# ONLINE DATA VISUALIZATION

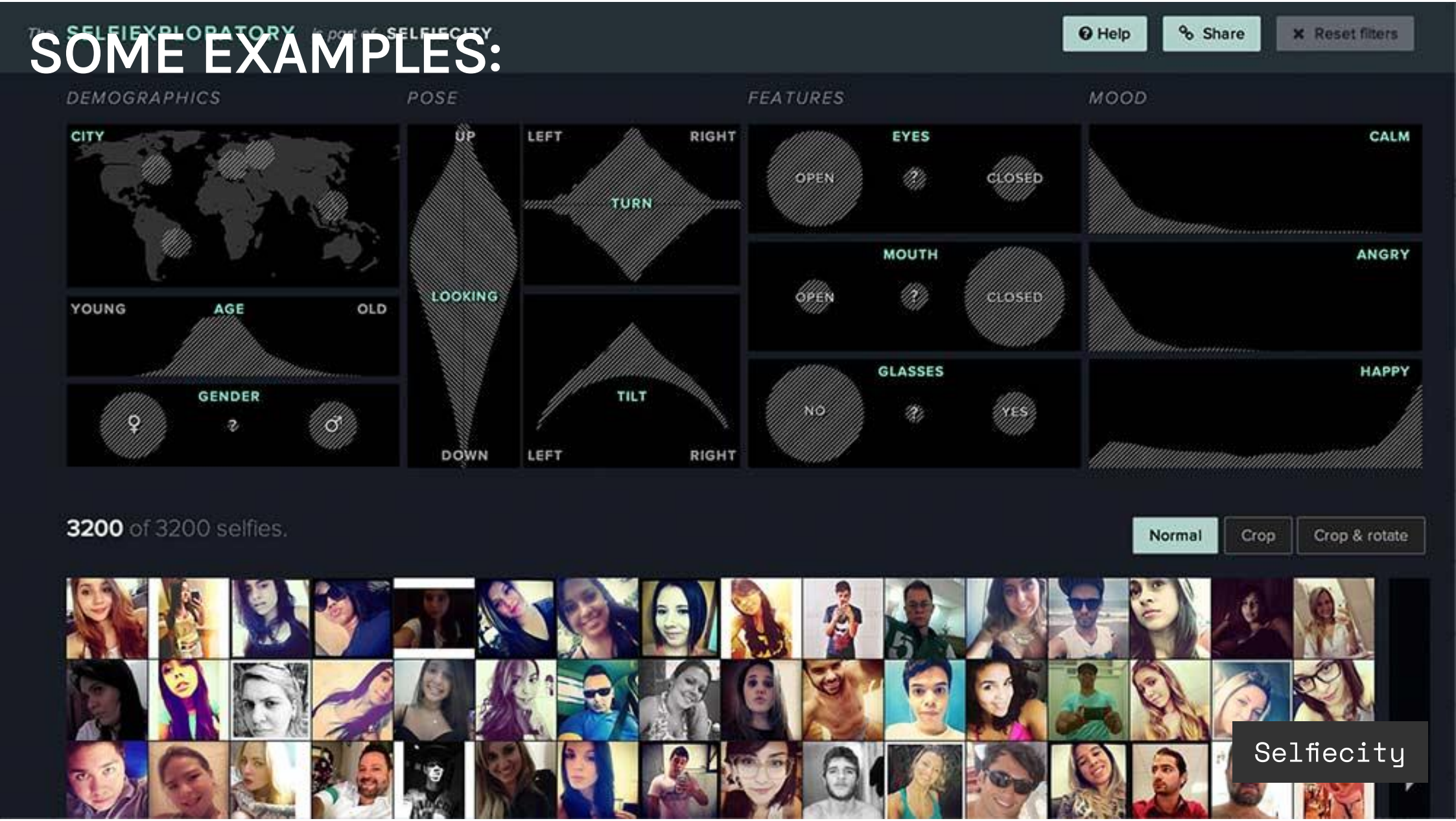
DR. DOMINIKUS BAUR
HTTPS://DO.MINIK.US

GRAPHICHUNTERS.NL

# DAY 1

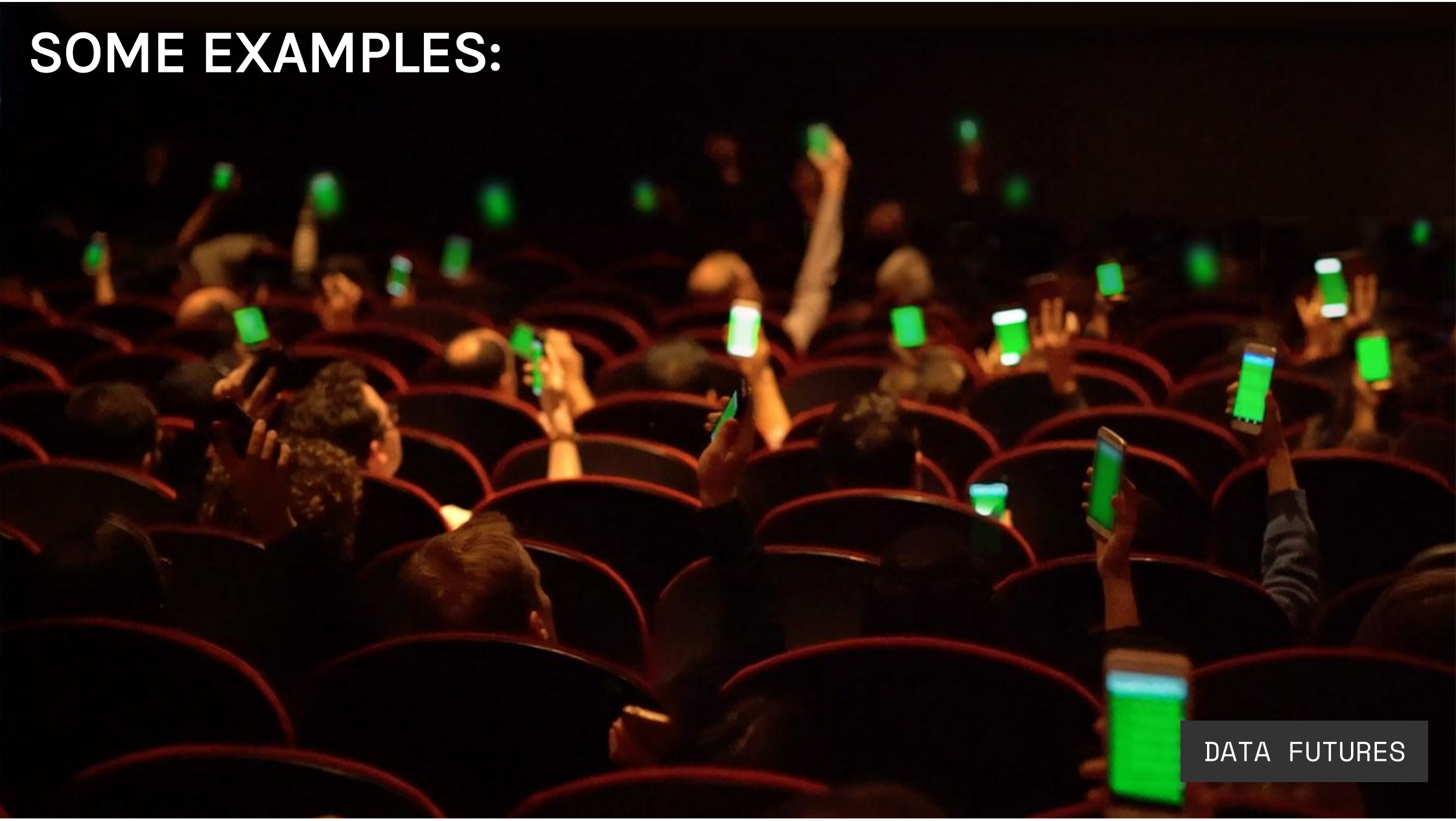
# ABOUT ME

# SOME EXAMPLES:



# SOME EXAMPLES:

# SOME EXAMPLES:



# WHAT ABOUT YOU?

# IN THIS WORKSHOP

# PHILOSOPHY:

- Give you pointers to materials
- Make you aware of certain concepts
- Teach you the tools to teach yourself
- Learn together

# Teach Yourself Programming in Ten Years

**Peter Norvig**

## Why is everyone in such a rush?

Walk into any bookstore, and you'll see how to *Teach Yourself Java in 24 Hours* alongside endless variations offering to teach C, SQL, Ruby, Algorithms, and so on in a few days or hours. The Amazon advanced search for [title: teach, yourself, hours, since: 2000 and found 512 such books. Of the top ten, nine are programming books (the other is about bookkeeping). Similar results come from replacing "teach yourself" with "learn" or "hours" with "days."

The conclusion is that either people are in a big rush to learn about programming, or that programming is somehow fabulously easier to learn than anything else. Felleisen *et al.* give a nod to this trend in their book *How to Design Programs*, when they say "Bad programming is easy. *Idiots* can learn it in *21 days*, even if they are *dummies*." The Abtruse Goose comic also had their take.

Let's analyze what a title like *Teach Yourself C++ in 24 Hours* could mean:

- **Teach Yourself:** In 24 hours you won't have time to write several significant programs, and learn from your successes and failures with them. You won't have time to work with an experienced programmer and understand what it is like to live in a C++ environment. In short, you won't have time to learn much. So the book can only be talking about a superficial familiarity, not a deep understanding. As Alexander Pope said, a little learning is a dangerous thing.

- **C++:** In 24 hours you might be able to learn some of the syntax of C++ (if you already know another language), but you couldn't learn much about how to use the language. In short, if you were, say, a Basic programmer, you could learn to write programs in the style of Basic using C++ syntax, but you couldn't learn what C++ is actually good (and bad) for. So what's the point? Alan Perlis once said: "A language that doesn't affect the way you think about prog
you have to learn a tiny b
Processing) because you
task. But then you're not
task.

- **in 24 Hours:** Unfortunately, this is not enough, as the next section shows.

## Translations

Thanks to the following authors, translations of this page are available in:

Arabic (Mohamed A. Yahya)

العربية

Bulgarian (Boyko Bantchev)

Chinese (Xiaogang Guo)

Croatian (Tvrtko Bedekovic)

Peter Norvig: Teach Yourself Programming In 10 Years

# PHILOSOPHY:

Learn the processes, not the commands.

# SCHEDULE

# SCHEDULE

do.minik.us: Workshop Info

# LET'S GET STARTED

# INTRO TO DATAVIS

# ONLINE DATA VISUALIZATION

# FLASH VIS

## Moritz Stefaner: Tag Maps


Java™

# JAVA VIS

http://wefeelfine.org/


OECD
Better Life
Index
Create Your Better Life Index
How's life?

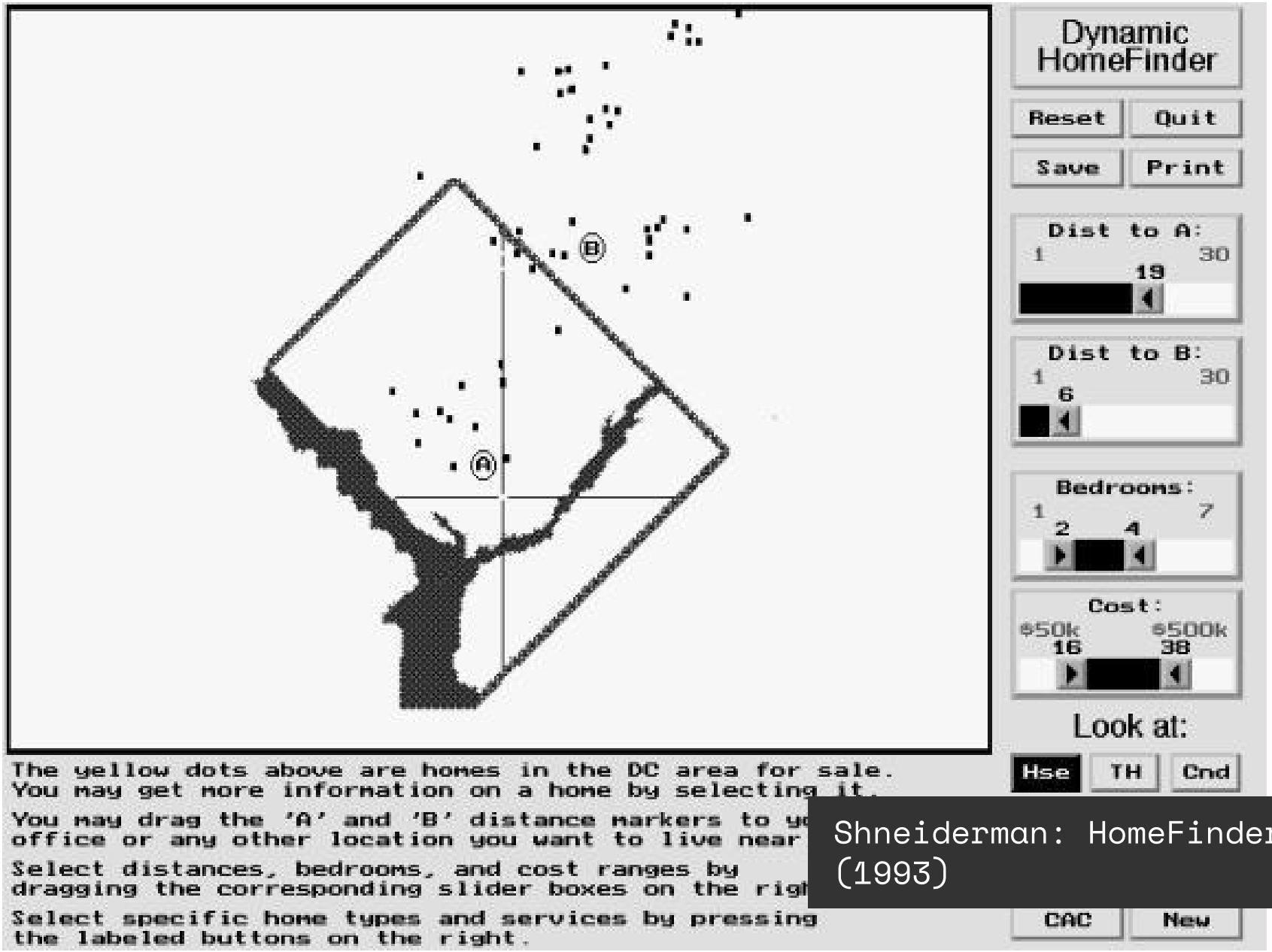
# DATA VISUALIZATION

# DATA VISUALIZATION

*The use of computer-supported, interactive, visual representations of abstract data to amplify cognition.*
*Stuart Card*

# DATA VISUALIZATION

In academia:
Information Visualization, Scientific Visualization, Visual Analytics, ...




