Module 10

Data Processing and Text

CS 106 Winter 2018

3,855,916,748

Internet Users in the world

1,340,377,646

Total number of Websites

204,637,856,125

Emails sent today

4,829,632,657

Google searches today

4,544,266

Blog posts written today

585,848,322

Tweets sent today

5,369,706,868

Videos viewed today
on YouTube

61,276,587

Photos uploaded today
on Instagram

99,302,167

Tumblr posts today

www.internetlivestats.com

TOTAL GLOBAL EMAIL & SPAM VOLUME FOR JANUARY 2018

Average Daily Legitimate Email Volume

73.13 BILLION

Email Volume Change from Previous Month

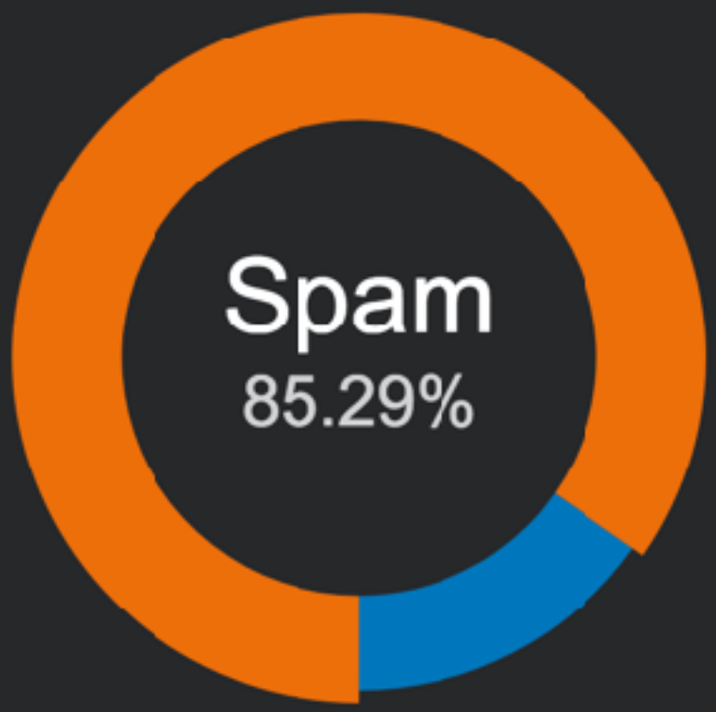
3.4%

Average Daily Spam Volume

421.81 BILLION

Spam Volume Change from Previous Month

+4%



DAILY EMAIL VOLUME

EMAIL TYPE	AVERAGE DAILY VOLUME (BILLIONS)	PERCENTAGE OF GLOBAL TRAFFIC
Legitimate	80.02	14.70%
Spam	464.12	85.29%

www.talosintelligence.com/reputation_center/email_rep

TRAFFIC FROM MOBILE & ONLINE MESSAGING TO REACH 438 BILLION PER DAY BY 2019

Hampshire, UK: 6th July 2015: New data from Juniper Research has shown that mobile and online messaging traffic will reach 160 trillion per annum by 2019, up from 94.2 trillion this year – equating to approximately 438 billion messages sent and received by users on a daily basis by 2019. These figures incorporate SMS, MMS, IM (Instant Messaging), Social Media and Email.

Last year, email accounted for the largest share of traffic, at around 35 trillion messages per year – although almost 80% of this figure (28 trillion) can be categorised as spam. However, within the next 12 months IM will overtake email generating almost 43 trillion messages annually.

YouTube Company Statistics	Data
Total number of people who use YouTube	1,325,000,000
Hours of video uploaded to YouTube every minute	300 hours
Number of videos viewed on YouTube everyday	4,950,000,000
Number of unique visits to YouTube every month	900,000,000
Total number of hours of video watched on YouTube each month	3.25 billion hours
Number of YouTube videos that have generated over 1 billion views	10,113
Percent of YouTube visitors that come from outside the U.S.	70 %
Number of countries with localized versions of YouTube	42
Total number of languages Youtube is broadcast in	54
User submitted video with the most views – "Charlie bit my finger"	829,000,000
Average number of mobile YouTube video views per day	1,000,000,000
Average time spent on YouTube per mobile session	40 minutes
Average YouTube partner channel payout per 5,000 views	$0.32

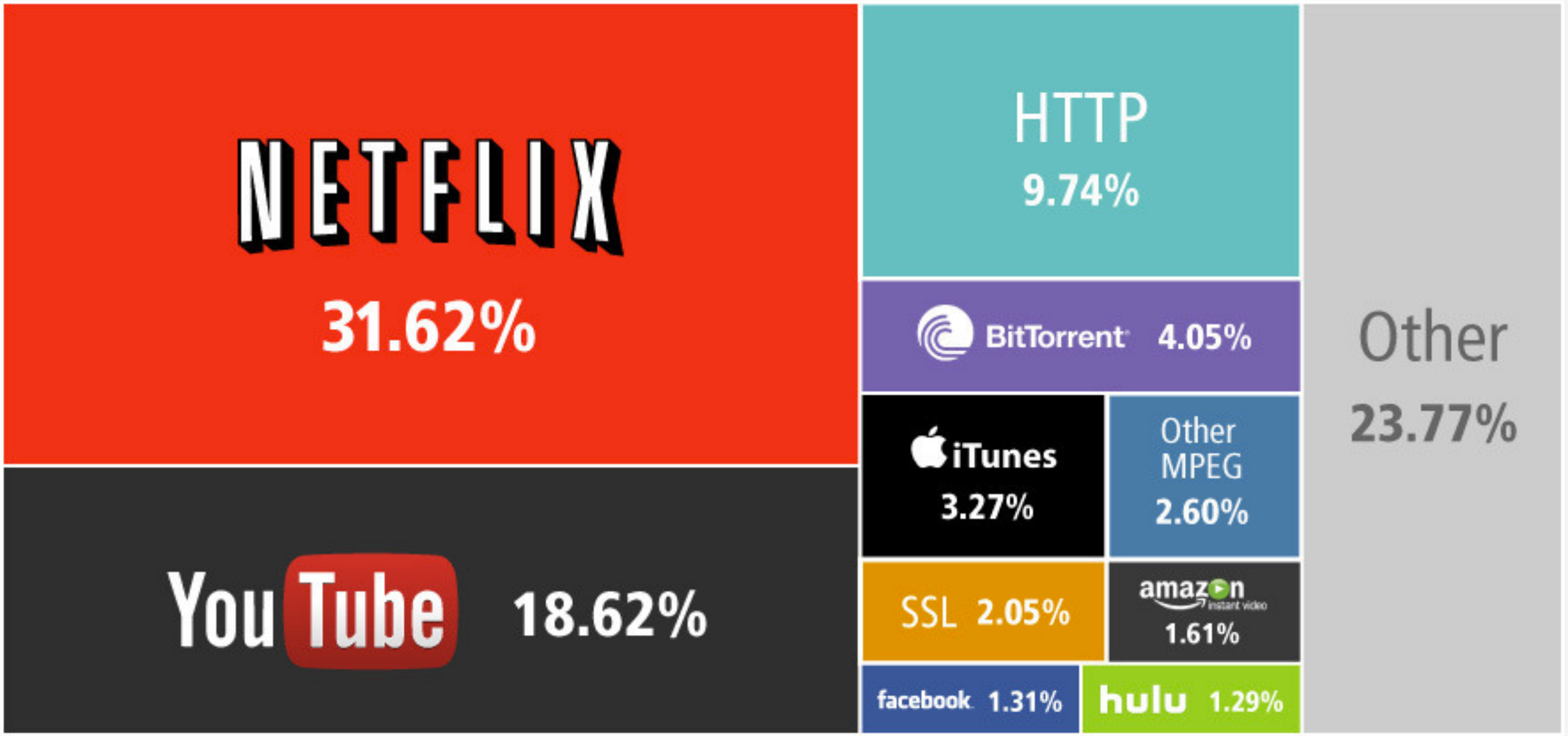
www.statisticbrain.com/youtube-statistics/

