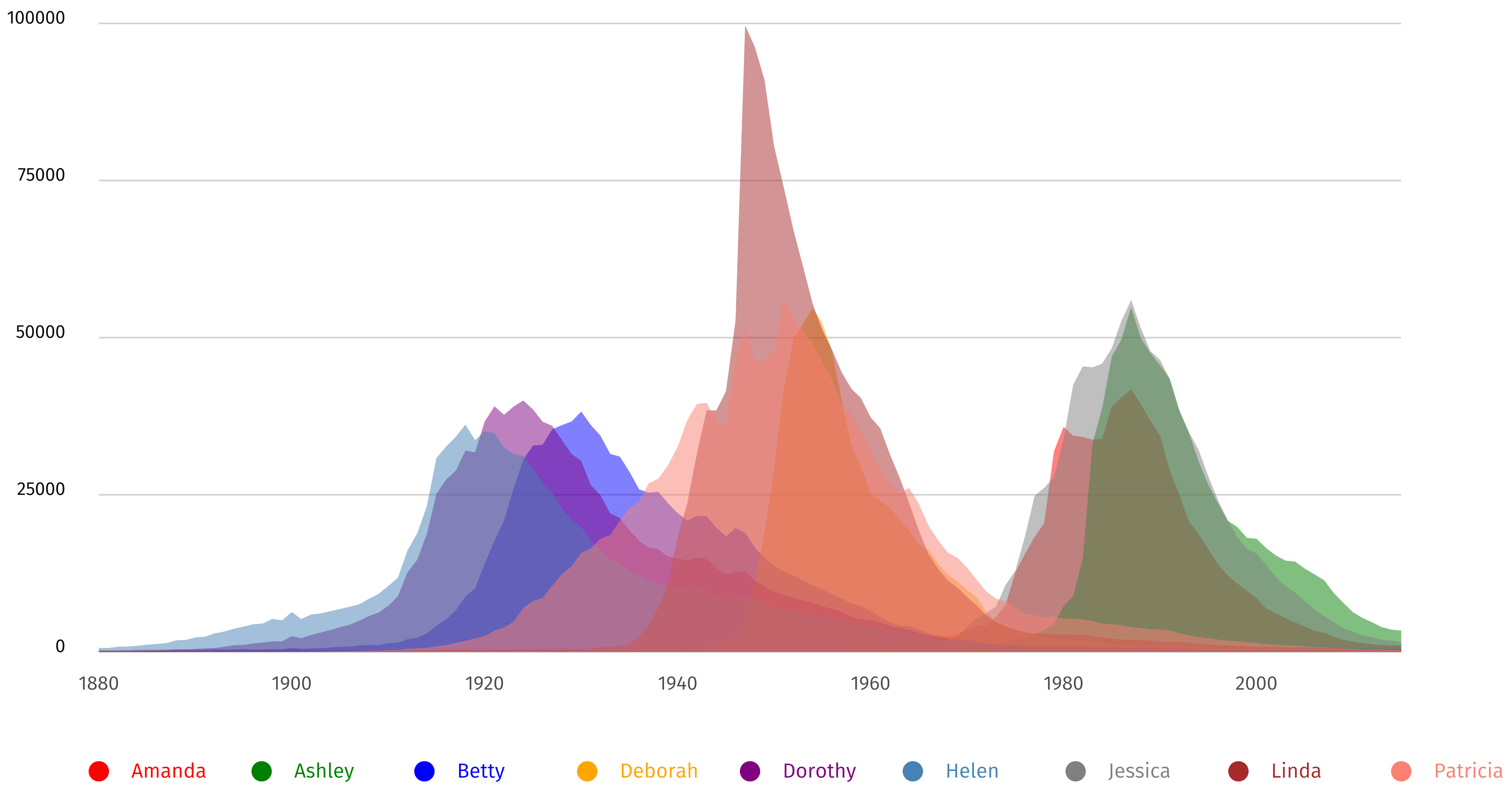
+96.56%

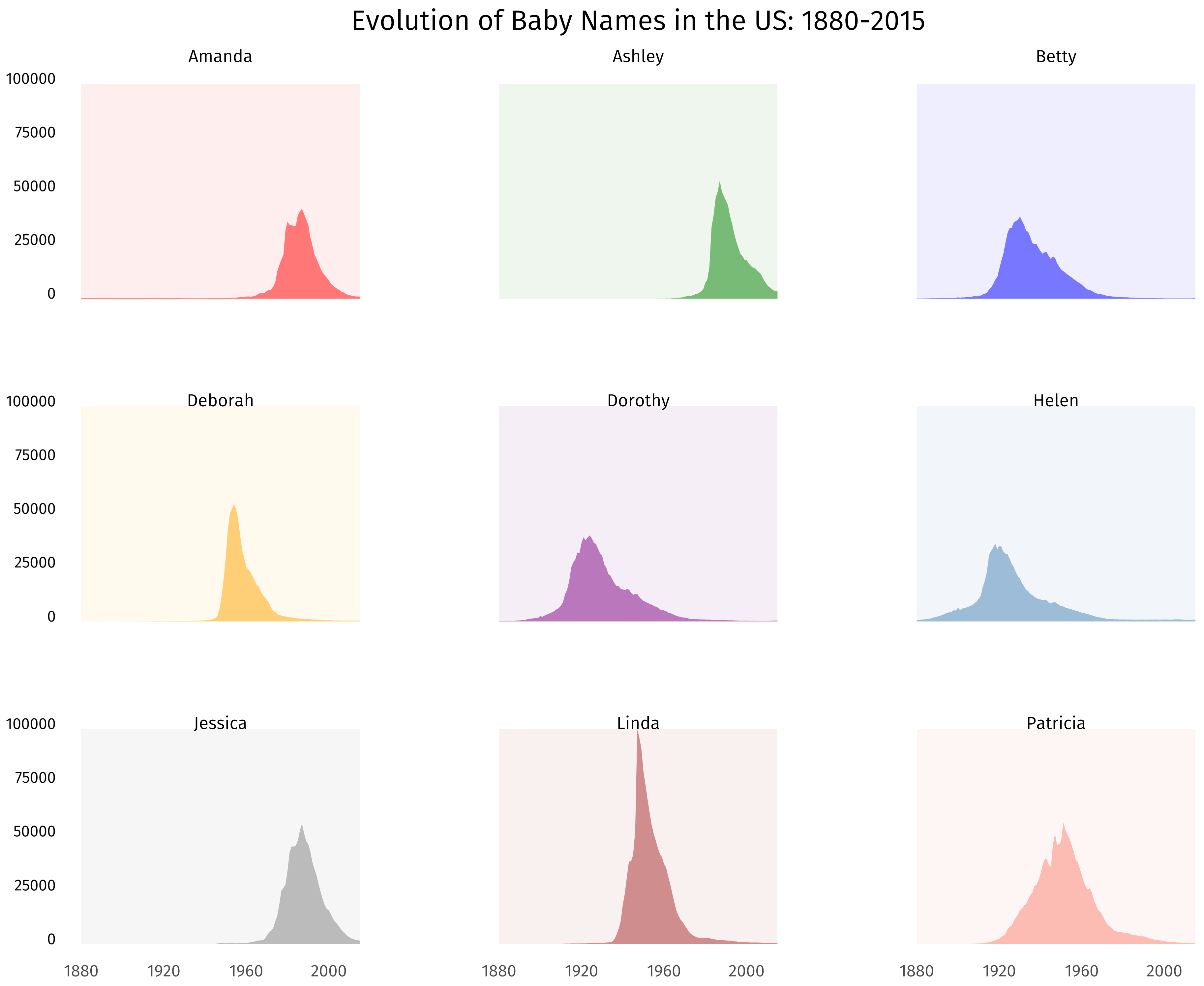
Sales over time



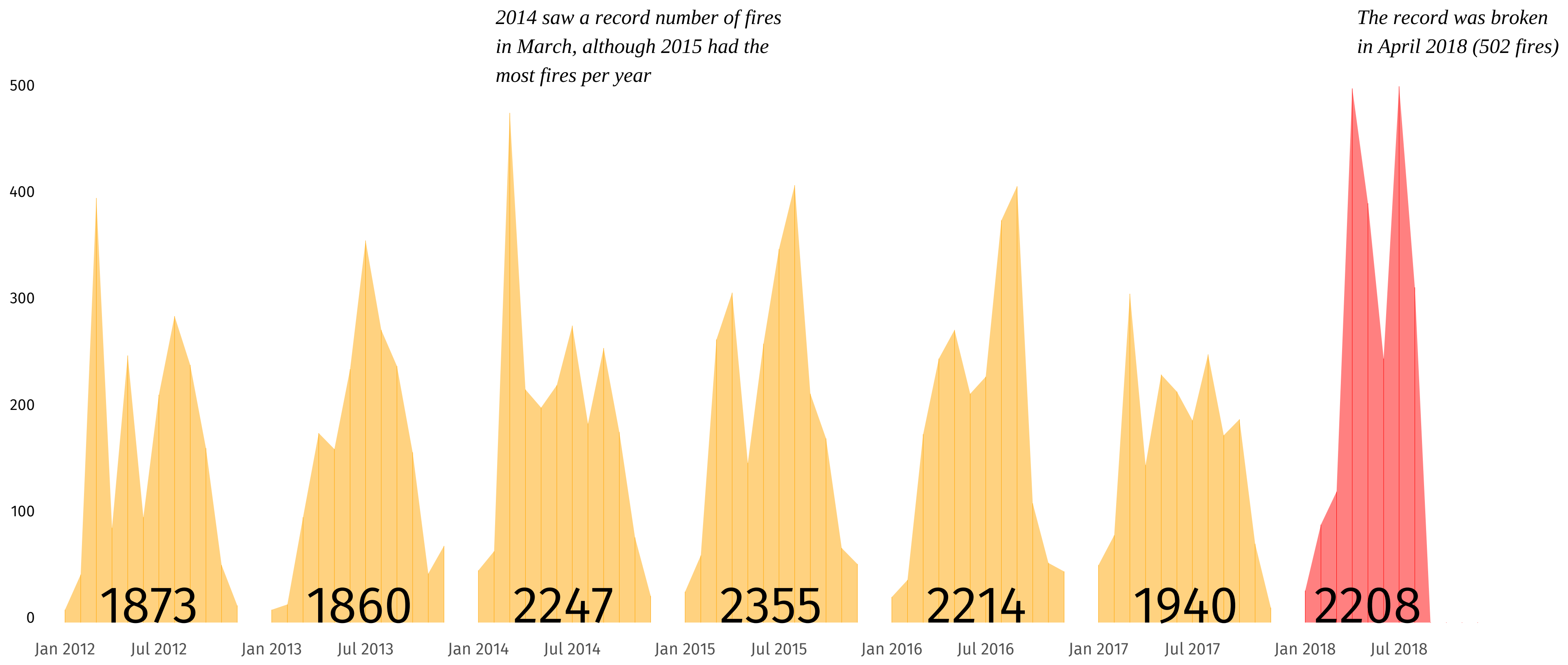
Storytelling with data, pg. 154

Evolution of Baby Names in the US: 1880-2015




Evolution of Baby Names in the US: 1880-2015
Amanda
Ashley
Betty
Deborah
Dorothy
Helen
Jessica
Linda
Patricia
100000
75000
50000
25000
0
1880
1920
1960
2000

German Wildfires 2012-2018



Chartable a blog by Datawrapper, Weekly Chart, September 6, 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