Shneiderman: HomeFinder (1993)

# Powerful Visualization Technique to achieve dream goals

8 Comments



Visualization | NLP Techniques

Attracting Money Wealth | Successful Money


Powerful Money Affirmations

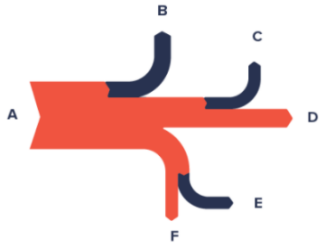
# DATA VISUALIZATION

Read more:

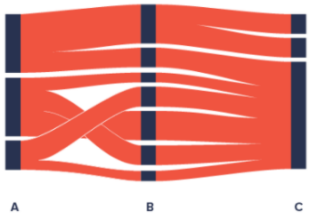
- eagereyes
- FlowingData
- Information Is Beautiful Awards

# CHART TYPES

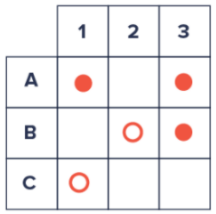
## Sankey Diagram




## Alluvial Diagram

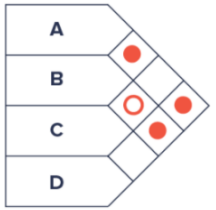



## Matrix Diagram



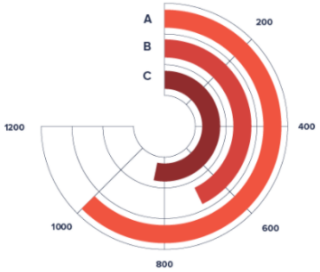

## Donut Chart


## Matrix Diagram (Roof Shaped)

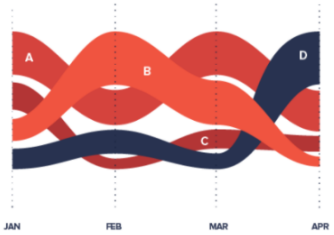



## Radial Histogram

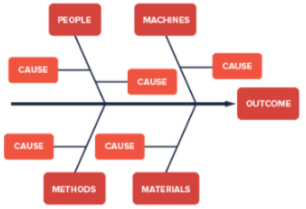

## Radial Bar Chart




## Sorted Stream Graph




## Fishbone Diagram

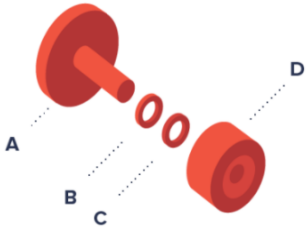



## Pictorial fraction chart


## Isoline Map

## Flow Map

## Exploded View Drawing




## Arc Diagram


## Pictorial Stacked Chart


http://datavizproject.com/

# CHART TYPES

## Nathan Yau: One Dataset, Visualized 25 Ways

# VISUAL VARIABLES

# Marks

- Points

- Lines

- Areas

# Channels

- Position
  - Horizontal
  - Vertical
  - Both

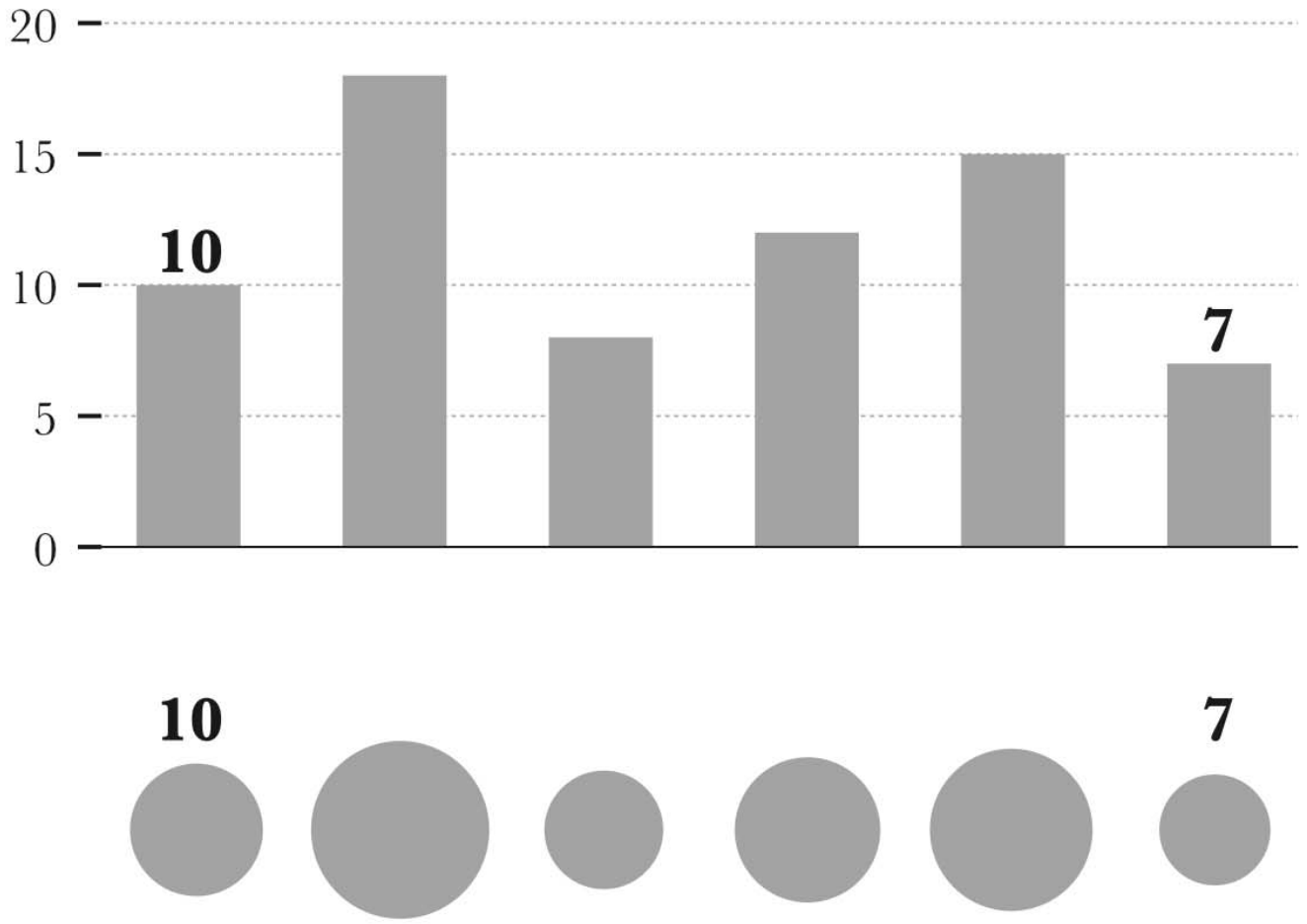
- Color

- Shape

- Tilt

- Size
  - Length
  - Area
  - Volume

Tamara Munzner: Visualization Analysis and Design

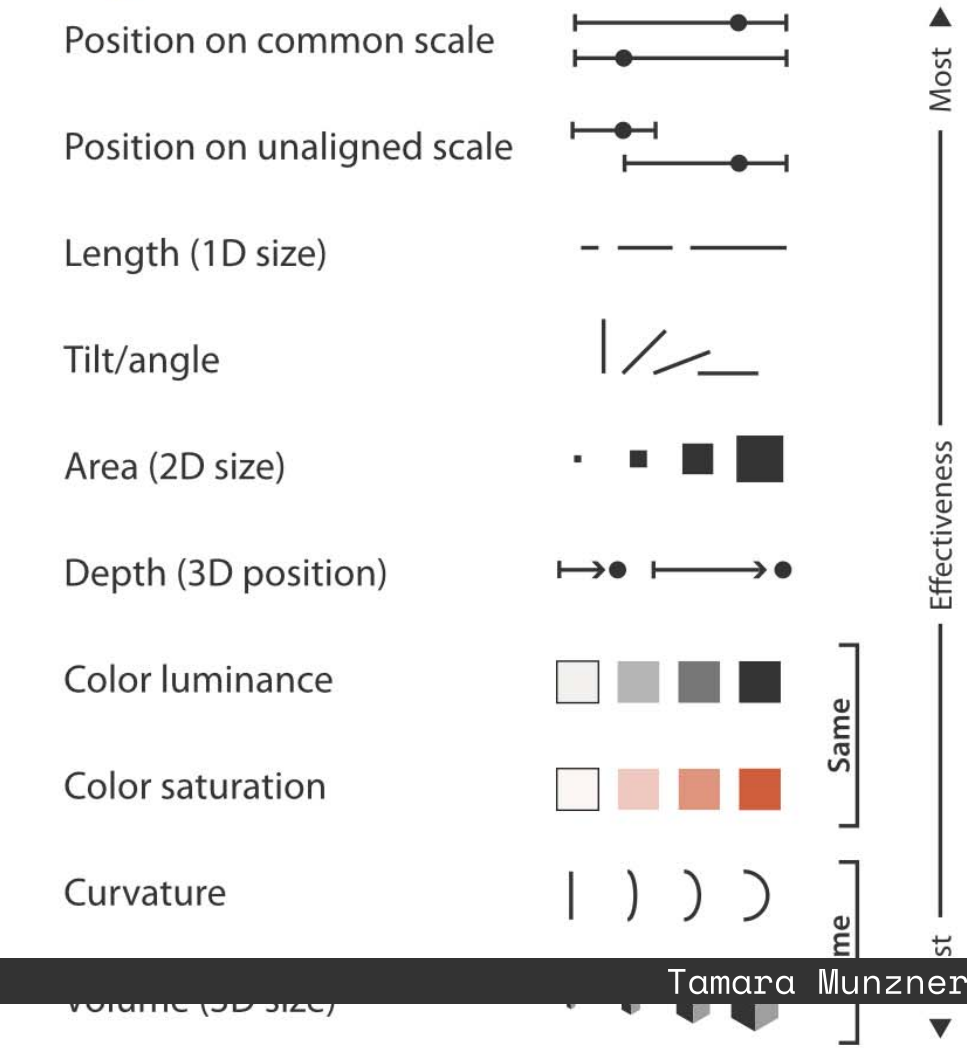



Tamara Munzner: Visualization Analysis and Design

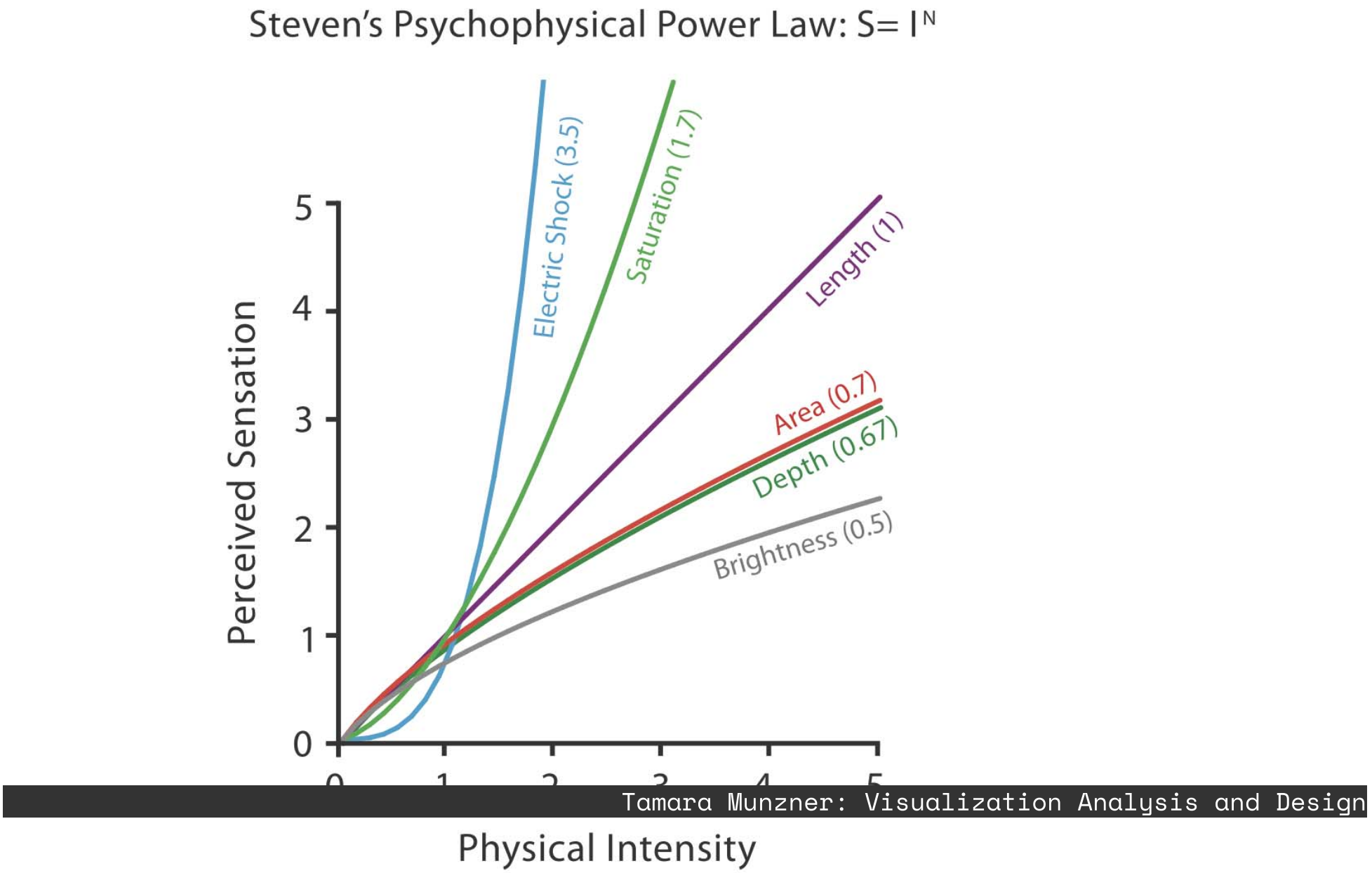
## Encoding magnitude

**Channels:** Expressiveness Types and Effectiveness Ranks

**Magnitude Channels: Ordered Attributes**




Tamara Munzner: Visualization Analysis and Design

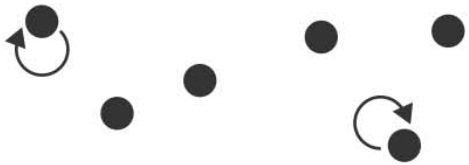

Steven's Psychophysical Power Law: S= I^N
Perceived Sensation
Physical Intensity
0
1
2
3
4
5
Electric Shock (3.5)
Saturation (1.7)
Length (1)
Area (0.7)
Depth (0.67)
Brightness (0.5)
Tamara Munzner: Visualization Analysis and Design

## Identity Channels: Categorical Attributes

Spatial region

Color hue

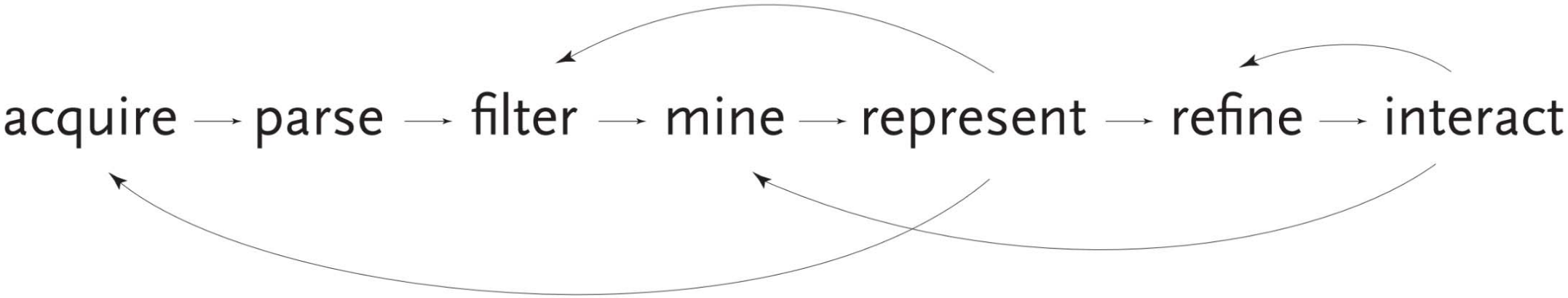
Motion



Shape

Tamara Munzner: Visualization Analysis and Design

# PROCESS




Ben Fry: Computational Information Design

# INTERACTION

# INTERACTION

*Overview first, zoom and filter, details on demand.*
*Ben Shneiderman*

# INTERACTION

TABLE 1: **Taxonomy of interactive dynamics for visual analysis**

| | |
|---|---|
| **Data & View Specification** | **Visualize** data by choosing visual encodings.<br>**Filter** out data to focus on relevant items.<br>**Sort** items to expose patterns.<br>**Derive** values or models from source data. |
| **View Manipulation** | **Select** items to highlight, filter, or manipulate them.<br>**Navigate** to examine high-level patterns and low-level detail.<br>**Coordinate** views for linked, multi-dimensional exploration.<br>**Organize** multiple windows and workspaces. |
| **Process & Provenance** | **Record** analysis histories for revisitation, review and sharing.<br>**Annotate** patterns to document findings.<br>**Share** views and annotations to enable collaboration.<br>**Guide** users through analysis tasks or stories. |

Heer, Shneiderman: Interactive Dynamics for Visual Analysis

# DATA(SET) TYPES

# DATA(SET) TYPES

Main dataset types of interest to us are:

- Tabular data (objects, attributes)
- Network data (nodes, links, attributes)

# DATA(SET) TYPES

These **attributes** of datasets can be further categorized into:

- **nominal/categorical**: no inherent order
- **ordinal**: order, but no numerical differences
- **continuous**: numerical differences

# DATA(SET) TYPES

## Nominal/Categorical:

Names
Countries
Colors
...