YouTube Company Statistics	Data
Total number of people who use YouTube	1,325,000,000
Hours of video uploaded to YouTube every minute	300 hours
Number of videos viewed on YouTube everyday	4,950,000,000
Number of unique visits to YouTube every month	900,000,000
Total number of hours of video watched on YouTube each month	3.25 billion hours
Number of YouTube videos that have generated over 1 billion views	10,113
Percent of YouTube visitors that come from outside the U.S.	70 %
Number of countries with localized versions of YouTube	42
Total number of languages Youtube is broadcast in	54
User submitted video with the most views – "Charlie bit my finger"	829,000,000
Average number of mobile YouTube video views per day	1,000,000,000
Average time spent on YouTube per mobile session	40 minutes
Average YouTube partner channel payout per 5,000 views	$0.32

www.statisticbrain.com/youtube-statistics/

Netflix and YouTube Are America's Biggest Traffic Hogs

Share of peak period downstream traffic in North America, by application*



* September 2013. Fixed access only.

statista The Statistics Portal

Mashable

Source: Sandvine

Data challenges

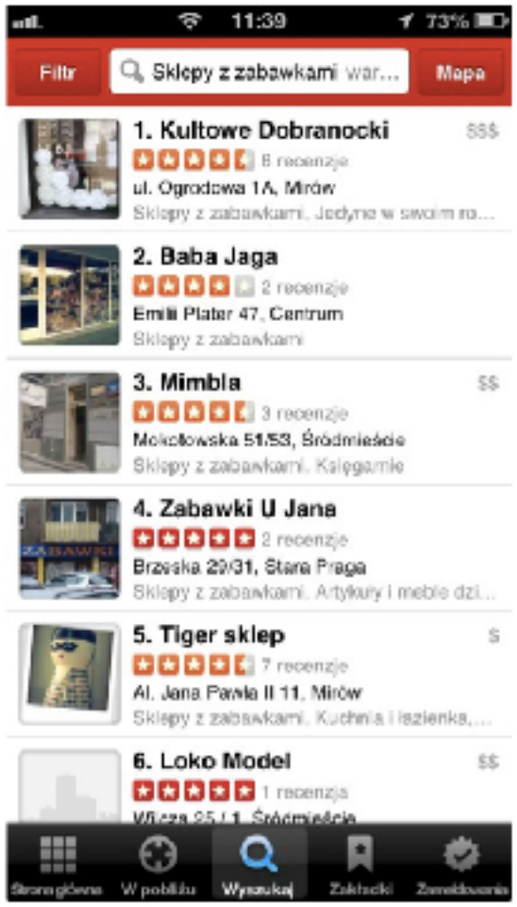
- Creating it
- Storing it
- Moving it around
- Keeping it private

