ggvis

Hadley Wickham • @hadleywickham
Winston Chang • @winston_chang
RStudio

May 2014

What is ggvis?

- A grammar of graphics (like `ggplot2`)
- Reactive (interactive & dynamic) (like `shiny`)
- A pipeline (a la `dplyr`)
- Of the web (drawn with `vega`)

```
library(ggvis)

# Number of packages downloaded each day in 2013
# from RStudio cran mirror
downloads <- readRDS("downloads.rds")
head(downloads)
#> Source: local data frame [6 x 3]
#>
#>         date     n n_ip
#> 1 2013-12-26 48607 3256
#> 2 2013-12-25 24411 2399
#> 3 2013-12-24 33838 3078
#> 4 2013-12-23 70296 4374
#> 5 2013-12-22 45301 2905
#> 6 2013-12-21 36120 3014
```

A grammar of graphics

```
downloads %>%
  ggvis(~date, ~n) %>%
  layer_lines()
```

Start with data

```
downloads %>%
  ggvis(~date, ~n) %>%
  layer_lines()
```

```
downloads
  ggvis(~date, ~n) %>%
  layer_lines()
```

Map vertical position to date

Map horizontal position to downloads

```
downloads %>%
  ggvis(~date, ~n) %>%
  layer_lines()
```

Layer on lines

Demo

Reactive

```
base <- downloads %>%
  ggvis(~date, ~n_ip) %>%
  layer_lines()

slider <- input_slider(0.1, 1, value = 0.75)
base %>%
  layer_smooths(stroke := "red", span = slider)
```

Demo

Data pipeline

```r
# ggvis has a stricly functional interface:
# Each ggvis function takes a visualisation as
# input (the first argument) and returns the
# a modified visualisation as output

# This means we need to create plots like this:
p <- ggvis(downloads, ~date, ~n)
p <- layer_lines(p)
p <- layer_smooths(p)
p

# (Interestingly this is also how ggplot worked)
# Or we could nest function calls

layer_smooths(
  layer_lines(
    ggvis(downloads, ~date, ~n)
  )
)

# But neither is very readable
# Instead we use the pipe operator (pronounced
# then) from the magrittr package.
#
# x %>% f(y) is equivalent to f(x, y)
#
# This makes it much easier to use!

downloads %>%
  ggvis(~date, ~n) %>%
  layer_lines()
```

```r
# Not surprisingly this also works with dplyr
# so you can do data manipulation inside
# a plot with familiar functions
library(dplyr)
library(lubridate)

downloads %>%
  ggvis(~date, ~n) %>%
  group_by(date = floor_date(date, "week")) %>%
  summarise(n = sum(n), days = n()) %>%
  filter(days == 7) %>%
  layer_lines()
```