# DATA(SET) TYPES

## Ordinal:

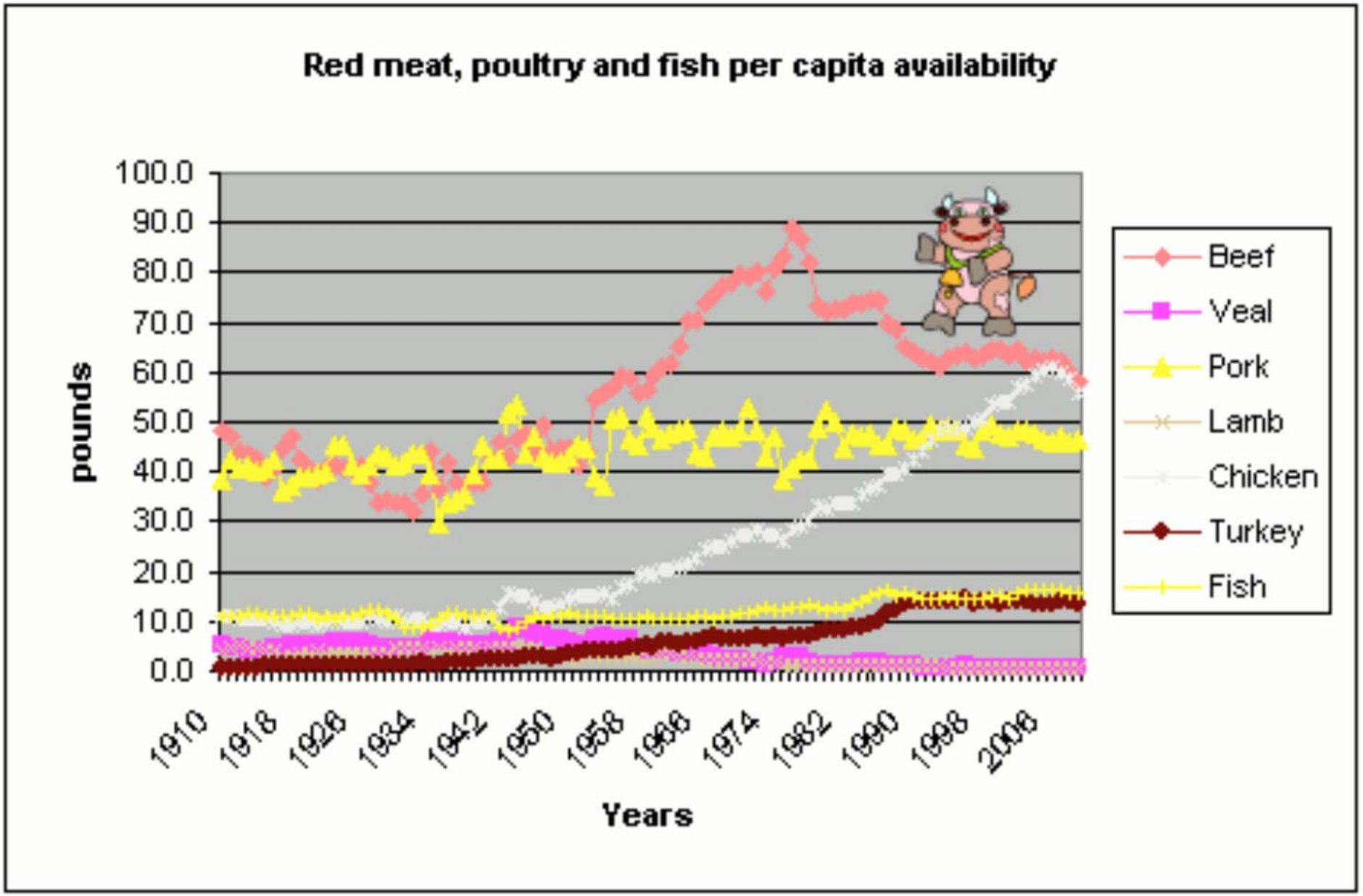
S, M, L, XL
Baby, Teen, Adult
Sports leagues
...

# DATA(SET) TYPES

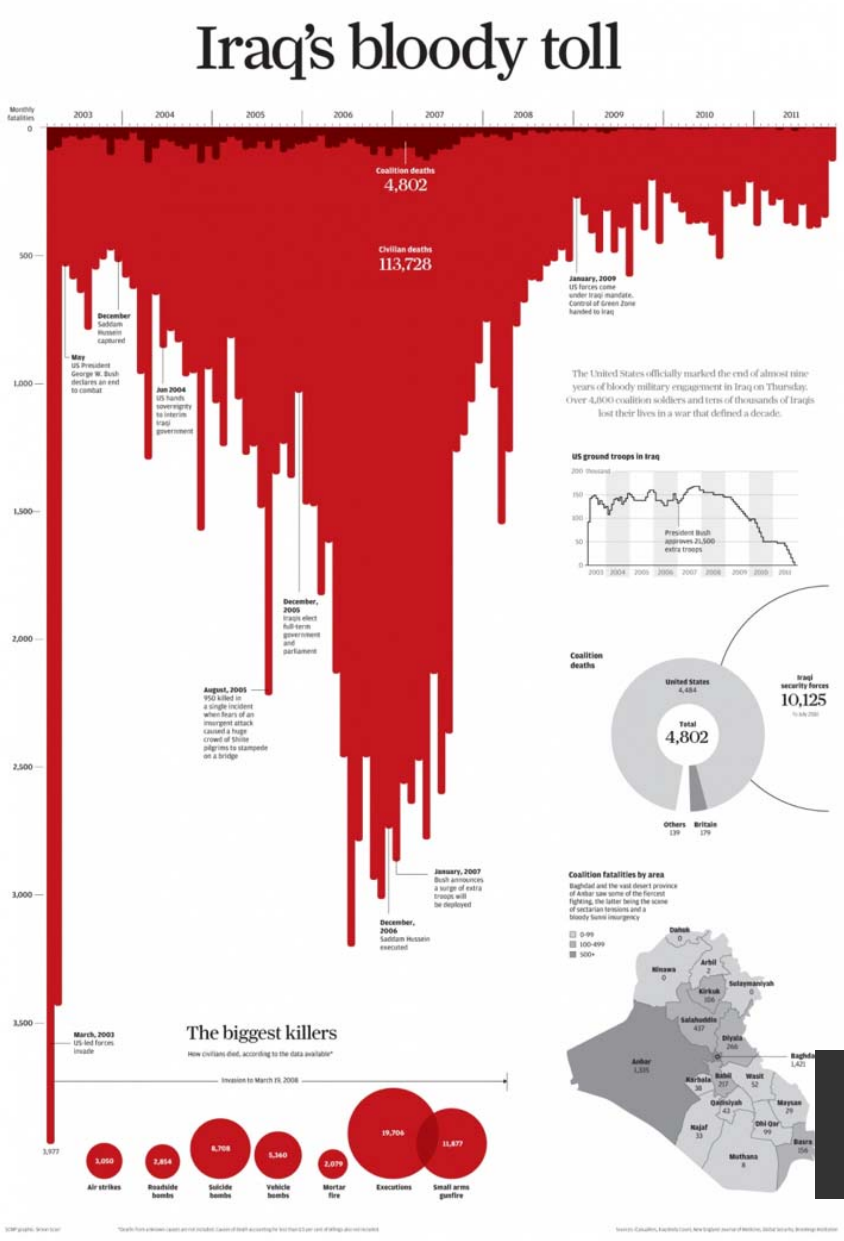
## Continuous:

Stock prices
Time
Body height/weight
...

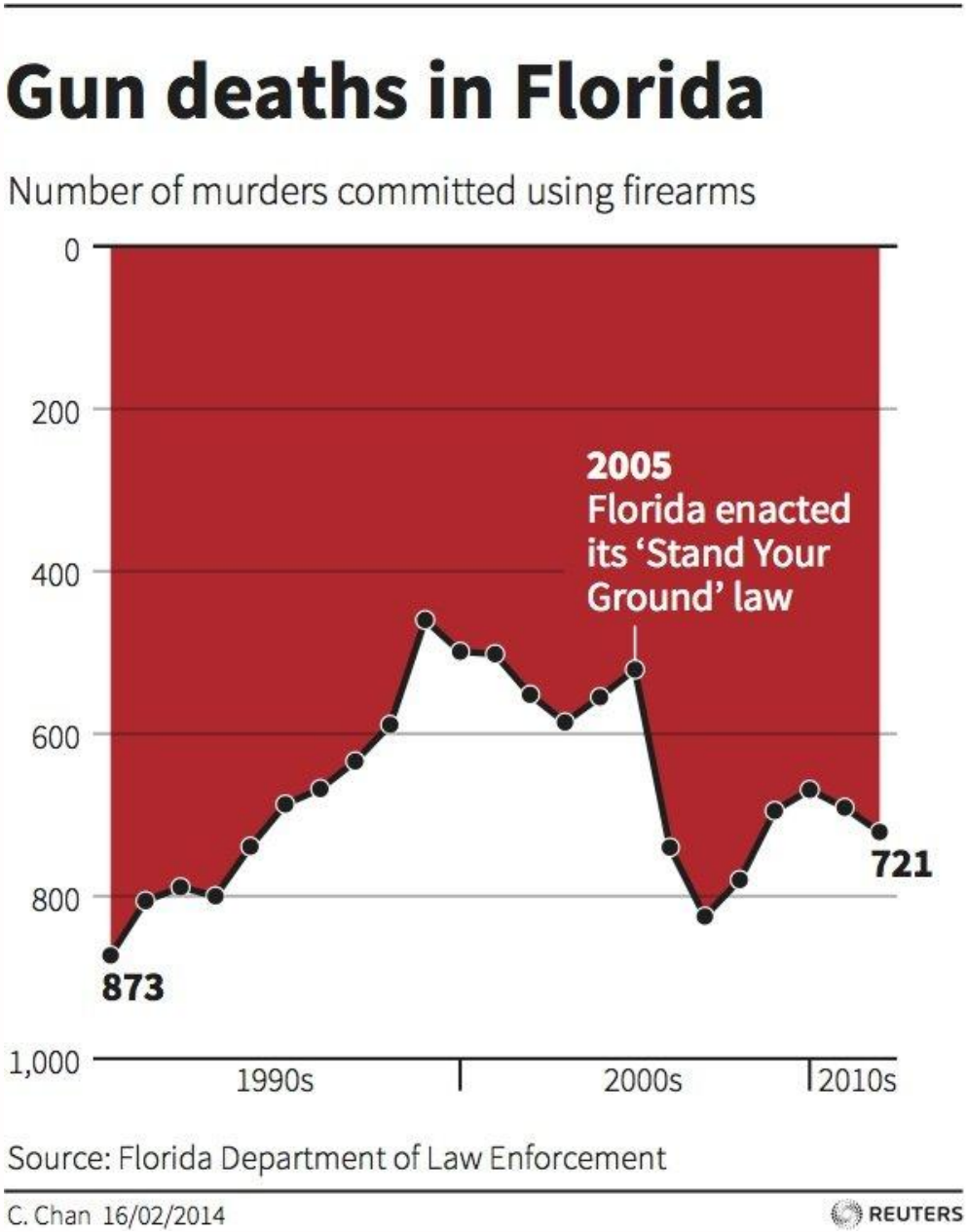
# DESIGN ASPECTS


Red meat, poultry and fish per capita availability
pounds
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
1910
1918
1926
1934
1942
1950
1958
1966
1974
1982
1990
1998
2006
Years
Beef
Veal
Pork
Lamb
Chicken
Turkey
Fish


RED MEAT, POULTRY AND FISH
per capita availability
pounds
100
75
50
25
0
Great Depression
89
Beef
Pork
Chicken
Fish
Turkey
1910
20
30
40
50
60
70
80
90
2000
2009
Years




SCMP: Iraq's Bloody Toll




Business Insider

# VISUAL HIERARCHY

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris convallis dui vel mauris congue bibendum. Etiam placerat felis eget malesuada eleifend. Quisque sed magna eget mi viverra pretium sit amet vehicula lectus. Proin nunc dolor, dignissim ut leo at, ultricies feugiat nulla. Duis luctus, dui eget rhoncus lobortis, urna velit congue dolor, eu fringilla leo j usto sit amet odio. Donec rhoncus, massa et tempus finibus, eros elit convallis quam, quis facilisis urna nisl vel lectus. Aenean tortor lacus, semper non erat in, ultrices vestibulum orci.

Vestibulum tincidunt ultrices erat. Cras at erat ac ipsum gravida varius. Donec vitae tincidunt est, ac posuere massa. Nulla tincidunt nec orci ut tristique. Duis vel molestie lectus. Curabitur pulvinar feugiat leo sit amet rhoncus. Nullam pharetra leo et urna efficitur, eget ullamcorper lectus congue. Maecenas eget mi in orci pharetra maximus. Donec fermentum turpis non nisi sagittis rhoncus. Curabitur sed felis scelerisque, condimentum magna id, porttitor augue. Integer lacinia maximus metus eget suscipit. Proin enim ex, finibus nec interdum eget, condimentum eget quam. Nunc eu purus eu nunc pellentesque convallis posuere in tellus. Sed sed ornare mauris. Nullam imperdiet augue ut faucibus auctor. Nunc eu mattis purus.

Aliquam ut lorem eu felis volutpat ultrices. Sed ultricies lectus ipsum, at laoreet neque consectetur sit amet. Sed laoreet malesuada nisl vitae mollis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam mattis sit amet dolor et cursus. Curabitur ac tortor maximus ante facilisis fringilla. Nullam sodales fringilla egestas. Nunc neque ipsum, hendrerit non urna eget, imperdiet consectetur orci. Nullam varius convallis consectetur. Maecenas fringilla, ipsum at imperdiet rhoncus, metus urna varius arcu, et tincidunt diam dui sed dolor. Duis in vehicula urna, sit amet pellentesque ligula. Nam vitae velit at arcu luctus tempus. Donec nibh neque, interdum placerat vulputate venenatis, pulvinar id velit.

Donec scelerisque vulputate aliquam. Vivamus blandit, magna ac luctus facilisis, urna purus condimentum metus, non rhoncus enim ligula quis mauris. Pellentesque aliquet, tellus ac maximus rhoncus, eros sapien pretium sapien, quis dictum nunc risus non turpis. Aliquam erat volutpat. Aliquam vestibulum metus et nisi tristique venenatis. Proin posuere felis purus, eget maximus erat rutrum non. Nunc eu enim eget tellus viverra vulputate.

Morbi efficitur porttitor ante, in viverra risus rutrum et. Vestibulum dictum enim in posuere viverra. Aenean nibh tellus, vulputate eget magna ut, condimentum sagittis diam. Sed vel tristique lorem. Sed at interdum eros, eget pulvinar dolor. Sed id purus euismod, interdum lacus et, gravida lorem. Praesent sed pulvinar massa, sed vulputate est. Proin in nisi ac tortor varius viverra. Pellentesque a tincidunt risus. Nulla mattis, tellus ut feugiat pellentesque, felis diam ullamcorper odio, ac egestas nisl libero id purus. Nam aliquet, dui ac condimentum eleifend, purus lacus blandit risus, sit amet rhoncus lorem tellus quis arcu. Curabitur eros libero, tincidunt a lacus sed, efficitur ullamcorper turpis. Aenean nisl felis, egestas vel varius in, porttitor ut enim. Quisque sit amet vehicula eros. Donec vitae nisl nulla. Aenean sollicitudin elementum neque, ut vehicula purus iaculis vitae.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris convallis dui vel mauris congue bibendum. Etiam placerat felis eget malesuada eleifend. Quisque sed magna eget mi viverra pretium sit amet vehicula lectus. Proin nunc dolor, dignissim ut leo at, ultricies feugiat nulla. Duis luctus, dui eget rhoncus lobortis, urna velit congue dolor, eu fringilla leo j usto sit amet odio. Donec rhoncus, massa et tempus finibus, eros elit convallis quam, quis facilisis urna nisl vel lectus. Aenean tortor lacus, semper non erat in, ultrices vestibulum orci.

Vestibulum tincidunt ultrices erat. Cras at erat ac ipsum gravida varius. Donec vitae tincidunt est, ac posuere massa. Nulla tincidunt nec orci ut tristique. Duis vel molestie lectus. Curabitur pulvinar feugiat leo sit amet rhoncus. Nullam pharetra leo et urna efficitur, eget ullamcorper lectus congue. Maecenas eget mi in orci pharetra maximus. Donec fermentum turpis non nisi sagittis rhoncus. Curabitur sed felis scelerisque, condimentum magna id, porttitor augue. Integer lacinia maximus metus eget suscipit. Proin enim ex, finibus nec interdum eget, condimentum eget quam. Nunc eu purus eu nunc pellentesque convallis posuere in tellus. Sed sed ornare mauris. Nullam imperdiet augue ut faucibus auctor. Nunc eu mattis purus.

Aliquam ut lorem eu felis volutpat ultrices. Sed ultricies lectus ipsum, at laoreet neque consectetur sit amet. Sed laoreet malesuada nisl vitae mollis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam mattis sit amet dolor et cursus. Curabitur ac tortor maximus ante facilisis fringilla. Nullam sodales fringilla egestas. Nunc neque ipsum, hendrerit non urna eget, imperdiet consectetur orci. Nullam varius convallis consectetur. Maecenas fringilla, ipsum at imperdiet rhoncus, metus urna varius arcu, et tincidunt diam dui sed dolor. Duis in vehicula urna, sit amet pellentesque ligula. Nam vitae velit at arcu luctus tempus. Donec nibh neque, interdum placerat vulputate venenatis, pulvinar id velit.

Donec scelerisque vulputate aliquam. Vivamus blandit, magna ac luctus facilisis, urna purus condimentum metus, non rhoncus enim ligula quis mauris. Pellentesque aliquet, tellus ac maximus rhoncus, eros sapien pretium sapien, quis dictum nunc risus non turpis. Aliquam erat volutpat. Aliquam vestibulum metus et nisi tristique venenatis. Proin posuere felis purus, eget maximus erat rutrum non. Nunc eu enim eget tellus viverra vulputate.

Morbi efficitur porttitor ante, in viverra risus rutrum et. Vestibulum dictum enim in posuere viverra. Aenean nibh tellus, vulputate eget magna ut, condimentum sagittis diam. Sed vel tristique lorem. Sed at interdum eros, eget pulvinar dolor. Sed id purus euismod, interdum lacus et, gravida lorem. Praesent sed pulvinar massa, sed vulputate est. Proin in nisi ac tortor varius viverra. Pellentesque a tincidunt risus. Nulla mattis, tellus ut feugiat pellentesque, felis diam ullamcorper odio, ac egestas nisl libero id purus. Nam aliquet, dui ac condimentum eleifend, purus lacus blandit risus, sit amet rhoncus lorem tellus quis arcu. Curabitur eros libero, tincidunt a lacus sed, efficitur ullamcorper turpis. Aenean nisl felis, egestas vel varius in, porttitor ut enim. Quisque sit amet vehicula eros. Donec vitae nisl nulla. Aenean sollicitudin elementum neque, ut vehicula purus iaculis vitae.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris convallis dui vel mauris congue bibendum. Etiam placerat felis eget malesuada eleifend. Quisque sed magna eget mi viverra pretium sit amet vehicula lectus. Proin nunc dolor, dignissim ut leo at, ultricies feugiat nulla. Duis luctus, dui eget rhoncus lobortis, urna velit congue dolor, eu fringilla leo j usto sit amet odio. Donec rhoncus, massa et tempus finibus, eros elit convallis quam, quis facilisis urna nisl vel lectus. Aenean tortor lacus, semper non erat in, ultrices vestibulum orci.