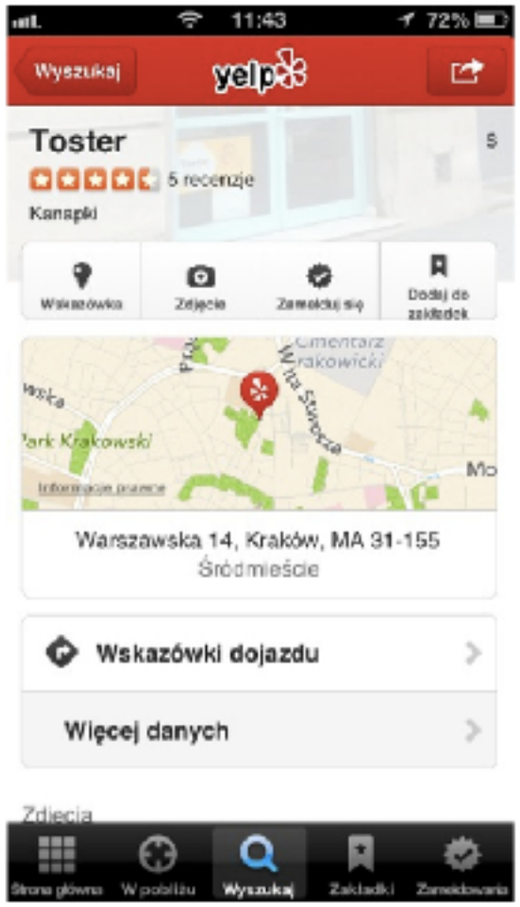
Data challenges

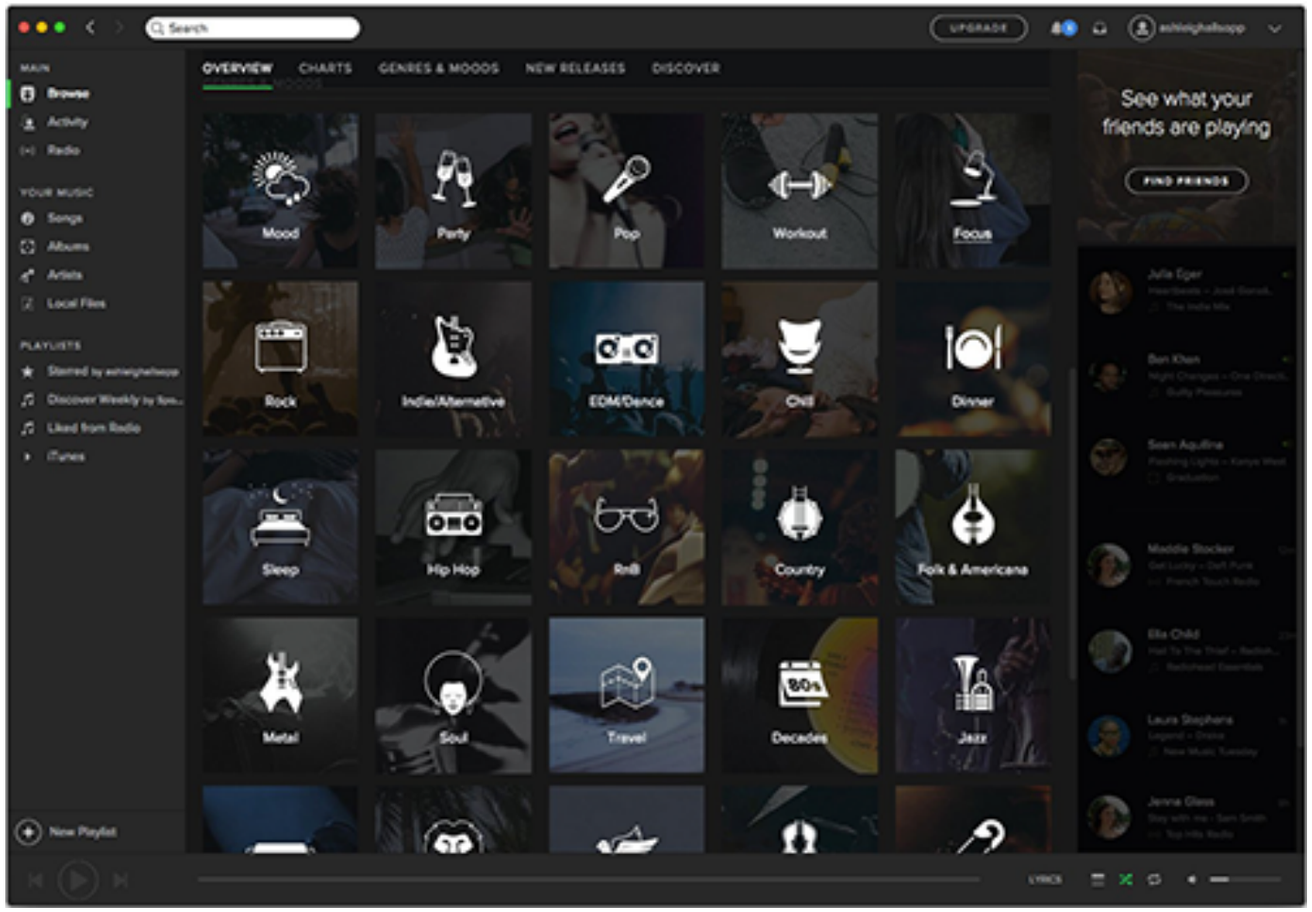
- Creating it
- Storing it
- Moving it around
- Keeping it private
- **Making sense of it**