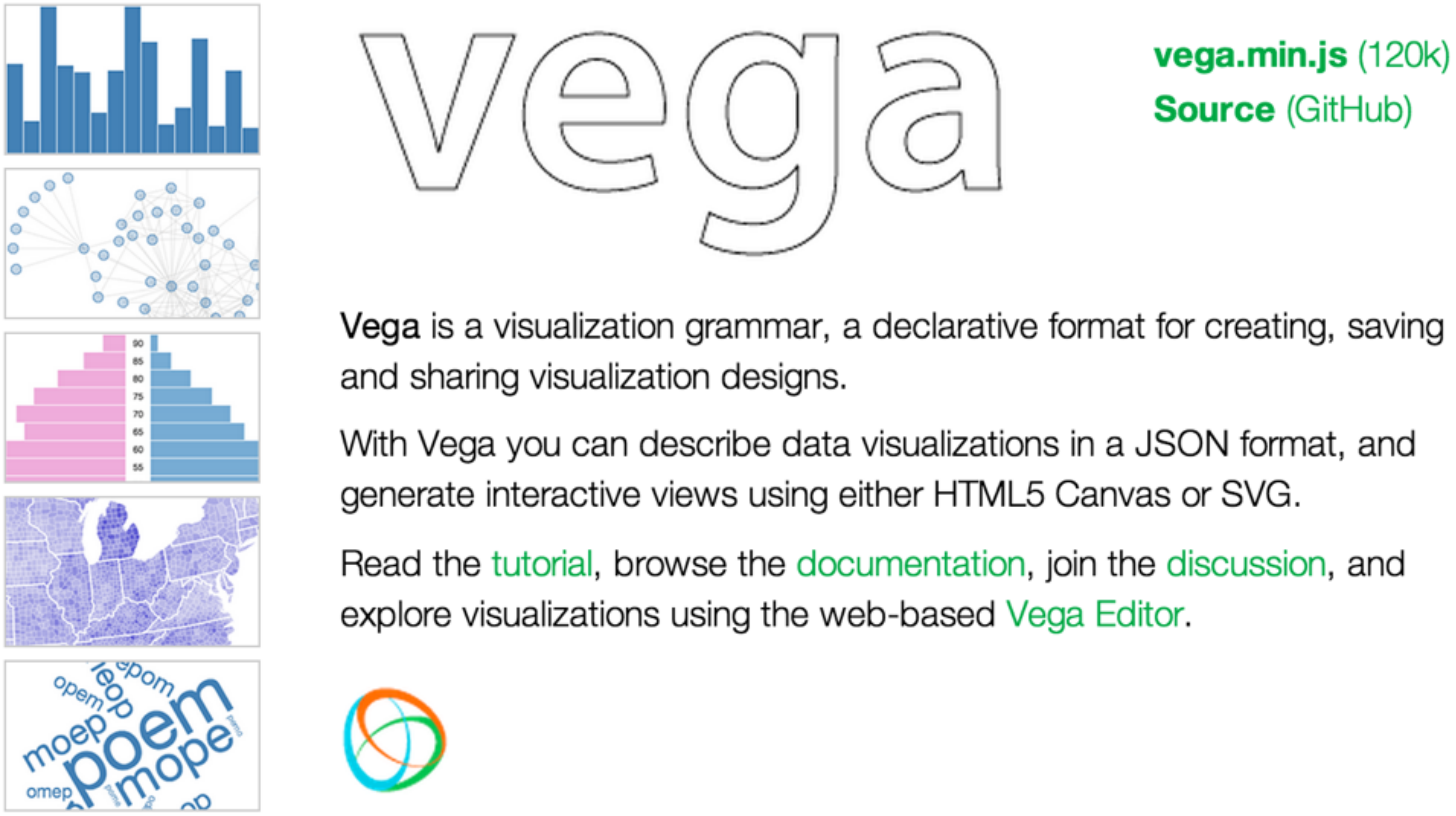
```
# And behind every layer function is a compute
# function that just modifies the data

downloads %>%
  compute_smooth(n ~ date)
#>          pred_    resp_
#> 1  2013-01-01 30595.30
#> 2  2013-01-05 32307.55
#> 3  2013-01-10 33970.64
#> 4  2013-01-14 35582.63
#> 5  2013-01-19 37141.57
#> 6  2013-01-23 38645.52
#> 7  2013-01-28 40092.52
#>...
```

Of the web

```
base <- downloads %>%
  ggvis(~date, ~n_ip) %>%
  layer_lines()

slider <- input_slider(0.1, 1, value = 0.75)
base %>%
  layer_smooths(stroke := "red", span = slider)
```



vega

vega.min.js (120k)

Source (GitHub)

Vega is a visualization grammar, a declarative format for creating, saving and sharing visualization designs.

With Vega you can describe data visualizations in a JSON format, and generate interactive views using either HTML5 Canvas or SVG.

Read the tutorial, browse the documentation, join the discussion, and explore visualizations using the web-based Vega Editor.

http://trifacta.github.io/vega/

R markdown demo

Future

Upcoming versions

- 0.3: zoom & pan; facetting + sub visualisations. (~1 July)
- 0.4: ggplot2 feature parity; mobile compatibility (~1 Oct)
- 0.5: performance improvements (~1 Dec)

Google for

“ggvis”