Vestibulum tincidunt ultrices erat. Cras at erat ac ipsum gravida varius. Donec vitae tincidunt est, ac posuere massa. Nulla tincidunt nec orci ut tristique. Duis vel molestie lectus. Curabitur pulvinar feugiat leo sit amet rhoncus. Nullam pharetra leo et urna efficitur, eget ullamcorper lectus congue. Maecenas eget mi in orci pharetra maximus. Donec fermentum turpis non nisi sagittis rhoncus. Curabitur sed felis scelerisque, condimentum magna id, porttitor augue. Integer lacinia maximus metus eget suscipit. Proin enim ex, finibus nec interdum eget, condimentum eget quam. Nunc eu purus eu nunc pellentesque convallis posuere in tellus. Sed sed ornare mauris. Nullam imperdiet augue ut faucibus auctor. Nunc eu mattis purus.

Aliquam ut lorem eu felis volutpat ultrices. Sed ultricies lectus ipsum, at laoreet neque consectetur sit amet. Sed laoreet malesuada nisl vitae mollis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nam mattis sit amet dolor et cursus. Curabitur ac tortor maximus ante facilisis fringilla. Nullam sodales fringilla egestas. Nunc neque ipsum, hendrerit non urna eget, imperdiet consectetur orci. Nullam varius convallis consectetur. Maecenas fringilla, ipsum at imperdiet rhoncus, metus urna varius arcu, et tincidunt diam dui sed dolor. Duis in vehicula urna, sit amet pellentesque ligula. Nam vitae velit at arcu luctus tempus. Donec nibh neque, interdum placerat vulputate venenatis, pulvinar id velit.

**Donec scelerisque vulputate aliquam. Vivamus blandit, magna ac luctus facilisis, urna purus condimentum metus, non rhoncus enim ligula quis mauris.** Pellentesque aliquet, tellus ac maximus rhoncus, eros sapien pretium sapien, quis dictum nunc risus non turpis. Aliquam erat volutpat. Aliquam vestibulum metus et nisi tristique venenatis. Proin posuere felis purus, eget maximus erat rutrum non. Nunc eu enim eget tellus viverra vulputate.

Morbi efficitur porttitor ante, in viverra risus rutrum et. Vestibulum dictum enim in posuere viverra. Aenean nibh tellus, vulputate eget magna ut, condimentum sagittis diam. Sed vel tristique lorem. Sed at interdum eros, eget pulvinar dolor. Sed id purus euismod, interdum lacus et, gravida lorem. Praesent sed pulvinar massa, sed vulputate est. Proin in nisi ac tortor varius viverra. Pellentesque a tincidunt risus. Nulla mattis, tellus ut feugiat pellentesque, felis diam ullamcorper odio, ac egestas nisl libero id purus. Nam aliquet, dui ac condimentum eleifend, purus lacus blandit risus, sit amet rhoncus lorem tellus quis arcu. Curabitur eros libero, tincidunt a lacus sed, efficitur ullamcorper turpis. Aenean nisl felis, egestas vel varius in, porttitor ut enim. Quisque sit amet vehicula eros. Donec vitae nisl nulla. Aenean sollicitudin elementum neque, ut vehicula purus iaculis vitae.

**FRANKENSTEIN** A WORLD PREMIERE FROM NATIONAL THEATRE LIVE

***"I followed nature into her lair, and stripped her of her secrets! I brought torrents of light to a darkening world! Is that wrong?"***

# *Slowly I learnt the ways of humans: how to ruin, how to hate, how to debase, how to humiliate.*

AND AT THE FEET OF MY MASTER I LEARNT THE HIGHEST OF HUMAN SKILLS, THE SKILL NO OTHER CREATURE OWNS: I FINALLY LEARNT HOW TO LIE.

*17 March:* Benedict Cumberbatch (Creature), Jonny Lee Miller (Victor)*

*24 March:* Jonny Lee Miller (Creature), Benedict Cumberbatch (Victor)*

Childlike in his innocence but grotesque in form, Frankenstein's bewildered creature is cast out into a hostile universe by his horror-struck maker. Meeting with cruelty wherever he goes, the friendless Creature, increasingly desperate and vengeful, determines to track down his creator and strike a terrifying deal.

***"All I ask is the possibility of love!"***

Urgent concerns of scientific responsibility, parental neglect, cognitive development and the nature of good and evil are embedded within this thrilling and deeply disturbing classic gothic tale.

Mary Shelley's Frankenstein is adapted for the stage by Nick Dear and realised by Danny Boyle in his return to the theatre after winning the Academy Award for best director for Slumdog Millionaire.

For the first time ever, National Theatre Live will broadcast two separate performances of a production. *Throughout the run of Frankenstein at the National Theatre, Benedict Cumberbatch and Jonny Lee Miller are alternating the roles of Victor Frankenstein and the Creature. Audiences in cinemas will have the chance to see both combinations, with two broadcasts a week apart.

*A NEW PLAY BY NIC*
*NOVEL BY MARY SH*

Smashing Magazine

# WHITESPACE

# Why whitespace matters

AUTHOR:
Paul Boag

DATE:
15 February 2011

CATEGORY:
Interface Design

READING TIME:
7 minutes

NUMBER OF SHARES (CLICK TO SHARE):
380 people

Designers love it, website owners want to fill it. Whitespace seems to be one of the most controversial aspects of design. Why then is it so important and how can we ensure it is maintained?

Whitespace is a fundamental building block of good design. Its one of the first thing any visual designer is taught. However, to many website owners it is simply a waste of space that could be used to better promote their messages, services or products.

In this post I aim to explain why whitespace matters and how to keep whitespace in a design without compromising business objectives. However, before I can do that

Boagworld

# GRID


Vermont
Symphony
Orchestra
Winter
2007
Season
Aaron Copland
The Tender Land
January 2007
01/12/07
Middlebury College
Center for the Arts
8:00 pm
01/19/07
Johnson State College
Dibden Center for the Arts
8:00 pm
01/26/07
Lyndon State College
Alexander Twilight Theater
8:00 pm
Eric Satie
Gymnopedie 1, 2
February 2007
02/03/07
Johnson State College
Dibden Center for the Arts
8:00 pm
02/10/07
Castleton State College
Fine Arts Center
8:00 pm
02/17/07
Middlebury College
Center for the Arts
8:00 pm

DI* How to Succeed Designer Resources Showcase Design Insights

# Using Layout Grids Effectively

There are two main types of layouts: vertical or landscape. There are also only two types of grids. One that has an even number of columns and one that has an odd number of columns. An experienced designer knows that a specific style of design can only be achieved by an odd number of columns, or alternatively, by using an even number of columns. Illustrated below are common examples of layouts using basic layout grids. Learning to choose the right grid for your design is crucial to its success.

## Here are examples of basic vertical layout grids.


Designers Insights

# INTRO TO WEBDEV

# BASICS

HTML

CSS

Javascript

# LOCAL WEBSERVER

# LOCAL WEBSERVER

```
> python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

# WHY A LOCAL WEBSERVER?

- Using a server-side language
- Security

More info on local webservers

# LOCAL WEBSERVER

Set up your local webserver, show a Hello World index.html on http://localhost:8080

# ONLINE DATASETS FORMATS

# ONLINE DATA COMES IN SEVERAL TYPICAL FORMATS:

## CSV (Comma-separated values):

```
name,age
Bob, 44
Carl, 17
```

# ONLINE DATA COMES IN SEVERAL TYPICAL FORMATS:

**DSV (Delimiter-separated values):** Variants of CSV come with tab (TSV), semicolons, colons, asterisks... as separators. They're all subsumed under the moniker "Delimiter-separated".

# ONLINE DATA COMES IN SEVERAL TYPICAL FORMATS:

## JSON (Javascript Object Notation):

```
[
{"name": "Bob", "age": "44"},
{"name": "Carl", "age": "17"}
    ]
```

# CHROME DEVTOOLS

# CHROME DEVTOOLS

- Insight into DOM-structure
- Insight into Javascript (output, debugging)
- Network metrics
- Performance
- Test devices/form factors

# CHROME DEVTOOLS

Build on your index.html from before. Add a <script> tag, console.log() and debugger. Add an error and 'Pause on exceptions'.

# LET'S GO EXPLORING:

## Pudding: Film Dialogue

# LET'S GO EXPLORING:

## Kim Albrecht: Science Paths

# LET'S GO EXPLORING:

FiveThirtyEight: Who would win the presidency today?

# LET'S GO EXPLORING:

## The Guardian: The Counted

# ONLINE GRAPHICS

# ONLINE GRAPHICS

There's three (main) technologies to create graphics on the web:

- 1. SVG (Scalable Vector Graphics)
- 2. Canvas
- 3. WebGL

# 0. HTML + CSS

Plain old * * tags can be styled quite a lot using CSS:
*background-color* makes them colored bars,
*border-radius* turns them into circles,
*transform* can be used to freely translate/rotate/scale them
...

# HTML + CSS:

One example:

2Q17 - Election visualization w/ Google News Lab

# 1. SVG

- Vector-based (so: explicitly declared shapes)
- consists of HTML-tags
- can be used in <img>-tags
- d3.js's main graphics technology

# 2. CANVAS

- Bitmap/raster based (pixel values)
- non-scalable
- opaque to the browser (and the devtools)
- usually faster than SVG

# 3. WEBGL

- Accelerated graphics for the web
- non-scalable
- opaque to the browser (and the devtools)
- difficult to develop for (in its raw form)
- the fastest of the three

# GREAT RESOURCE FOR JAVASCRIPT FUNCTIONS: MDN WEB DOCS

Arrays

Objects

Strings

(Global Objects)

# Can I use foreach ? Settings

2 results found

## # ECMAScript 5 - OTHER

Global 93.92% + 3.63% = 97.55%

Full support for the ECMAScript 5 specification. Features include `Function.prototype.bind`, Array methods like `indexOf`, `forEach`, `map` & `filter`, Object methods like `defineProperty`, `create` & `keys`, the `trim` method on Strings and many more.

Current aligned | Usage relative | Date relative | Show all

| IE | Edge * | Firefox | Chrome | Safari | Opera | iOS Safari * | Opera Mini * | Android Browser * | Chrome for Android |
|---|---|---|---|---|---|---|---|---|---|
| | | | 49 | | | | | | |
| | | 52 | 60 | | | 10.2 | | | |
| | 15 | 55 | 61 | 10.1 | | 10.3 | | 4.4 | |
| 11 | 16 | 56 | 62 | 11 | 48 | 11 | 1 all | 56 | 61 |
| | | 57 | 63 | TP | 49 | | | | |
| | | 58 | 64 | | 50 | | | | |
| | | 59 | 65 | | | | | | |

Notes | Sub-features (2) | Known issues (0) | Resources (4) | Feedback

https://caniuse.com

README.md

# Front-End Checklist

chat on gitter | Front-End Checklist followed | contributors 56 | tech stack | license CC0

The **Front-End Checklist** is an exhaustive list of all elements you need to have / to test before launching your site / HTML page to production.

It is based on Front-End developers' years of experience, with the additions coming from some other open-source checklists.

*Help to share the Front-End Checklist by voting and recommending on Product Hunt*


## Table of Contents