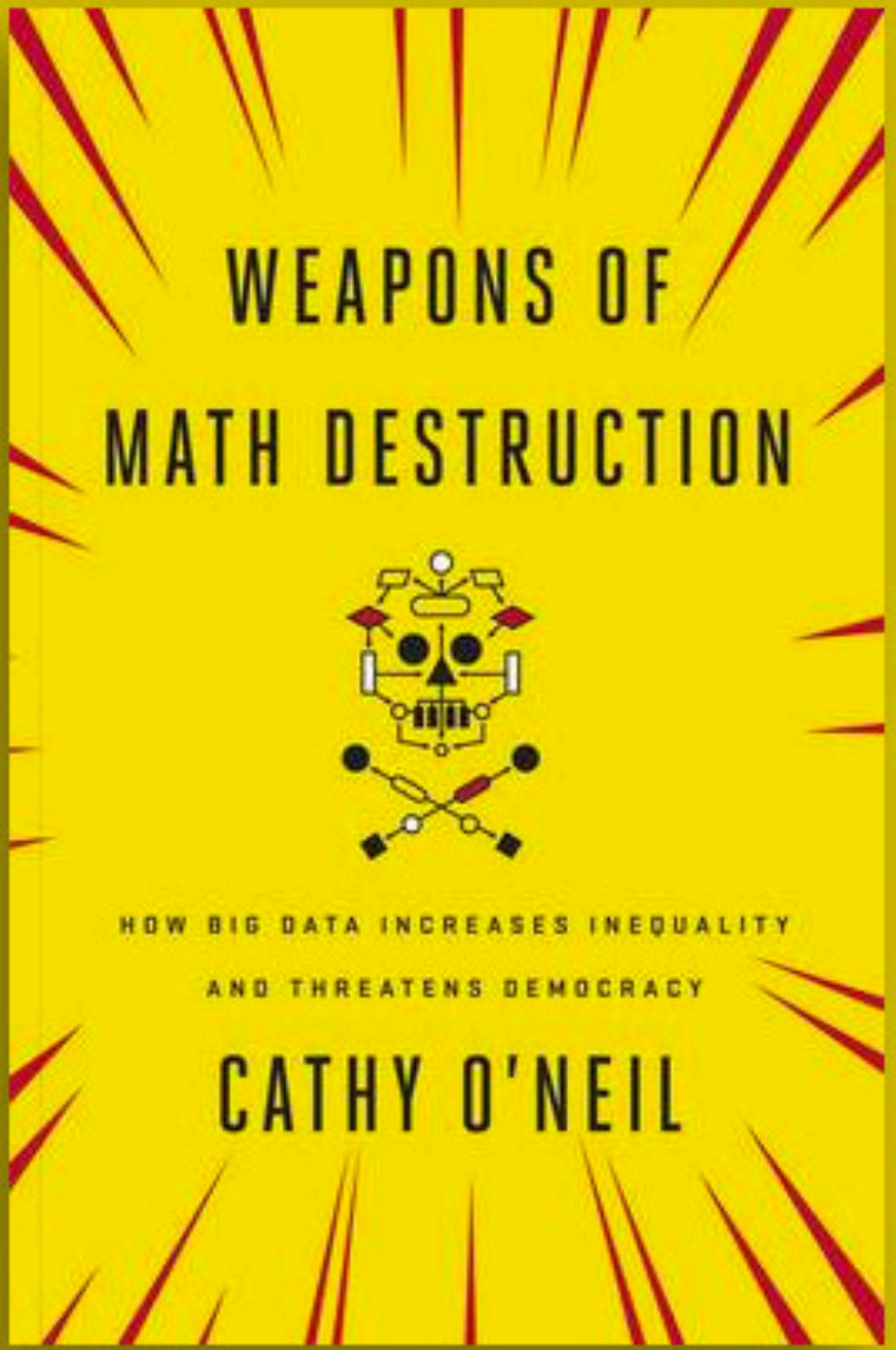
Searching, indexing

Collecting, Correlating, Recommending

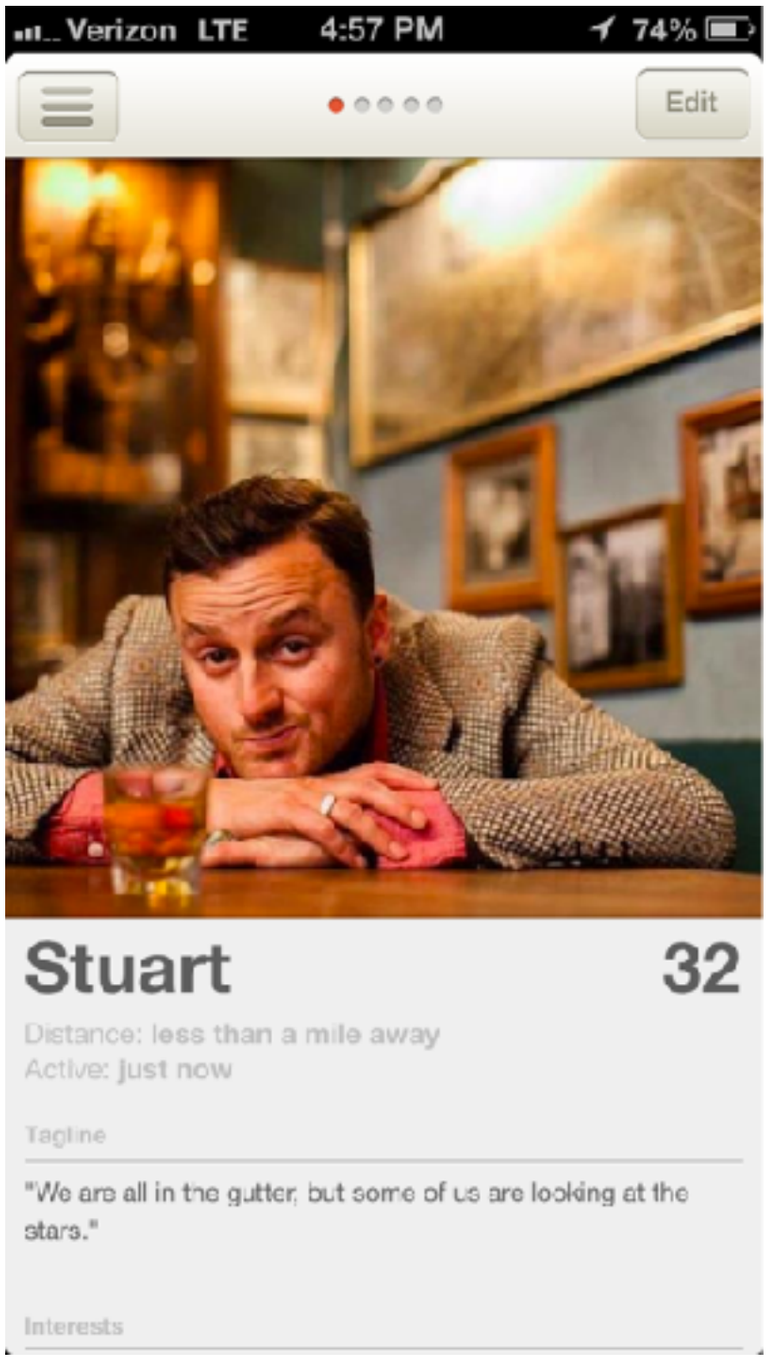







WEAPONS OF
MATH DESTRUCTION
HOW BIG DATA INCREASES INEQUALITY
AND THREATENS DEMOCRACY
CATHY O'NEIL

Collecting, Correlating, Recommending



GOOGLE

online advertising

All News Images Videos Books More Search tools

About 570,000,000 results (0.64 seconds)

Online Advertising - Convert Shoppers with Relevant Ads

Ad www.criteo.com/Mobile

Make More Sales with Criteo Today.

10,000 brands · 130 countries

Services: Transparent CPC Pricing, Unparalled Technology, Dynamic Creative, Cross ...

Contact Us

Tell Us A Little About Yourself.

Find a Local Office Near You.

What We Do

Driving Better Marketing Results.

Learn About Our Technology Today

Advertising Online - Marketing360.com

Ad www.marketing360.com/**OnlineAdvertising** +1 855-773-8169

#1 Marketing Platform® For **Advertising Online**. Free Demo!

Online Advertising - Reach More Customers Online

Ad www.rogersoutrank.com/Client_Le**ads**

Demo OutRank's Powerful Platform.

Online advertising - Wikipedia, the free encyclopedia

https://en.wikipedia.org/wiki/**Online_advertising**

Online advertising, also called online marketing or **Internet advertising** or **web advertising**, is a form of marketing and advertising which uses the Internet to ...