```
https://github.com/thedaviddias/Front-End-Checklist
```

# SIMPLE ONLINE CHARTS

# PUBLIC DATASETS

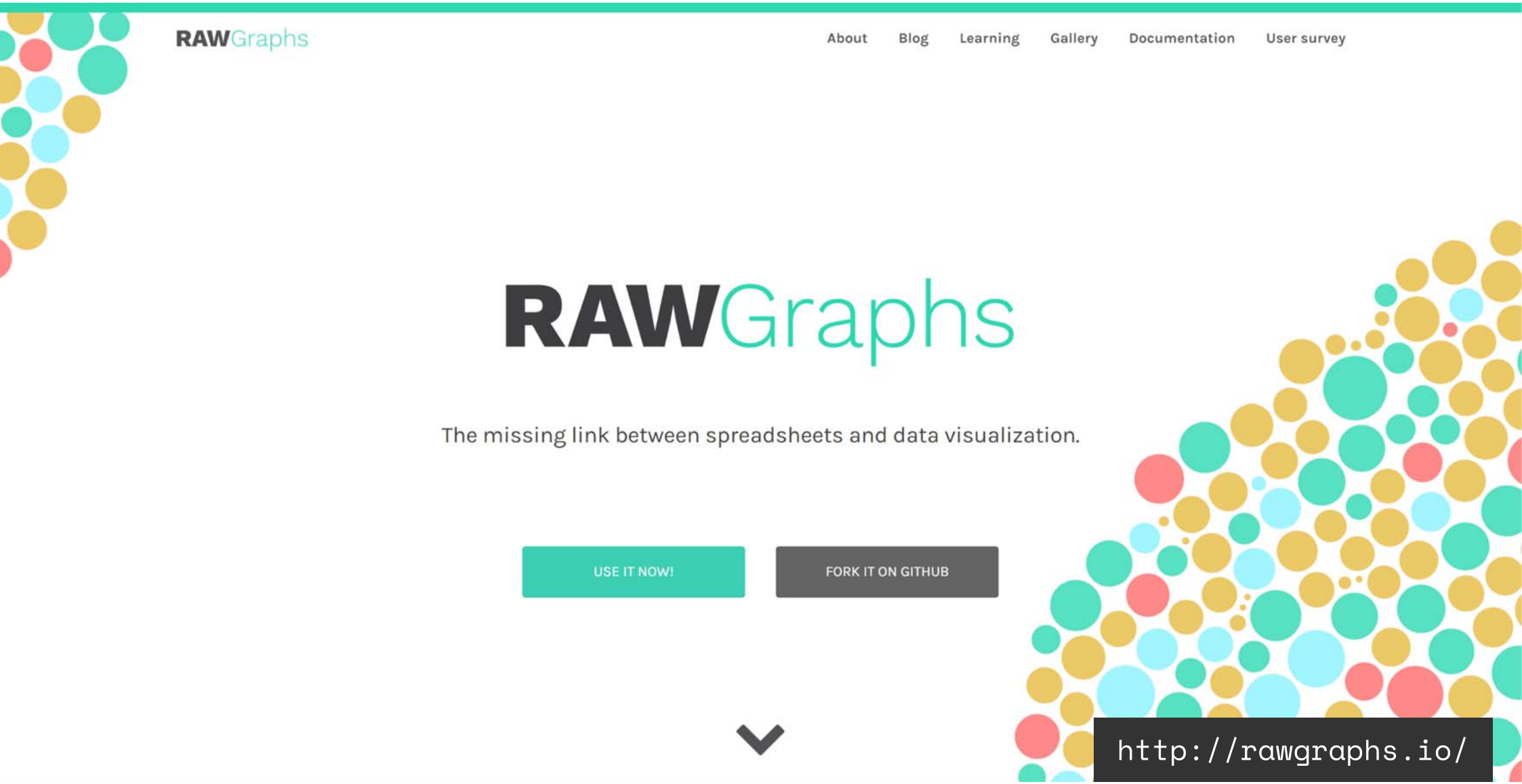
Awesome Public Datasets (github)

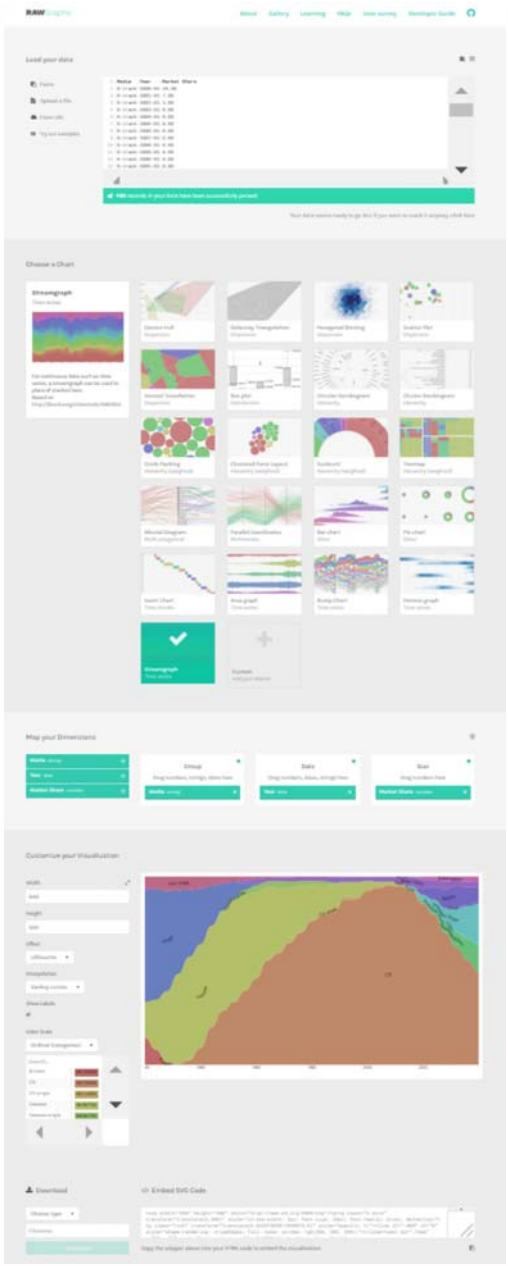
Kaggle Datasets

Cool Datasets (twitter)

EU OpenData Portal

# RAWGRAPHS

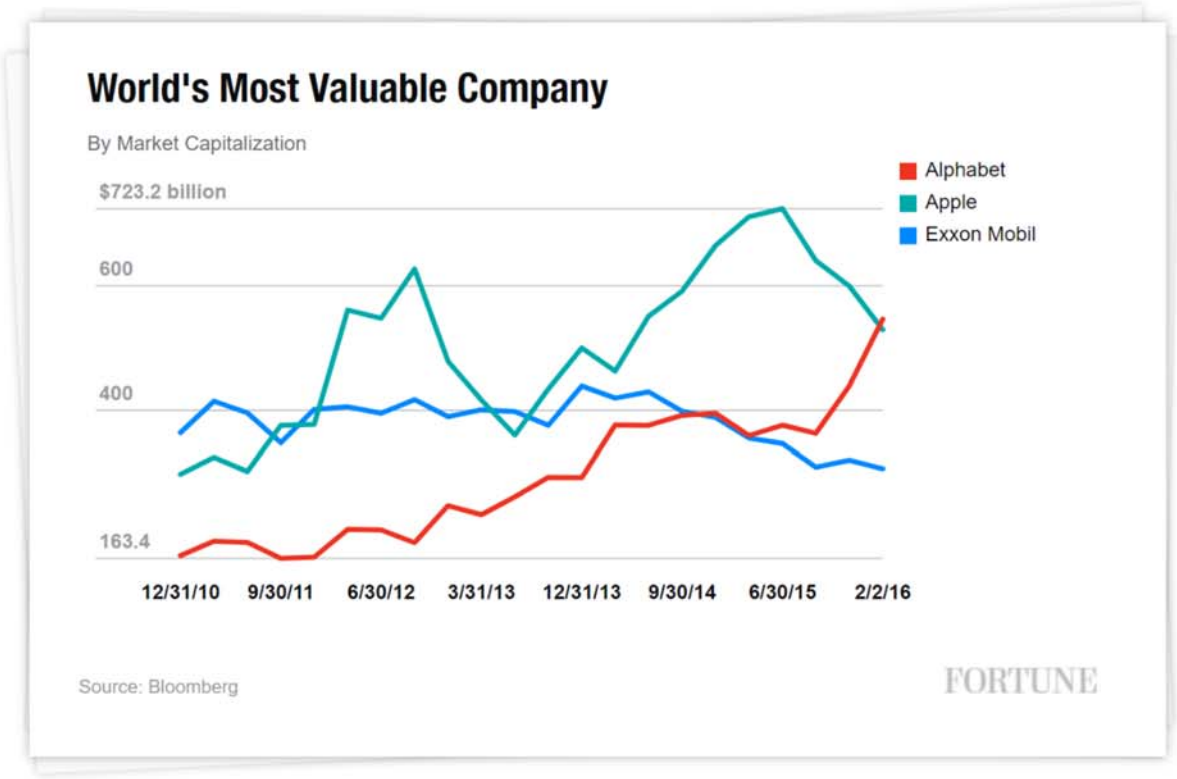

RAWGraphs
About
Blog
Learning
Gallery
Documentation
User survey
RAWGraphs
The missing link between spreadsheets and data visualization.
USE IT NOW!
FORK IT ON GITHUB
http://rawgraphs.io/



http://rawgraphs.io/

# DATAWRAPPER

# With Datawrapper you add the power of data to your stories.




published in **Fortune Magazine**, created with Datawrapper

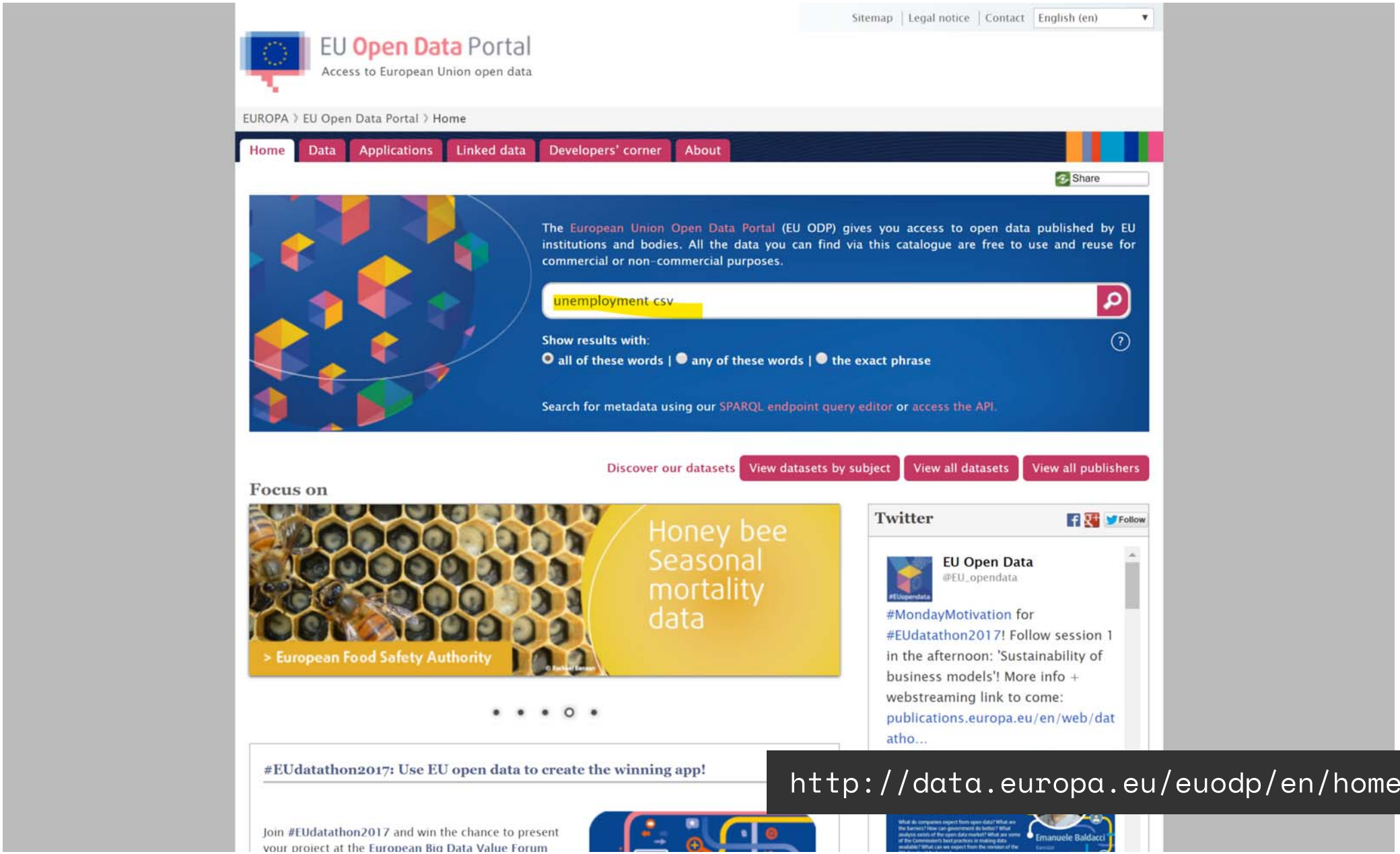
- Line chart
- Bar chart
- Stacked bar chart
- Map
- Donut
- Table

Want more charts? Check out the best ones in our Gallery!

http://datawrapper.de/

CREATE YOUR OWN CHART

# LET'S TRY SOME REAL DATA...


Sitemap | Legal notice | Contact | English (en)
EU Open Data Portal
Access to European Union open data
EUROPA › EU Open Data Portal › Home
Home
Data
Applications
Linked data
Developers' corner
About
Share
The European Union Open Data Portal (EU ODP) gives you access to open data published by EU institutions and bodies. All the data you can find via this catalogue are free to use and reuse for commercial or non-commercial purposes.
unemployment csv
Show results with:
all of these words | any of these words | the exact phrase
Search for metadata using our SPARQL endpoint query editor or access the API.
Discover our datasets
View datasets by subject
View all datasets
View all publishers
Focus on
Honey bee Seasonal mortality data
> European Food Safety Authority
Twitter
Follow
EU Open Data
@EU_opendata
#MondayMotivation for #EUdatathon2017! Follow session 1 in the afternoon: 'Sustainability of business models'! More info + webstreaming link to come: publications.europa.eu/en/web/datatho...
#EUdatathon2017: Use EU open data to create the winning app!
Join #EUdatathon2017 and win the chance to present your project at the European Big Data Value Forum
Emanuele Baldacci
http://data.europa.eu/euodp/en/home

# DATAVIZ LIBRARIES

# Awesome Charting

A curated list of chart and dataviz resources that developers may find useful. Focused on relevant and currently active JavaScript charting libraries for different use cases. Ordered alphabetically in each category.

*Inspired by the Awesome thing.*

awesome build passing

Table of Contents



https://github.com/zingchart/awesome-charting

# PROBLEMS WITH DATAVIZ LIBRARIES:

- Flexibility
- Ugly/non-functional defaults
- Longevity
- **Documentation**

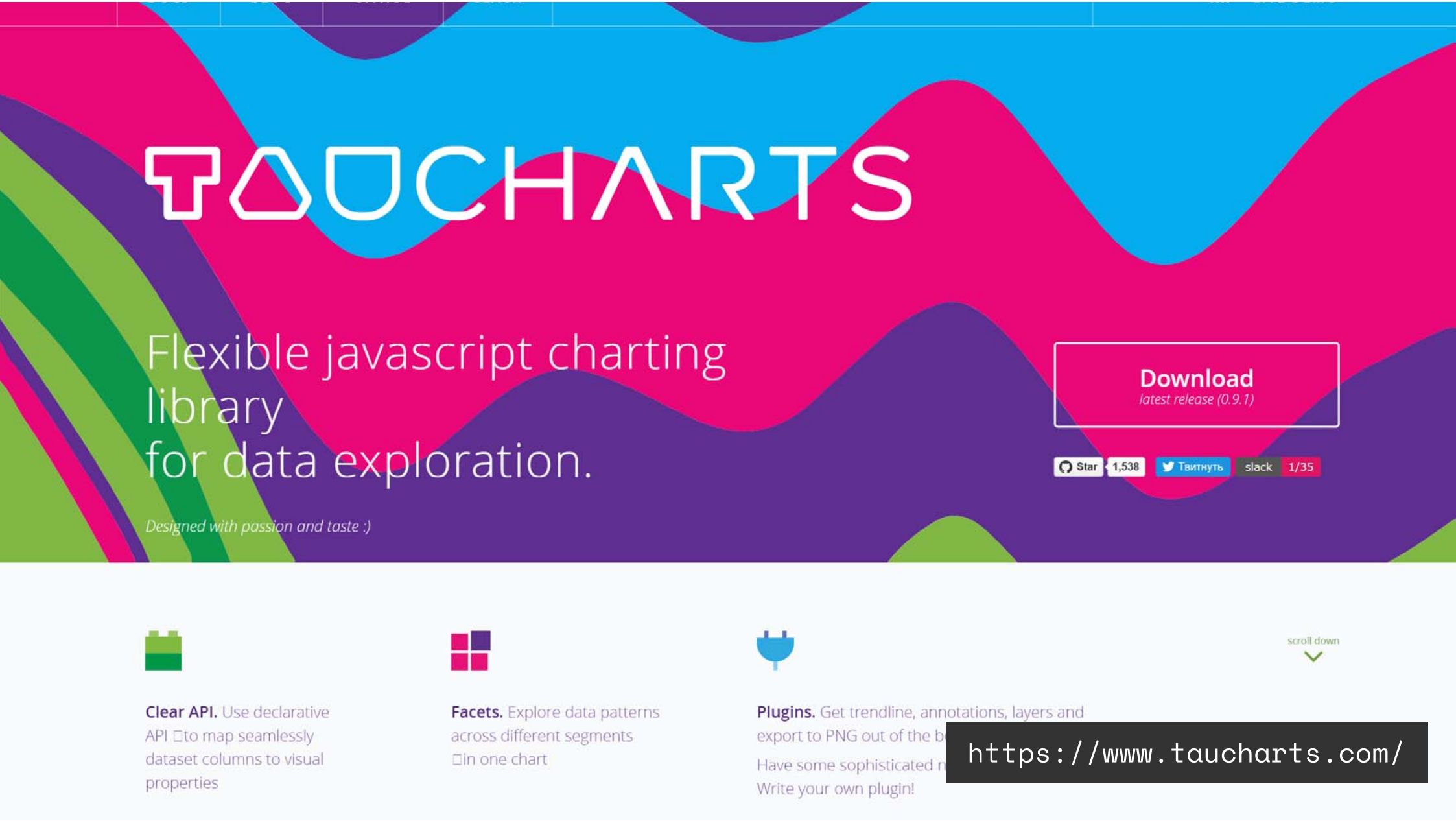
# CODE EXAMPLES

# CODE EXAMPLES

Since we're starting with the code now: you can find all the examples on github:

https://github.com/dominikus/online-data-visualization

# TAUCHARTS


TAUCHARTS
Flexible javascript charting library for data exploration.
Designed with passion and taste :)
Download
latest release (0.9.1)
Star 1,538
Твитнуть
slack 1/35
Clear API. Use declarative API to map seamlessly dataset columns to visual properties
Facets. Explore data patterns across different segments in one chart
Plugins. Get trendline, annotations, layers and export to PNG out of the b
Have some sophisticated r
Write your own plugin!
scroll down
https://www.taucharts.com/

# TAUCHARTS BASICS

```
<script
 src="https://cdn.jsdelivr.net/d3js/3.5.17/d3.min.j
 charset="utf-8"></script>
<script
 src="https://cdn.jsdelivr.net/npm/taucharts@1/buil
 type="text/javascript"></script>

<link rel="stylesheet" type="text/css"
 href="https://cdn.jsdelivr.net/npm/taucharts@1/bui
```

# TAUCHARTS BASICS II

```
var chart = new tauCharts.Chart({
      data: defData,
      type: 'bar',
      x: 'date',
      y: 'count',
      color: 'type'
  });

  chart.renderTo('#chart');
```


LET ME SHOW YOU
MY POKEMANS

# FETCH API

# FETCH API

Easy way to get remote (or local!)
data into your script.
Replaces the old XMLHttpRequest API.

MDN: Fetch API

# FETCH API

## Basic syntax (text):