Display advertising - Web banner - Mobile advertising - Paywall

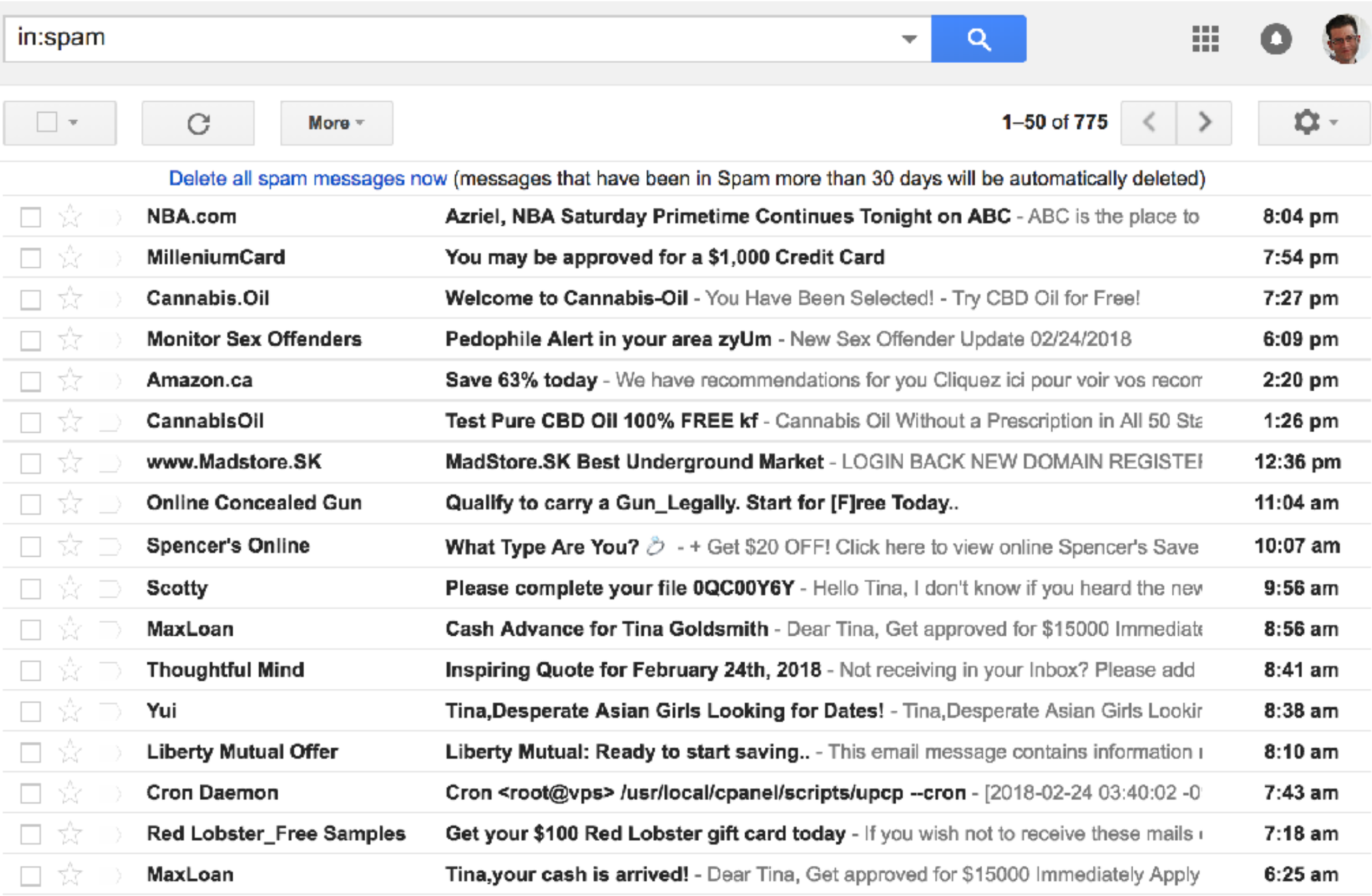
Rose repeats to Hammerbacher–who's a founder of data analytics company Cloudera–a line from an interview he gave *Businessweek* back when he was an early employee hustling stats for Harvard bud Zuckerberg at Facebook:

> "The best minds of my generation are thinking about how to make people click ads."

And Rose, in his politeness, left off the last part of the line: "That sucks."

www.fastcompany.com/3008436/why-data-god-jeffrey-hammerbacher-left-facebook-found-cloudera

Patterns, trends, predictions



Inbox — Gmail (15 messa

Mailboxes | Inbox | VIPs | Drafts (95) | Sent (1) | Flagged

Mailboxes

- Inbox
  - UW Connect
  - Gmail
  - UW-CS
  - Isohedral
- VIPs
- Flagged 32
- Drafts 95
- Sent
  - UW Connect
  - Gmail 1
  - UW-CS
  - Isohedral
- Junk
  - UW Conn... 5,361
  - Gmail 611
  - Isohedral 243

Sort by Date

Hang w/ 2015-03-15
Welcome To Hang w/ - Private Con Man Access...

Humble Bundle 2015-09-10
Your Humble Bundle order is ready

Con Man Productions 2016-04-18
😂 Con Man Commentary Tracks with Alan Tudy...

Con Man Productions 2016-05-18
🐵 Stretch Goal Reward: Free Spectrum Issue #0!

Cye Waldman 2016-07-04
[The Tiling List] Voderberg rant

XVideos 2016-11-27
Your account on Xnxx.com

Official Pokémon GO Team 2017-01-06
Videos: Ditto, New Pokémon and Fun Facts for T...

Ubisoft Account Support 2017-01-17
Welcome to Ubisoft

FirstMediaX Info 2017-01-22
Informasi mengenai Link Account 9995805904...

CanadaHelps 2017-02-07
Your charitable tax receipt for 2016 is ready!

Kitchener-Waterloo Symphony 2017-03-04
Renew your 2017/18 subscription by March 17 fo...

billing@canadianwebhosting.com 2017-03-09
Customer Invoice

Official Pokémon GO Tea
Videos: Ditto, New Pokémon
To: ezemoney126441
Reply-To: Official Pokémon

Welcome To

MalaysianCupid.com!

MalaysianCupid.com

Dear Member

Congratulations on joining MalaysianCupid.com!
This email contains your logon information and important information that will help you get the most out of your membership and help you to find your perfect partner.

YOUR LOGIN DETAILS:

Login Email: azriel@gmail.com

Your Password is: **ayang78**

Please save this email for future reference!

Thanks for your payment, processed on June 02, 2015.

Hello, AZRIEL TEPPER

American Express Card
81009

We processed your scheduled payment.

$987.91

PROCESSED ON

June 02, 2015

It's processed today - but give us 24 - 36 hours for your payment to appear online.
View your account.

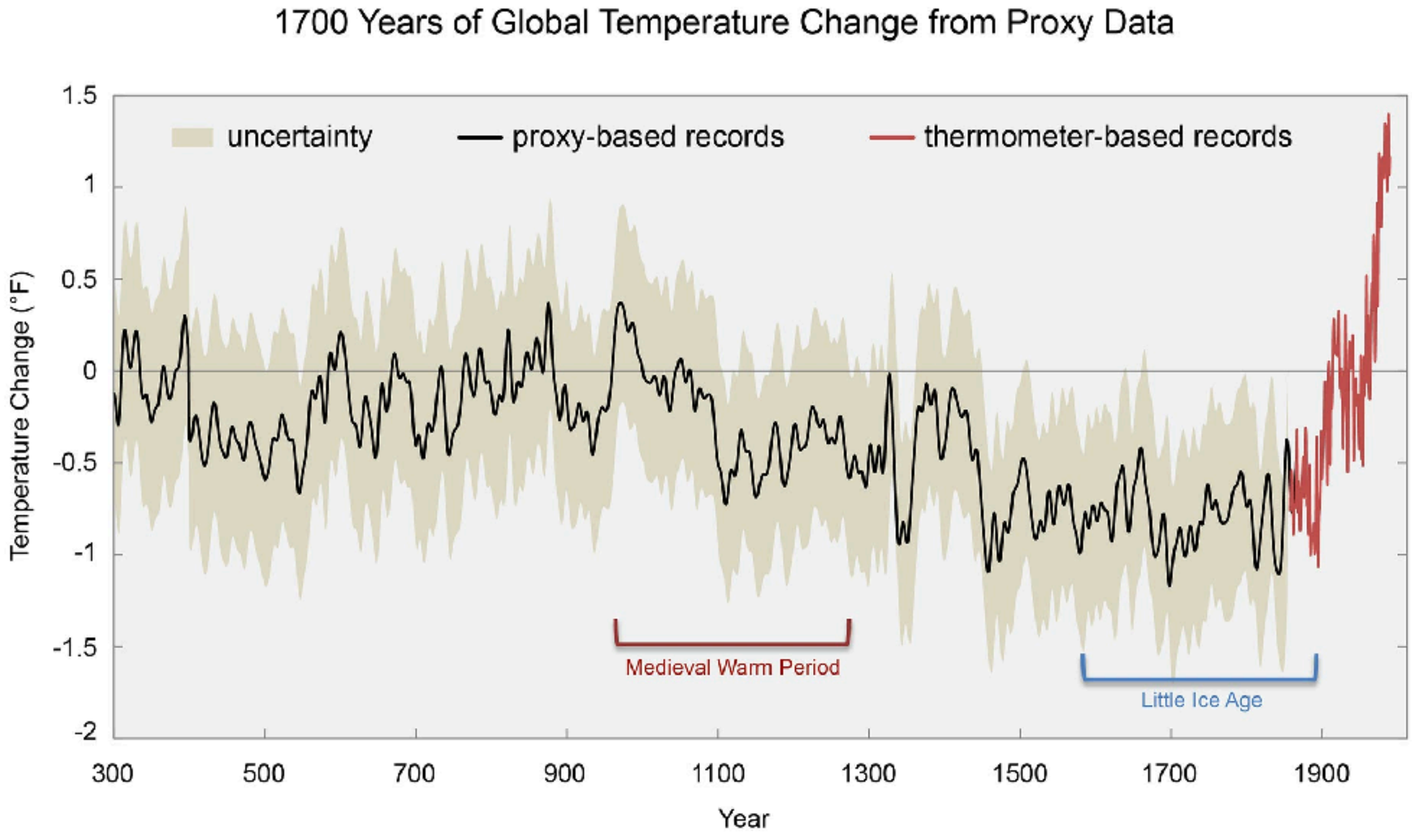
Thanks for your Card Membership,
American Express Customer Care