```
fetch(remoteURL)
  .then(function(result){
    return result.text();
  })
  .then(function(parsedResult){
    // do something with the text...
  });
```

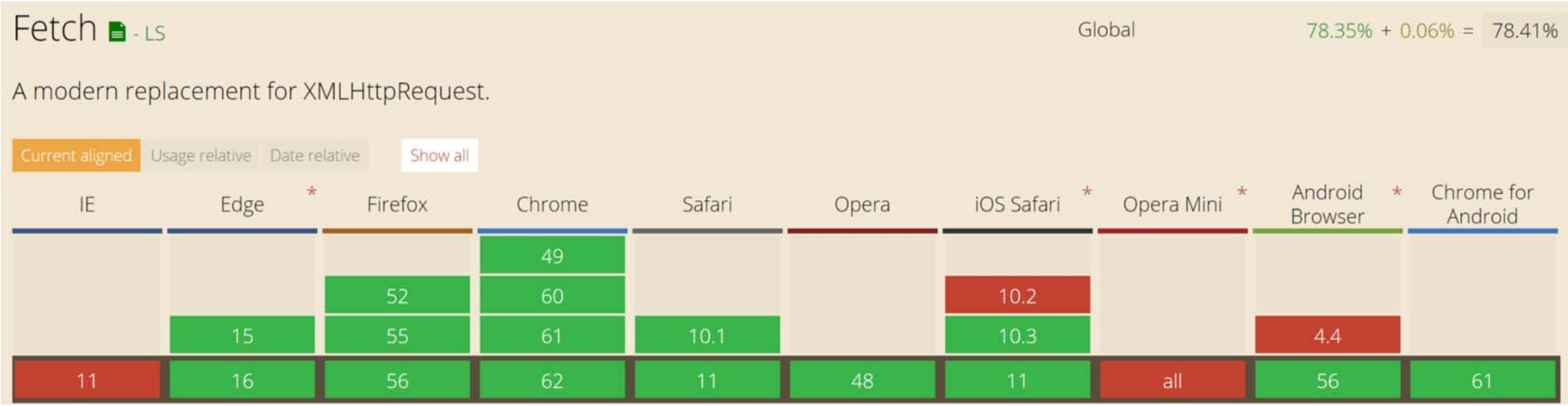
# FETCH API

## Basic syntax (JSON):

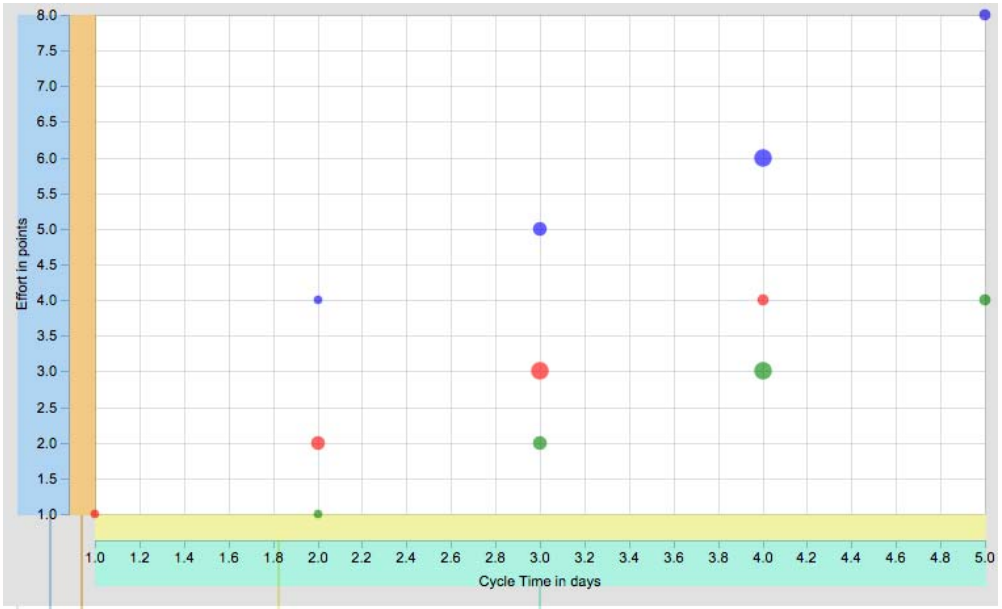
```
fetch(remoteURL)
  .then(function(result){
    return result.json();
  })
  .then(function(parsedResult){
    // do something with the JSON...
  });
```

# FETCH API

Widely supported. If you need IE support, use polyfill.

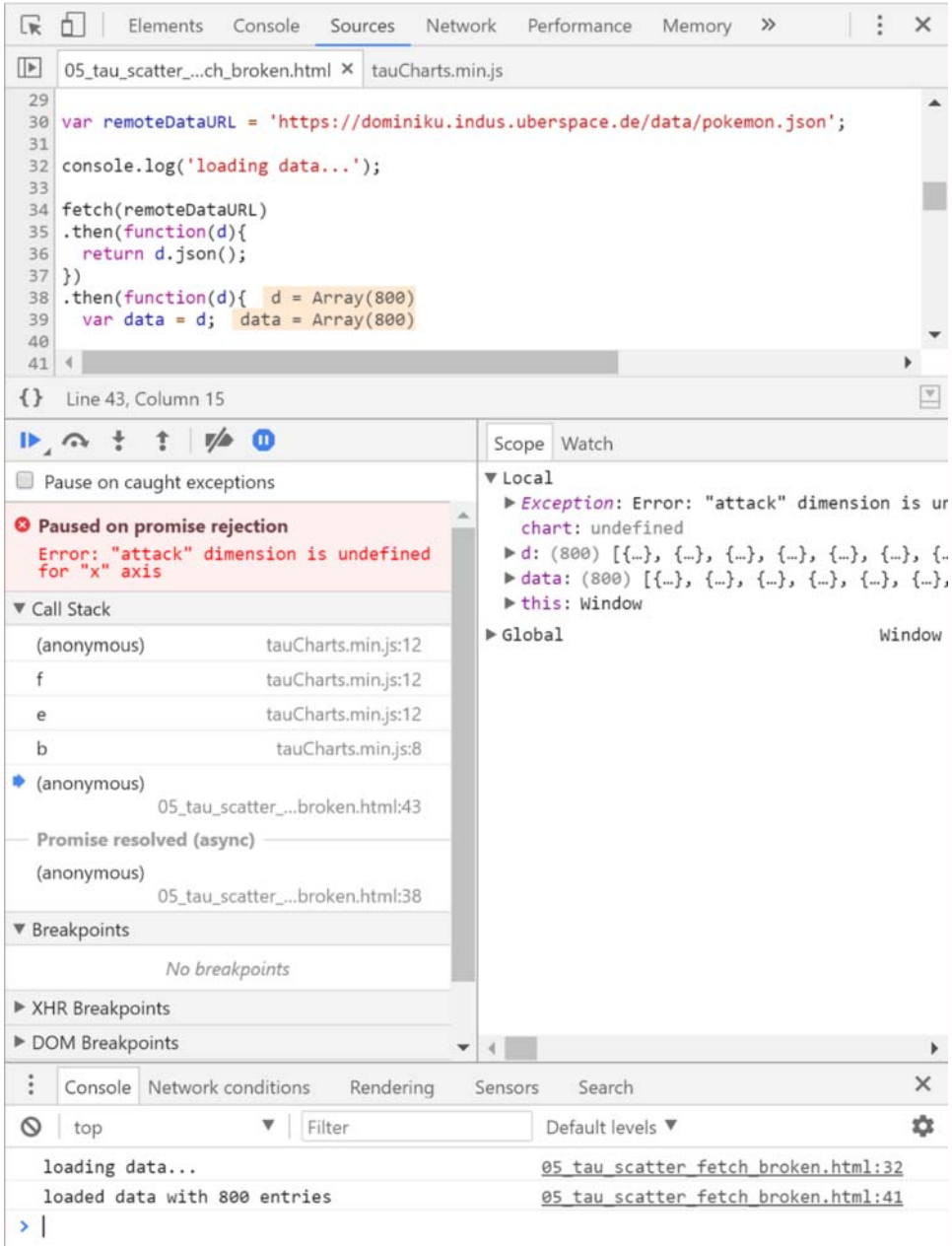



MDN: Fetch API




```
var chart = new tauChart.Chart({
    guide: {
        x: {
            padding: 20,
            label: {text: 'Cycle Time in days', padding: 35}
        },
        y: {
            padding: 20,
            label: {text: 'Effort in points', padding: 40}
        },
        padding: {b: 70, l: 70, t: 10, r: 10},
        showGridLines: 'xy',
        color: {
            brewer: ['color-red', 'color-green', 'color-blue']
        }
    },
    data: defData,
    type: 'scatterplot',
    x: 'cycleTime',
    y: 'effort',
    color: 'team',
    size: 'count'
});
```

https://api.taucharts.com/basic/guide.html


Elements
Console
Sources
Network
Performance
Memory
05_tau_scatter_...ch_broken.html
tauCharts.min.js
var remoteDataURL = 'https://dominiku.indus.uberspace.de/data/pokemon.json';
console.log('loading data...');
fetch(remoteDataURL)
.then(function(d){
return d.json();
})
.then(function(d){ d = Array(800)
var data = d; data = Array(800)
Line 43, Column 15
Pause on caught exceptions
Paused on promise rejection
Error: "attack" dimension is undefined for "x" axis
Call Stack
(anonymous) tauCharts.min.js:12
f tauCharts.min.js:12
e tauCharts.min.js:12
b tauCharts.min.js:8
(anonymous) 05_tau_scatter_...broken.html:43
Promise resolved (async)
(anonymous) 05_tau_scatter_...broken.html:38
Breakpoints
No breakpoints
XHR Breakpoints
DOM Breakpoints
Scope
Watch
Local
Exception: Error: "attack" dimension is ur
chart: undefined
d: (800) [{…}, {…}, {…}, {…}, {…}, {…}, {…
data: (800) [{…}, {…}, {…}, {…}, {…}, {…},
this: Window
Global Window
Console
Network conditions
Rendering
Sensors
Search
top
Filter
Default levels
loading data... 05_tau_scatter_fetch_broken.html:32
loaded data with 800 entries 05_tau_scatter_fetch_broken.html:41

# DATA WRANGLING IN JS

# DATA WRANGLING IN JS

Unfortunately, often necessary...

# Let's start with a simple example:

```
var myArray = [2,4,5,10];
/* how to get a new array with
each number multiplied by 3? */
```

# Something more interesting:

```
var myArray = [2,4,5,10];
/* how to get a new array with
each EVEN number divided by 2? */
```

# Something even more interesting:

```
var myArray = [2,4,5,10];
/* create an array of objects containing:
   NUM: original_number,
   SQUARED: original_number_squared
*/
```

# TWO MAIN PROGRAMMING STYLES:

**Imperative**:
How to do something

**Declarative**:
What to do (usually associated with a more Functional Programming style)

More info on imperative vs declarative in JS

# JAVASCRIPT ARRAY FUNCTIONS

# JAVASCRIPT ARRAY FUNCTIONS

**sort:**

sorts an array

# JS: SORT

```
console.log([4,2,3,1].sort());
```

```
=> [1,2,3,4]
```

# JAVASCRIPT ARRAY FUNCTIONS

**sort:**

sorts an array, also works with a compare function (a,b). If the compare function returns

< 0 => a > b,

= 0 => a = b,

> 0 => a < b

# JS: SORT

```
console.log([4,2,3,1].sort(
    function(a, b){
        return b - a;
    }
));
```

```
=> [4,3,2,1]
```

# JAVASCRIPT ARRAY FUNCTIONS

**forEach:**
traverses an array, performs given function with each element as parameter

# JS: FOREACH

```
[1,2,3,4].forEach(
function(d){
console.log(d * 2);
});
```

=>
2
4
6
8

# JAVASCRIPT ARRAY FUNCTIONS

**map:**

performs function with each element as parameter, returns the result as an array

# JS: MAP

```
var roots = [1,2,3,4].map(
function(d){
  return Math.sqrt(d);
});
console.log(roots);
```

```
=> [1, 1.414.., 1.732.., 2]
```

# JAVASCRIPT ARRAY FUNCTIONS

**filter:**
performs function with each element as parameter, return array with all elements where function returns true.

# JS: FILTER

```
var evens = [1,2,3,4].filter(
function(d){
  return d % 2 === 0;
});
console.log(evens);
```

```
=> [2, 4]
```

# JAVASCRIPT ARRAY FUNCTIONS

**reduce:**

performs functions with an "accumulator" and each element as parameters. Return the value of the accumulator.

# JS: REDUCE

```
var sum = [1,2,3,4].reduce(
    function(acc, val){
        return acc + val;
    },
    0);
console.log(sum);
```

```
=> 10 // == 1 + 2 + 3 + 4
```

# JAVASCRIPT OBJECT FUNCTIONS

# JAVASCRIPT OBJECT FUNCTIONS

**Object.keys:**

Returns all keys of an object as array.

# JS: OBJECT.KEYS

```
var obj = { 'a': 4, 'b': 12, 'c': 36 };
console.log(Object.keys(obj));
```

```
=> ['a', 'b', 'c']
```

# JAVASCRIPT OBJECT FUNCTIONS

**Object.values:**
Returns all values of an object as array.

# JS: OBJECT.VALUES

```
var obj = { 'a': 4, 'b': 12, 'c': 36 };
console.log(Object.values(obj));
```

```
=> [4, 12, 36]
```

# JAVASCRIPT CHAINING

# JAVASCRIPT CHAINING

Each of the previous functions returns the result of its computation.
So you can **chain** them to have the next function work with the previous one's results:

```
var result = arr.filter(...)
    .map(...)
    .reduce(...);
```

# ES6 ARROW FUNCTIONS

# ES6 ARROW FUNCTIONS

(ECMAScript 6), the latest, widely-distributed version of Javascript allows a short-hand writing for functions using arrows:

```
var myFunc = function(a){
    return a * 2;
}
// becomes:
var myFunc = (a) => { return a * 2; };
// or even:
var myFunc = a => a * 2;
```

# Arrow functions make declarative calls even shorter:

```
var arr = [1,5,14,28];

arr.filter(d => d % 2 === 0)
    .map(d => d * 2)
    .reduce((a, v) => a + v);

// => 84
```

# LODASH

# LODASH

Lodash is a helper library containing even more functions for various array and object manipulations.

Each function name starts with _. (underscore dot) and usually takes an array/object and a string/function as parameters.

# LODASH _.WITHOUT

```
_.without([2, 1, 2, 3], 1, 2);
// => [3]
```

# LODASH _.KEYBY

```
var array = [
{ 'dir': 'left', 'code': 97 },
{ 'dir': 'right', 'code': 100 }
];

_.keyBy(array, function(o) {
return String.fromCharCode(o.code);
});
// => { 'a': { 'dir': 'left', 'code': 97 },
//      'd': { 'dir': 'right', 'code': 100 } }

_.keyBy(array, 'dir');
// => { 'left': { 'dir': 'left', 'code': 97 },
```

```
//       'right': { 'dir': 'right', 'code': 100 } }
```

# LEARN MORE:

http://learnjsdata.com

# D3.JS

# Data-Driven Documents

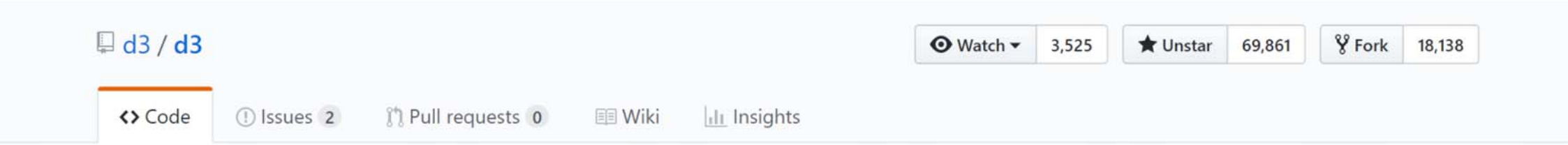
**D3.js** is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

See more examples.

Download the latest version (4.11.0) here:

- d3.zip

d3.js




https://github.com/d3/d3


https://twitter.com/mbostock

# D3.JS BASICS

From d3js.org:

*D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document.*

# D3.JS BASICS

Despite being sold as a data visualization library, at its core, d3.js is a general purpose library for manipulating the DOM - very similar to jQuery, mootools, ...

```
/* jQuery: */
$('h1.update').html('New headline!');
/* d3.js: */
d3.select('h1.update').html('New headline!');
```

# D3.JS BASICS

... but: it brings with it various additions, specifically geared towards datavis!

```
// define a color scale:
var scale = d3.scaleCategory10();
// recolor <p>s:
d3.selectAll('p')
  .style('background-color', function(d,i){
    return scale(i);
  });
```

# D3.JS BASICS

**Chaining**: d3.js encourages chaining functions. Most functions return some sensible value to keep working with.

```
d3.select('div#chart') // select <div>...
    .append('p') // add <p> ...
    .text('chart goes here'); // set its text
```

# D3.JS BASICS

**Getter/Setter**: most d3.js functions work as both getters and setters. If you leave out the parameter, you get the value back.

```
d3.select('p').text(); // get the text content of <
d3.select('p').text('hullo'); // set the text conte
                              // of <p>
```

# D3.JS BASICS

... plus: d3.js setter functions also take functions as parameters, making them very flexible. These functions take parameters *d* (the current data) and *i* (the number in the selection array.)

```
d3.selectAll('p').text(function(d,i){
  return 'node: ' + i + ' with data: ' + d;
});
```

# D3.JS BASICS

d3.js follows the **declarative approach** - it talks about the *What*, not the *How*.

# D3.JS BASICS

Also, d3.js recently underwent the transition from version 3 to 4 - with some breaking changes. Check each example that you find for the version number!

Irene Ros: d3 v4 - what's new?

# MORE RESOURCES:

d3.js API Reference

d3.js Tutorials Wiki

bl.ocks.org (massive collection of examples)

blockbuilder (live coding/search)

d3.js questions on StackOverflow

# d3.js on Slack

# D3.JS SELECTIONS

# D3.JS SELECTIONS

The core of d3.js is the ability to select one or more DOM elements. The syntax is based on the W3C Selectors API, just like CSS (so nothing new to learn!)

# D3.JS SELECTIONS

```
#foo          // <any id="foo">
foo           // <foo>
.foo          // <any class="foo">
[foo=bar]     // <any foo="bar">
foo bar       // <foo><bar/></foo>
```

# D3.JS SELECTIONS

## d3.select(SELECTOR):

*d3.select* selects the first element that fits the selector. If it doesn't find anything it stays empty (this is going to be important later!).

```
d3.select('body'); // <body>
d3.select('h1 i'); // first <i> in an <h1>
d3.select('svg#chart'); // first <svg> with id "cha
```

# D3.JS SELECTIONS

## d3.selectAll(SELECTOR):

Just like *d3.select*, but selects ALL elements that fit the selector.

```
d3.selectAll('div'); // selects all <div>s
d3.selectAll('p.text'); // selects all <p>s with cl
d3.selectAll('div#chart *'); // selects everything
```

# D3.JS SELECTIONS

*d3.selectAll().nodes()* gives you an array of selected elements. So to get the third element in the selection you can either use 'nth-child()' in the selector or:

```
d3.selectAll('p').nodes()[2]
```

# SELECTIONS

Since d3 selections work just like CSS selections, there's lot of training material on the web:

CSS Diner

w3cschools overview

# D3: THE JQUERY PARTS

# D3 DOM HELPERS

You can add/remove DOM elements using d3 with *append()* and *remove()*.