Was this e-mail helpful? Give us your feedback

STAY CONNECTED

@AskAmex
Amex Customer Care, at your service.

Community_@Amex
Your questions. Your interests. Your community


1700 Years of Global Temperature Change from Proxy Data
uncertainty
proxy-based records
thermometer-based records
Temperature Change (°F)
1.5
1
0.5
0
-0.5
-1
-1.5
-2
300
500
700
900
1100
1300
1500
1700
1900
Year
Medieval Warm Period
Little Ice Age

TRUMPING

The Data That Turned the World Upside Down

HG

HANNES GRASSEGGER AND MIKAEL KROGERUS
Jan 28 2017, 9:15am

Psychologist Michal Kosinski developed a method to analyze people in minute detail based on their Facebook activity. Did a similar tool help propel Donald Trump to victory? Two reporters from Zurich-based Das Magazin went data-gathering.

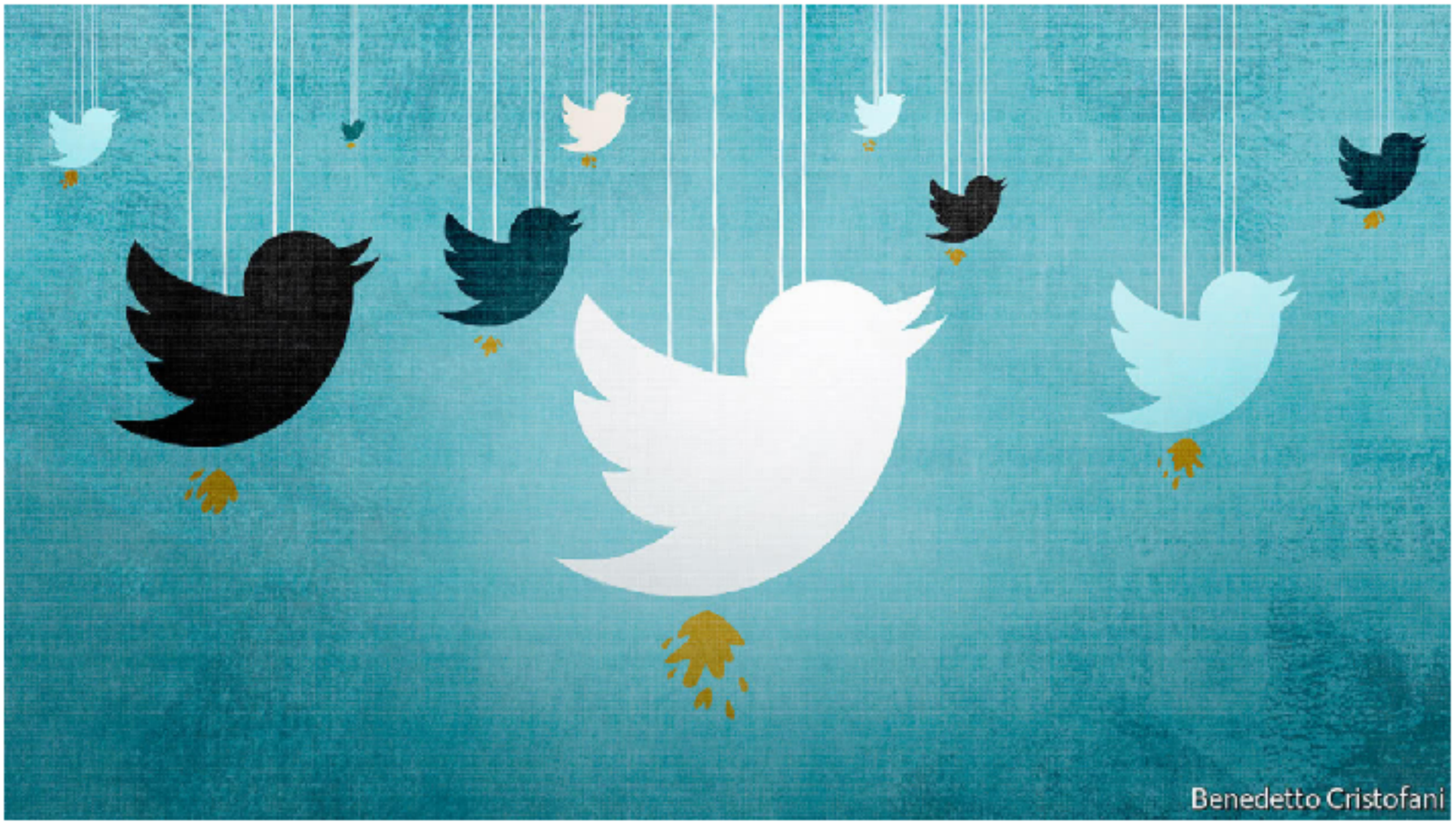
An earlier version of this story appeared in Das Magazin in December.

motherboard.vice.com/en_us/article/how-our-likes-helped-trump-win

How to be a Dadaist troll

Inside the Internet Research Agency's lie machine

Serving up fake news for Mr Putin's chef



www.economist.com/news/briefing/21737296-serving-up-fake-news-mr-putins-chef-inside-internet-research-agencys-lie-machine

Vault 7: CIA Hacking Tools Revealed

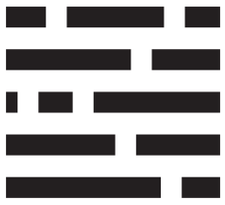
Contents

wikileaks.org/ciav7p1/

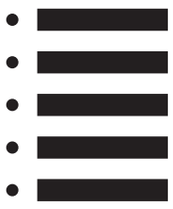
The shape of data

How is your information organized? How do the parts relate to each other?

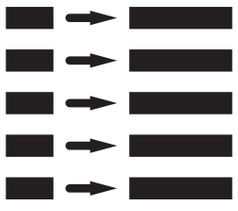
These questions profoundly affect the tools you use and the code you write.



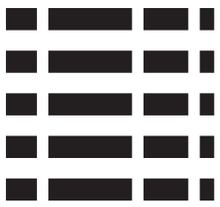
Raw text



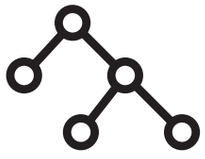
Sequence



Dictionary



Table

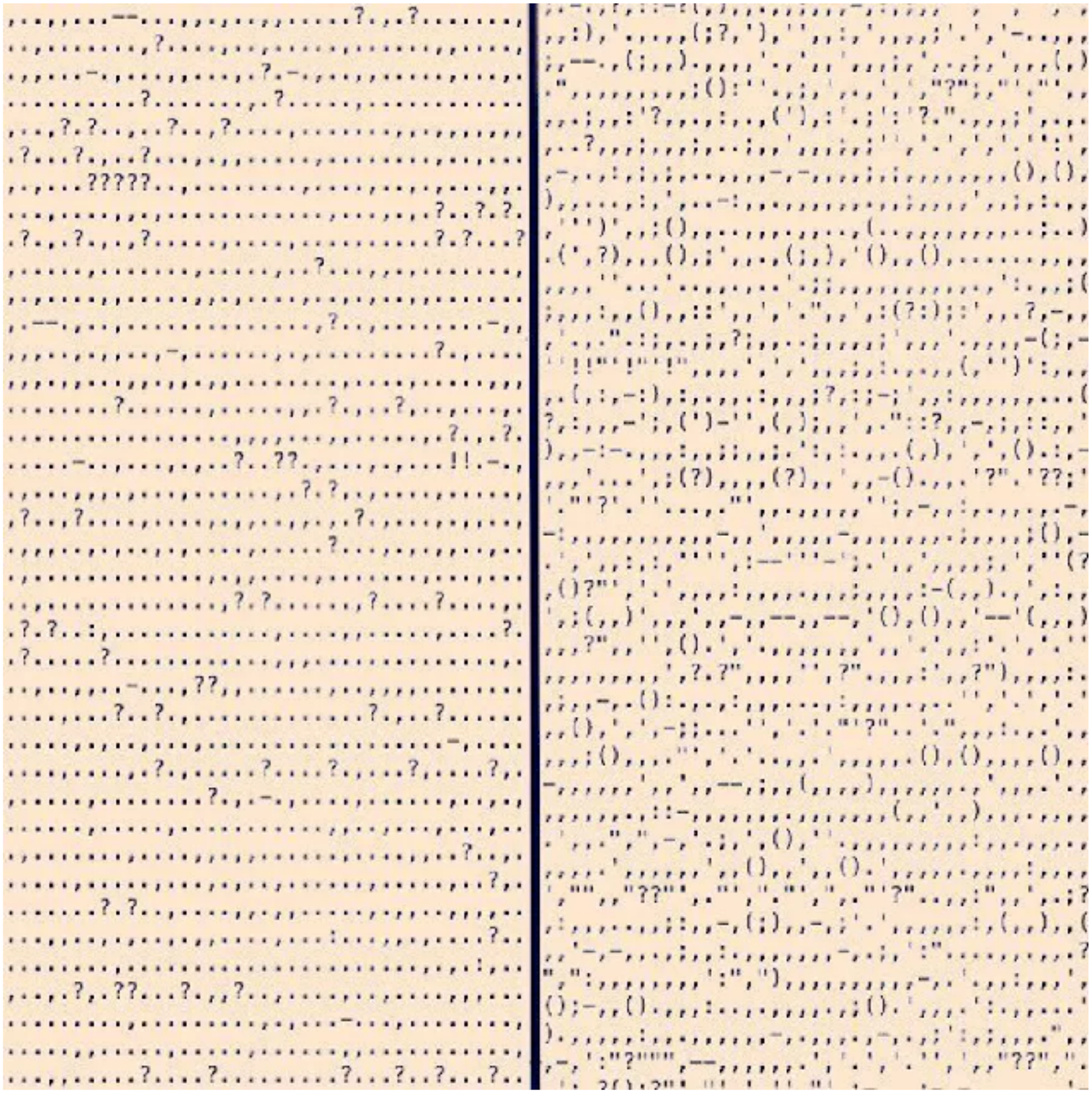


Tree

Graph

Raw text

Call me Ishmael. Some years ago—never mind how long precisely —having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off—then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in



McCarthy Faulkner

medium.com/@neuroecology/punctuation-in-novels-8f316d542ec4

Absalom, Absalom!