```
// add an <svg> tag to body:
d3.select('body').append('svg');
// remove all <div>s from the document:
d3.selectAll('div').remove();
```

# D3 DOM HELPERS

You can also change text content with *text()* and html content with *html()*.

```
// change the text in a <p>
d3.select('p').text('this is new text');
// add italic text to <p>:
d3.select('p').html('<i>yo</i>');
```

# D3 DOM HELPERS

Set/remove class attributes with *classed()*:

```
// check if the first <p> has class 'warn':
d3.select('p').classed('warn');
// set the classes 'important' and 'visible' for th
d3.select('div').classed('important visible', true)
```

# D3 DOM HELPERS

*d3.select(...).attr()* lets you access/change DOM attributes of the selected node:

```
// read the x1 attribute of a line:
var x1 = d3.select('line').attr('x1');
// ... then change it:
d3.select('line').attr('x1', x1 + 40);
```

# D3 DOM HELPERS

*d3.select(...).style()* lets you access/change the CSS attributes of the selected node:

```
// get the opacity of a <p>
var op = d3.select('p').style('opacity');
// change all <p>s opacities:
var psNum = d3.selectAll('p').size();
d3.selectAll('p').style('opacity', function(d,i){
  return i / (psNum - 1);
});
```

# D3.JS SELECTIONS

Let's try some selections:

JSFiddle: d3 selector playground

# D3 + SVG

# LET'S GO EXPLORING:

Guardian: How China's economic slowdown could weigh on the rest of the world

# LET'S EXPLORE SOME MORE:

Pudding: She giggles, he gallops

# SVG

# SVG

SVG (Scalable Vector Graphics) is the vector graphics format of the web.

(Vector graphics: lines, circles, shapes
Bitmap/raster graphics: pixels)

# SVG

SVG is human-readable and part of the HTML specification. SVG is defined using HTML-tags:

```
<svg width="150" height="150" xmlns="http://www.w3.
    <circle cx="50px" cy="50px" r="20px">
    </circle>
</svg>
```

JSFiddle: Basic SVG

# SVG

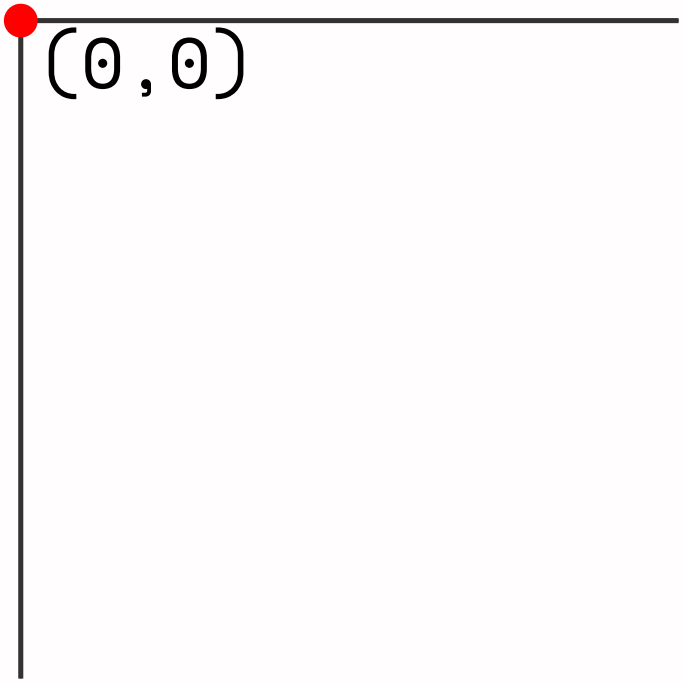
SVG comes with several basic shapes:

```
<svg width="550" height="100">
    <circle cx="50px" cy="50px" r="30px"></circle>
    <rect x="95px" y="20px" width="60px" height="60
    <ellipse cx="200px" cy="50px" rx="30px" ry="20p
</svg><
```

MDN: Basic SVG Shapes

# SVG

SVG's coordinate system starts in the top-left corner:



# D3 + SVG

# D3 + SVG

d3.js was originally built with SVG in mind. All its DOM manipulation works with SVG.

```
var svg = d3.select('body').append('svg')
  .attr('width', 300)
  .attr('height', 300);
```

# D3 + SVG

```
var svg = d3.select('body').append('svg')
  .attr('width', 300)
  .attr('height', 300);

svg.append('circle').classed('my-circle', true)
  .attr('cx', 30)
  .attr('cy', 30)
  .attr('r', 25);
```

# D3 + SVG

Let's draw some circles with d3!

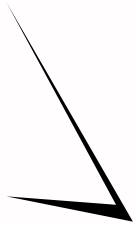
(Use code/d3-basics/template.html as a starting point)

# MORE SVG SHAPES

# SVG

**Polylines** and **polygons** have a single *points* attribute, containing a list of coordinates:

```
<polyline points="0 0, 65 120, 0 115, 75 130"/>
```



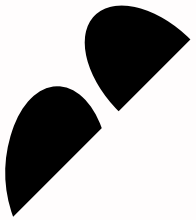
MDN: Basic SVG Shapes

# SVG

**Path** is the most flexible shape, has its own mini-language and can be used to draw lines, arcs, curves, ...

```
<path d="M10 315 L 110 215 A 30 50 0 0 1 162.55 162
  L 172.55 152.45 A 30 50 -45 0 1 215.1 109.9 L 315
```

MDN: SVG Paths



# SVG STYLING

# SVG STYLING

SVG elements can be styled using a variety of attributes:

```
<circle cx="50px" cy="50px" r="20px"
  stroke="blue"
  fill="pink" />
```

# SVG STYLING

There's also attributes for opacity (in addition to the regular CSS-opacity):

```
<circle cx="50px" cy="50px" r="20px"
  stroke="blue"
  fill="pink"
  stroke-opacity="0.5"
  fill-opacity="0.5" />
```

# SVG STYLING

Plus attributes for line-styles:

```
<circle cx="50px" cy="50px" r="20px"
  stroke="blue"
  fill="pink"
  stroke-width="20" />
```

# SVG STYLING

Most attributes can be set via CSS (except the "position/radius" ones):

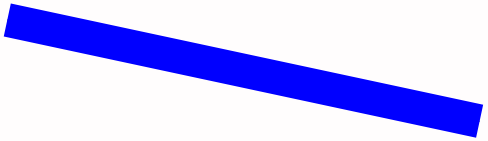
```
circle {
  fill: white; stroke: blue;
  stroke-width: 20; opacity: 0.5;
}
```

```
<circle cx="50px" cy="50px" r="20px"/>
```

# SVG STYLING

Some elements (e.g., *<line>*) have weird defaults and need styling to be visible at all:

```
<line x1="20" y1="20" x2="300" y2="80"
stroke="blue" stroke-width="20"></line>
```



# SVG STYLING

SVG also supports fills with gradients and general patterns:

MDN: Gradients

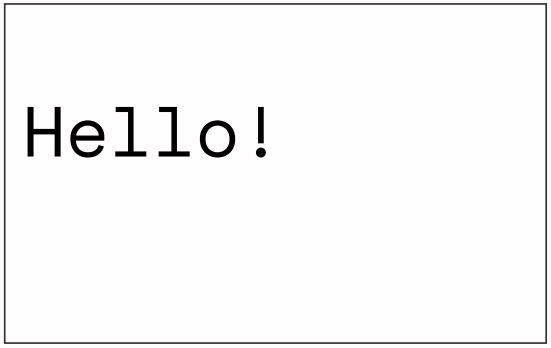
MDN: Patterns

# SVG TEXT

# SVG TEXT

SVG also has a *<text>* tag:

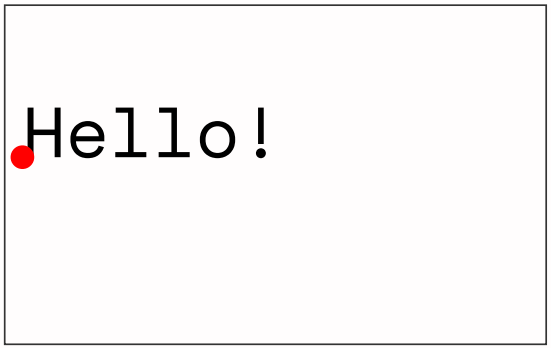
```
<text x="10" y="90">Hello!</text>
```



# SVG TEXT

Note that the *y* attribute determines the text's baseline!

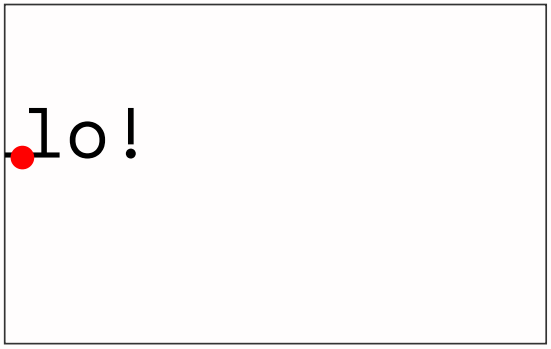
```
<text x="10" y="90">Hello!</text>
```



# SVG TEXT

SVG Text ignores *text-align*, use *text-anchor* (*start*, *middle*, *end*) instead.

```
<text x="10" y="90" text-anchor="middle">Hello!</te
```



# SVG TEXT

You can style SVG text using the regular CSS font attributes:

```
text {
  font-family: serif; font-style: italic;
  font-variant: small-caps; font-size: 94px;
}
```

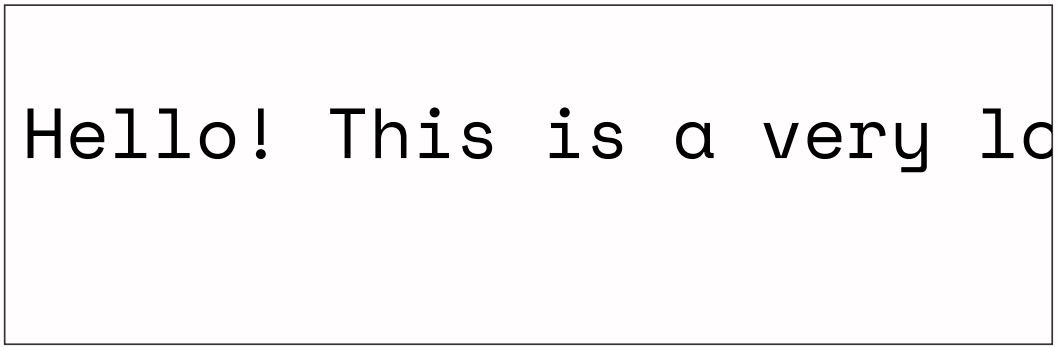
```
<text x="10" y="90">Hello!</text>
```


HELLO!

# SVG TEXT

SVG Text doesn't do line breaks :(

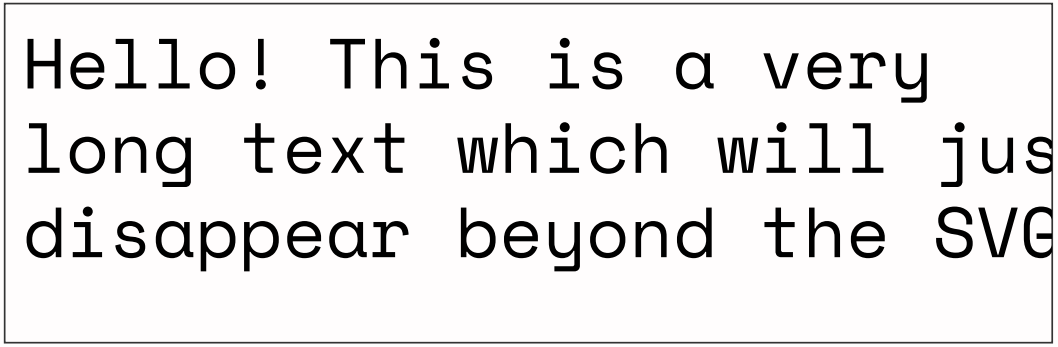
```
<text x="10" y="90">Hello! This is a very long text
  will just disappear beyond the SVG's borders.</te
```



# SVG TEXT

You can use the *tspan* element to do that manually...

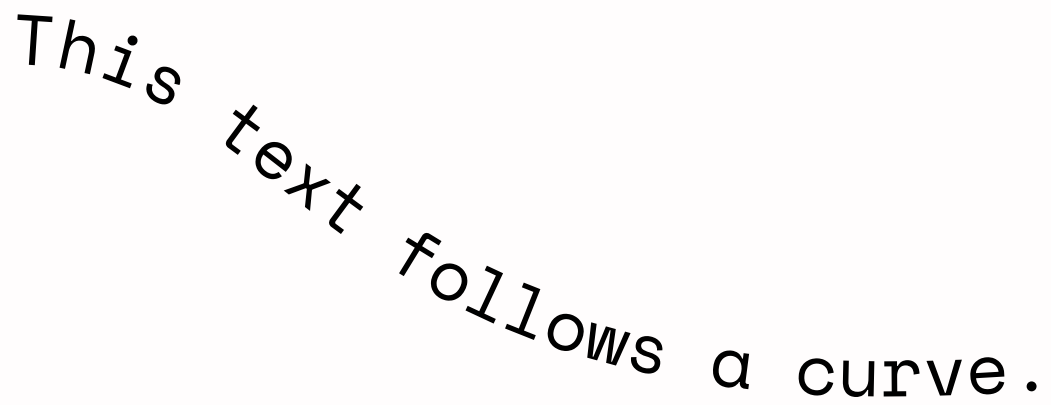
```
<text x="10" y="50"><Hello! This is a very
<tspan x="10" y="100">long text which will just</ts
<tspan x="10" y="150">disappear beyond the SVG's
borders.</tspan></text>
```



# SVG TEXT

Use *textPath* to make text paths:

```
<path id="my_path"
d="M 60,60 C 240,60 240,300 700,250" fill="transpar
<text>
  <textPath xlink:href="#my_path">This text follows
</text>
```

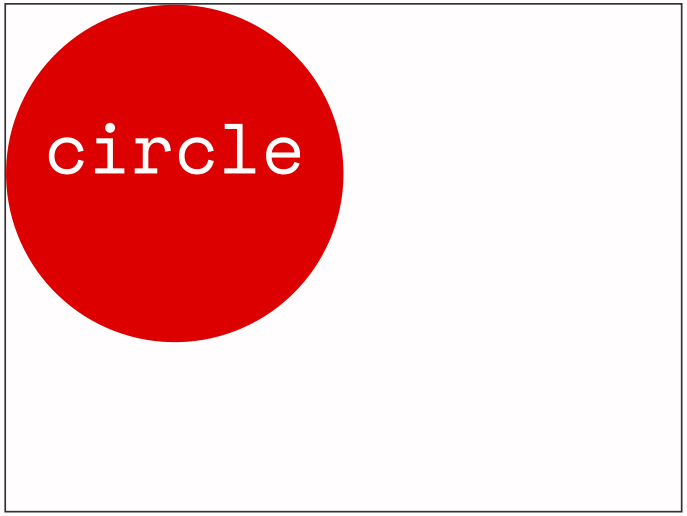

This text follows a curve.

# SVG GROUPS

# SVG GROUPS

Use groups *<g>* to apply attributes to several elements:

```
<g transform="translate(100, 100)">
  <circle cx="0" cy="0" r="100" fill="#dd0000" />
  <text x="0" y="0" fill="white" text-anchor="middl
</g>
```


circle

# SVG GROUPS

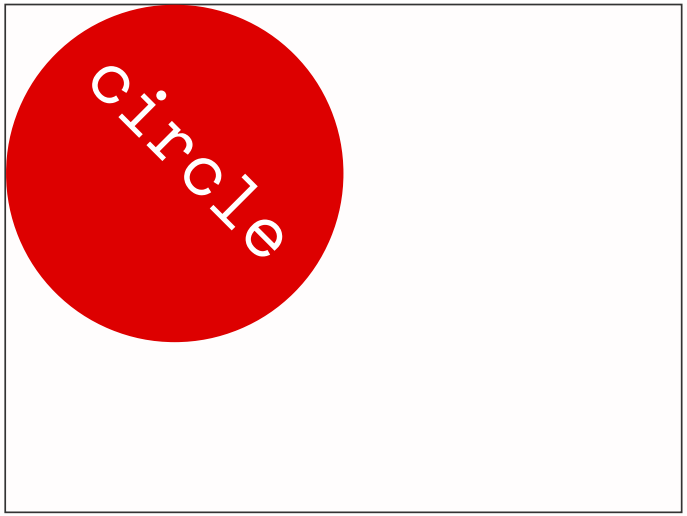
Use groups *<g>* to apply attributes to several elements:

```
<g transform="rotate(45)">
  <circle cx="0" cy="0" r="100" fill="#dd0000" />
  <text x="0" y="0" fill="white" text-anchor="middl
</g>
```

# SVG GROUPS

Use groups *<g>* to apply attributes to several elements:

```
<g transform="translate(100, 100) rotate(45)">
  <circle cx="0" cy="0" r="100" fill="#dd0000" />
  <text x="0" y="0" fill="white" text-anchor="middl
</g>
```


circle

# D3 DATA JOINS

# DATA JOINS

(This is the heart of d3 and where the name "data-driven documents" comes from)

# DATA JOINS

How do we tie our data to our DOM elements?

# DATA JOINS

d3 provides the function *data()* for its selectors:

```
d3.selectAll('p').data(myData);
```

Bostock: Thinking With Joins

# DATA JOINS

Let's have a look at code/d3-joins/01_data-join.html...

# DATA JOINS

Remember the mysterious *d* parameter from the d3 functions?

```
d3.select('p')
  .style('color', function(d){
    return d.color;
  });
```

THIS is your __data__ !

# DATA JOINS

Let's build on this in 01_data-join.html!

# DYNAMICALLY ADDING ELEMENTS

# DYNAMICALLY ADDING ELEMENTS

We'd want to add html-elements based on the data. Let's try that with code/d3-joins/02_fixed_data.html.

# DYNAMICALLY ADDING ELEMENTS

There's a better way for that in d3 (remember d3 being declarative instead of imperative?)

But it will also take us deeper into the very (weird) heart of d3...

# DYNAMICALLY ADDING ELEMENTS

Enter the *enter()* function.
*enter()* returns a selection with placeholder nodes for each elements of the data that's missing in the DOM.

# DYNAMICALLY ADDING ELEMENTS

```
var missingNodes = d3.select('#chart')
  .selectAll('p')
  .data(myData)
  .enter();

  console.log(missingNodes.nodes());
```

```
=> (5) [A, A, A, A, A]
```

# DYNAMICALLY ADDING ELEMENTS

Using *enter()* we can now add new nodes:

```
d3.select('#chart')
    .selectAll('p')
    .data(myData)
    .enter()
    .append('p');
```

# DYNAMICALLY ADDING ELEMENTS

... and even build on them:

```
d3.select('#chart')
  .selectAll('p')
  .data(myData)
  .enter()
  .append('p')
  .text(function(d,i){
    return "This node has data " + d;
  });
```

# DYNAMICALLY ADDING ELEMENTS

Let's try that with code/d3-joins/02_fixed_data.html.

# DYNAMICALLY ADDING ELEMENTS

Remember when we said that d3 was built for SVG?
Let's see about that and try building a bar chart (finally!) based on code/d3-joins/02_fixed_data.html

# DYNAMICALLY ADDING ELEMENTS

But what about the *Dynamically* in the title here?
Let's work with code/d3-joins/03_dynamic_data.html instead.

# DYNAMICALLY UPDATING ELEMENTS

# DYNAMICALLY UPDATING ELEMENTS

Similar to *enter()*, *exit()* returns a selection of all DOM elements that don't have corresponding data.

# DYNAMICALLY UPDATING ELEMENTS

```
d3.select('#chart')
  .selectAll('p')
  .data(myData)
  .exit()
  .remove()
```

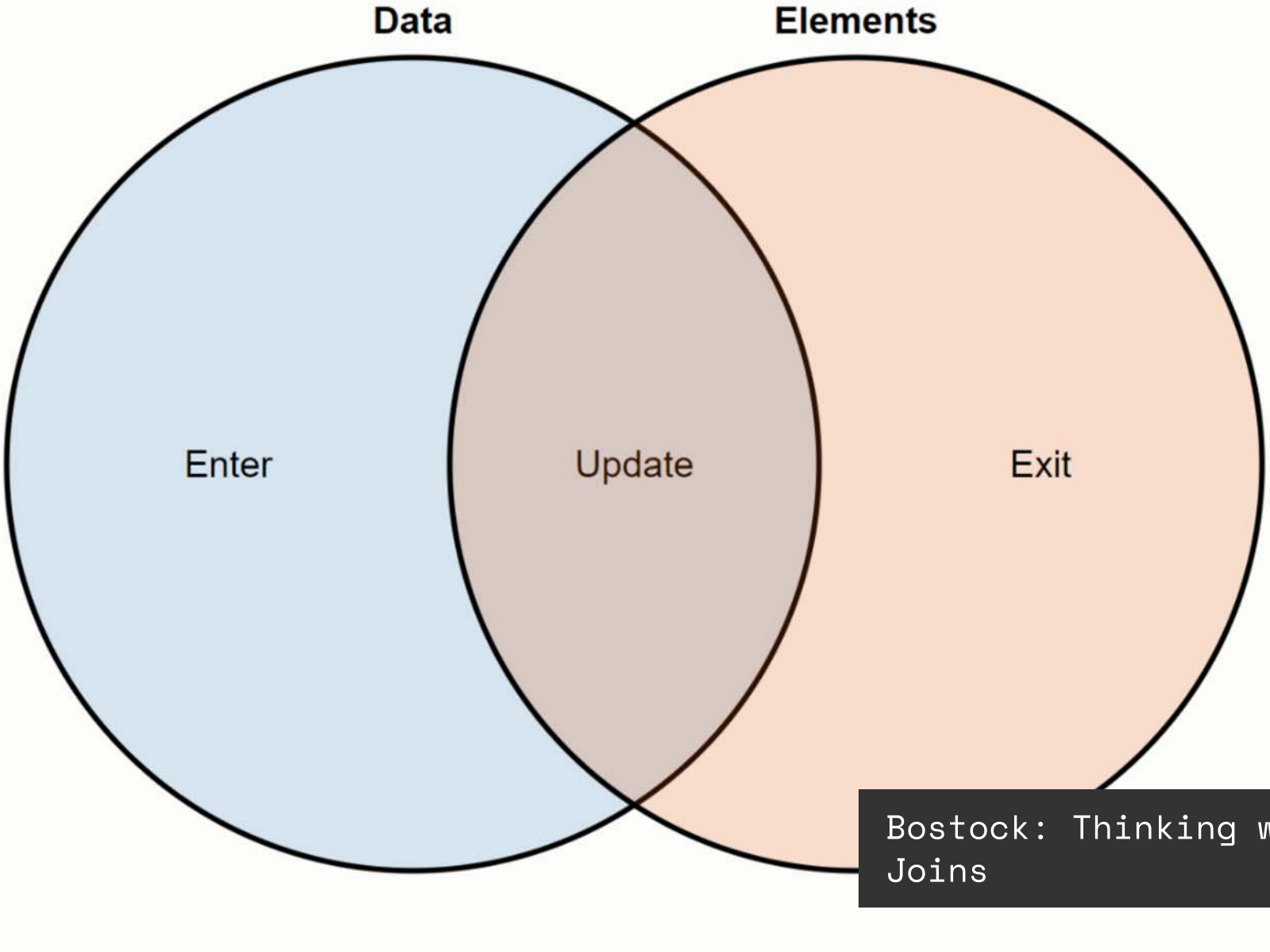
# DYNAMICALLY UPDATING ELEMENTS

*merge(GROUP)* lets you merge a selection with another.

```
var enter = d3.select('#chart')
    .selectAll('circle')
    .data(myData)
    .enter();
  var merge = d3.select('#chart').selectAll('circle
```

# ENTER, UPDATE, EXIT

## (A.K.A: THE GENERAL UPDATE PATTERN)




Bostock: Thinking with Joins

# ENTER, UPDATE, EXIT

```
var circle = svg.selectAll("circle")
    .data(data);

  circle.exit().remove();

  circle.enter().append("circle")
      .attr("r", 2.5)
    .merge(circle)
      .attr("cx", function(d) { return d.x; })
      .attr("cy", function(d) { return d.y; });
```

Bostock: Thinking with Joins

# ENTER, UPDATE, EXIT

Why declarative?

- works with changing data (realtime, interaction)
- Simpler code (less errors, easier debugging)

# DYNAMICALLY UPDATING ELEMENTS

Let's try it on code/d3-joins/04_dynamic_update.html.

# D3 ANIMATIONS

# D3 ANIMATIONS

Animations in d3 are super-simple: Just add *.transition()* between your selections and *.attr()/.style()*.

```
// non-animated:
d3.selectAll('rect').style('opacity', 0);
// animated:
d3.selectAll('rect').transition().style('opacity',
```

# D3 ANIMATIONS

Let's make our animations in code/d3-joins/04_dynamic_update.html nicer!

# D3 SCALES

# SCALES

Scales are functions that "map a dimension of abstract data to a visual representation". This can be anything from screen coordinates, to circles radii, to colors.

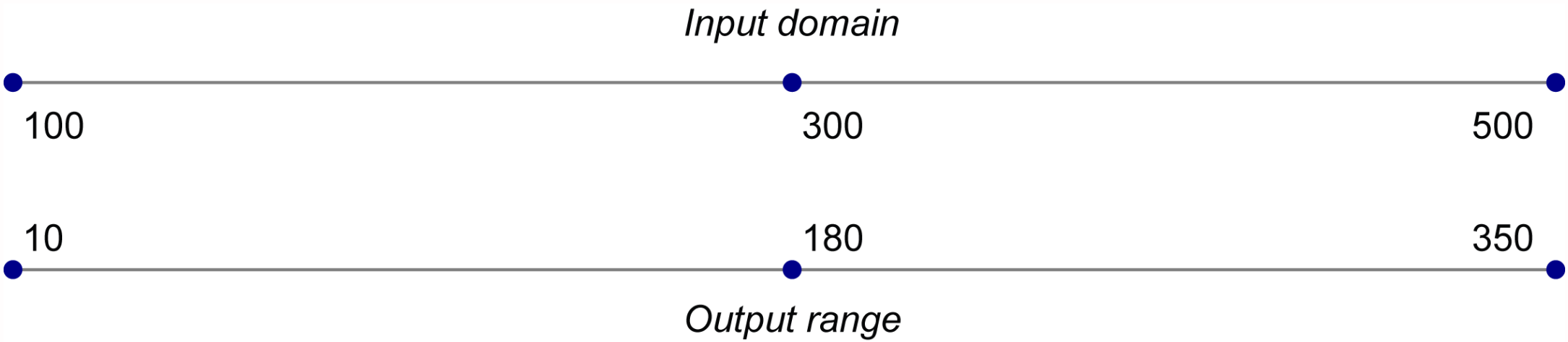
Github: d3-scale

# SCALES

Let's try it with a simple example: code/d3-scales/01_write_a_scale.html.

# SCALES

Scales map an input from a **domain** to an output **range**.




```
Murray: Interactive Data Visualization for the
Web
```

# SCALES

## d3 solution to 01_write_a_scale.html:

```
var myScale = d3.scaleLinear()
  .domain([0, 60])
  .range([0, 200]);
```

# CONTINUOUS SCALES

# CONTINUOUS SCALES

Linear scales (*d3.scaleLinear()*) map an input linearly to an output range:

```
d3.scaleLinear().domain([20, 100]).range([50, 500])
// ... gets the same results as:
var scale = function(d){ return
  ((d - 20) / (100 - 20)) * (500 - 50) + 50;
}
```

# CONTINUOUS SCALES

Linear scales do not automatically cut off input values outside of the domain:

```
var scale = d3.scaleLinear()
  .domain([20, 100])
  .range([50, 500]);

scale(0);    // => -62.5
scale(200);  // => 1062.5
```

# CONTINUOUS SCALES

Use *clamp()* to enforce cut-offs:

```
var scale = d3.scaleLinear()
  .domain([20, 100])
  .range([50, 500])
  .clamp();

scale(0);    // => 50
scale(200);  // => 500
```

# CONTINUOUS SCALES

d3 linear scales even work with color ranges:

```
var scale = d3.scaleLinear()
  .domain([20, 100])
  .range(["aqua", "firebrick"]);

scale(40);  // => "rgb(45, 200, 200)"
scale(80);  // => "rgb(134, 89, 89)"
```

# CONTINUOUS SCALES

All scales support *invert()*, which returns the corresponding input for a given output value:

```
var scale = d3.scaleLinear()
  .domain([20, 100])
  .range([50, 500]);

scale.invert(500);  // => 100
scale.invert(50);  // => 20
```

# CONTINUOUS SCALES

d3 comes with more built-in continuous scales:

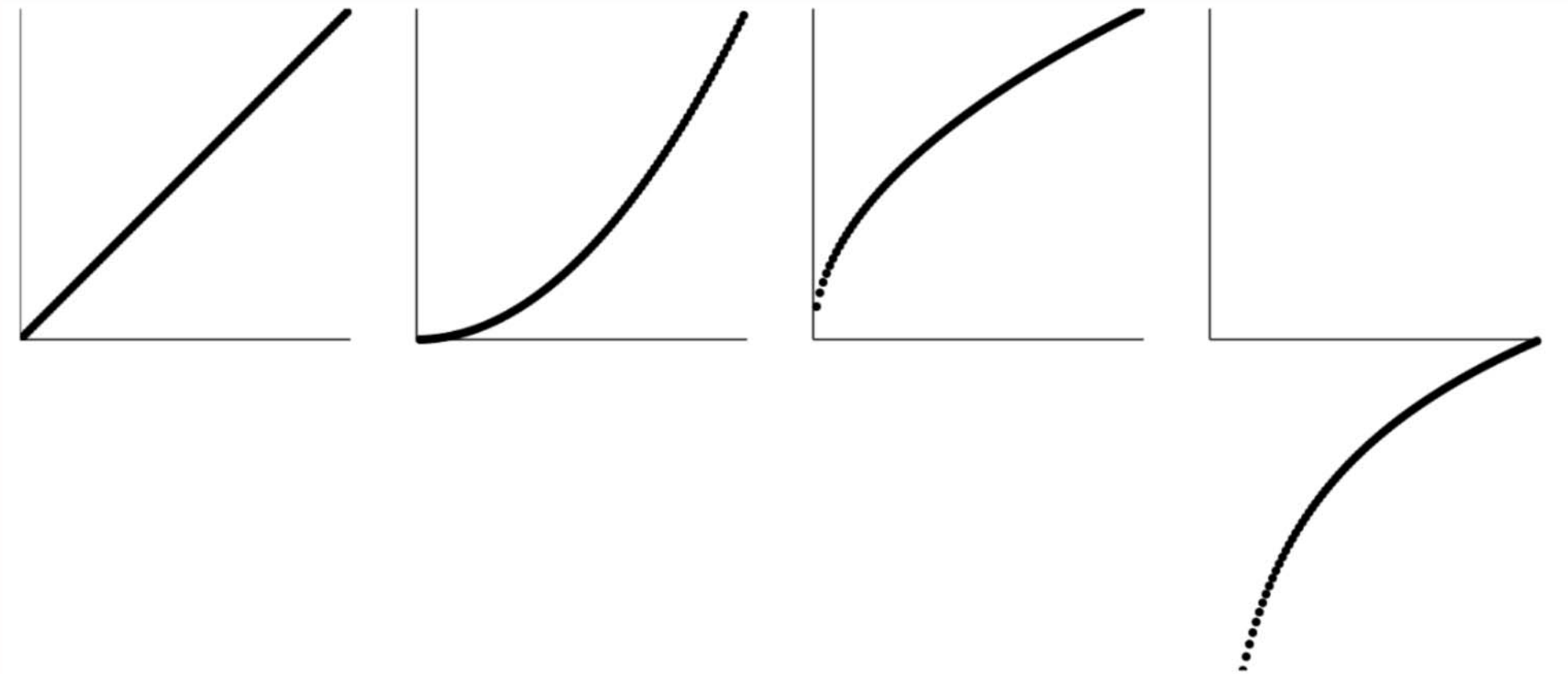
*d3.scalePow* for power scales (set exponent with *.exponent()*)

*d3.scaleSqrt* for a square root scale

*d3.scaleLog* for a logarithmic scale

# CONTINUOUS SCALE

Let's plot these scales... (code/d3-scales/02_continuous.html)



# TIME SCALES

# TIME SCALES

One commonly used input domain is time. *d3.scaleTime()* works as a linear scale for that domain.

# TIME SCALES

```
var s = d3.scaleTime()
  .domain([new Date(2017, 0, 1), new Date(2017, 11,
  .range([0, 800]);

s(new Date(2017, 2, 1))  // => 129.31506849315068
s(new Date(2017, 7, 10)) // => 484.2922374429223
```

# ORDINAL SCALES

# ORDINAL SCALES

*d3.scaleOrdinal()* takes an array of ordinal values as its input domain:

```
var s = d3.scaleOrdinal()
  .domain(["apples", "bananas", "oranges"])
  .range([0, 300]);

s("apples");  // => 0
s("bananas"); // => 150
s("batman");  // => 300
```

# D3 AXES

# AXES

d3 comes with built-in axes generators.

Github: d3-axes

# AXES

Axes are based on scales. d3 produces four different ones:

```
d3.axisTop(scale);
d3.axisRight(scale);
d3.axisBottom(scale);
d3.axisLeft(scale);
```

# AXES

They can be called into existence from a selection:

```
var myScale = d3.scaleLinear()...;

var myAxis = d3.axisLeft(myScale);
svg.append("g")
  .call(myAxis);
```

# AXES

Let's try them out: code/d3-scales/03_axes_missing.html.

# AXES CONFIG

# AXES CONFIG

*ticks* is useful for configuring the axis generator:

```
var myAxis = d3.axisLeft(myScale);
// use 5 ticks:
myAxis.ticks(5);
// format ticks:
myAxis.ticks(5, "s")
```

# AXES CONFIG

d3 formatting:

d3.format

d3.timeFormat

# AXES CONFIG

# AXES CONFIG

Use *tickValues* to explicitly set ticks:

```
var myAxis = d3.axisLeft(myScale);
myAxis.tickValues([0, 0.5, 1]);
```