A Farewell To Arms

Alice in Wonderland

Blood Meridian

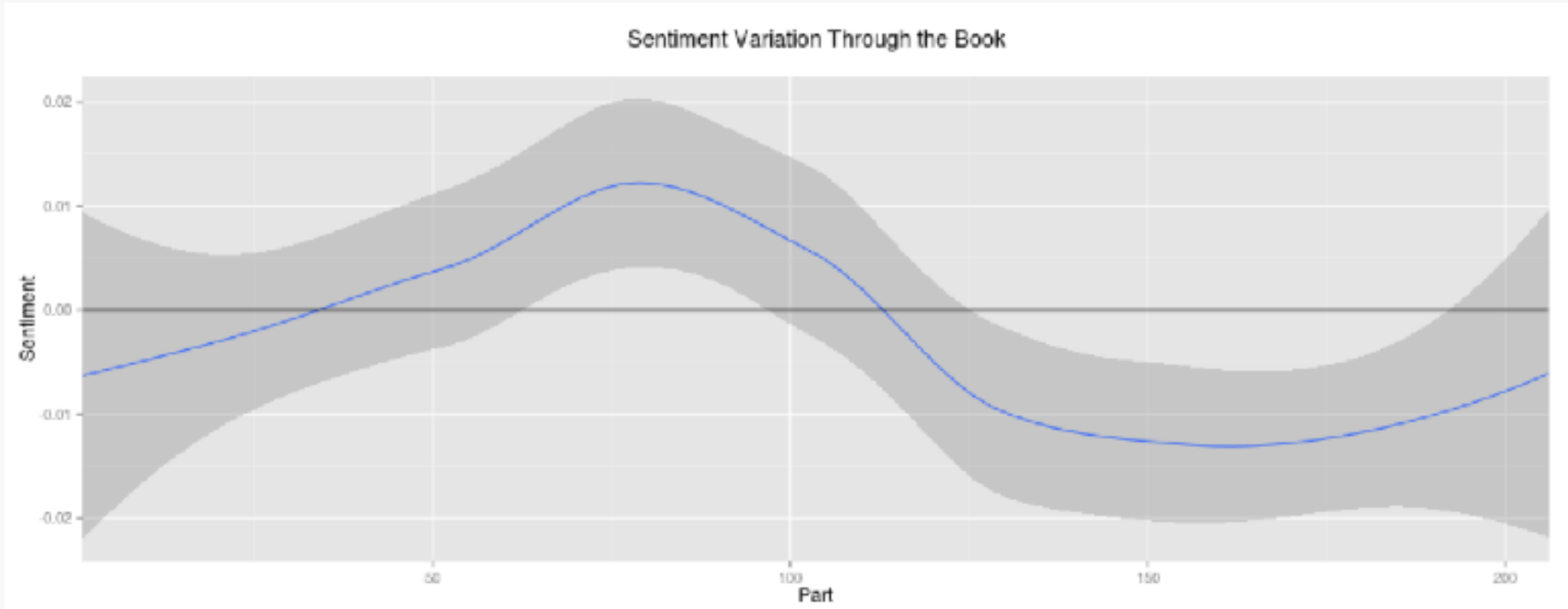
Frankenstein

Great Expectations

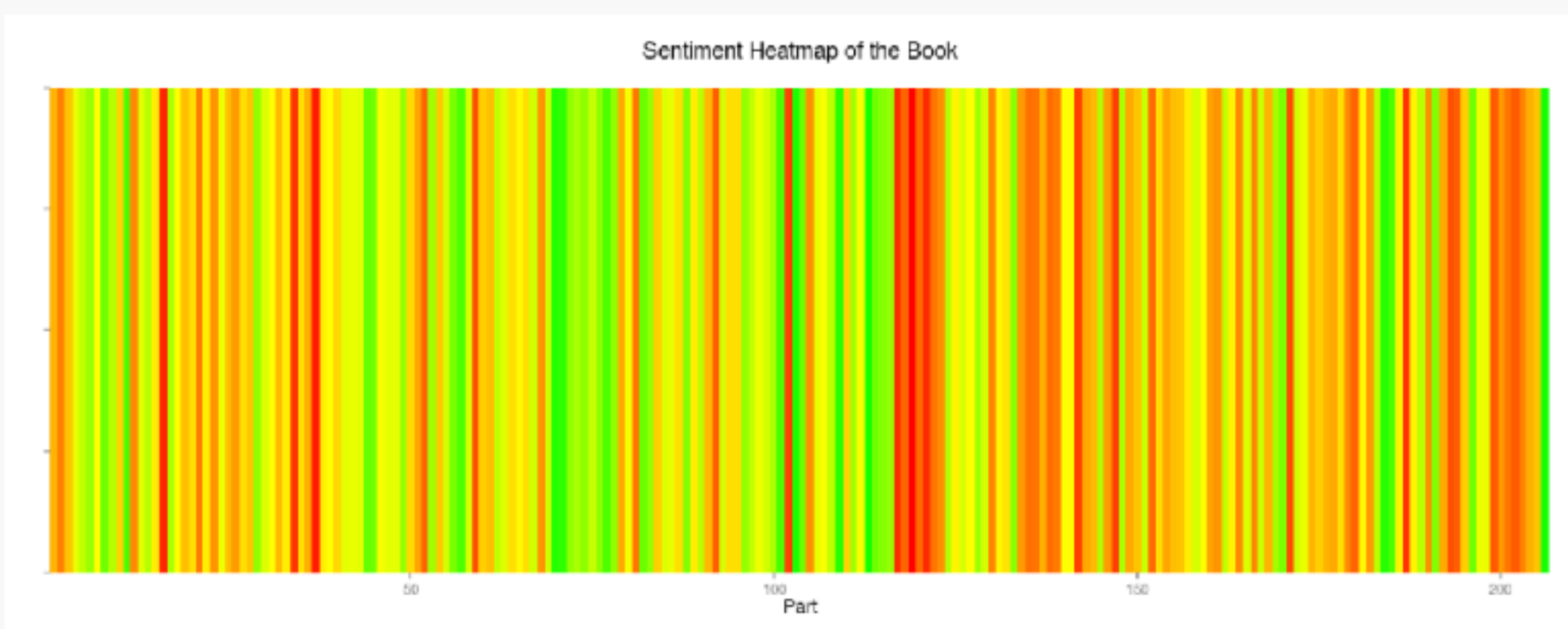
Huckleberry Finn

Pride and Prejudice

Ulysses



The values above zero indicate 'positive' emotions, and the values below zero indicate 'negative' emotions



Red is negative, green is positive, yellow is neutral

www.hotdamndata.com/2013/07/visualizing-book-sentiments.html

```
Received: from CONNMBX02.connect.uwaterloo.ca ([129.97.149.109]) by
 connhub1.connect.uwaterloo.ca ([129.97.149.101]) with mapi id 14.03.0319.002;
 Tue, 17 Jan 2017 15:57:38 -0500
From: Rishabh Moudgil <rishabh.moudgil@uwaterloo.ca>
To: Craig Kaplan <csk@uwaterloo.ca>
CC: Kevin Harrigan <kevinh@uwaterloo.ca>, Kristina Bayda
  <kbayda@uwaterloo.ca>, Travis Bartlett <travis.bartlett@uwaterloo.ca>
Subject: A01 Marking Scheme
Thread-Topic: A01 Marking Scheme
Thread-Index: AdJw/+DUxNKRRlCRRKOZfc2CQLKSng==
Date: Tue, 17 Jan 2017 20:57:36 +0000
Message-ID: <748888CA42FDF349AF07A8978DDED060281C9EC0@connmbx02>
Accept-Language: en-CA, en-US
Content-Language: en-CA
X-MS-Exchange-Organization-AuthAs: Internal
X-MS-Exchange-Organization-AuthMechanism: 04
X-MS-Exchange-Organization-AuthSource: connhub1.connect.uwaterloo.ca
X-MS-Has-Attach:
X-MS-Exchange-Organization-SCL: -1
X-MS-TNEF-Correlator:
Content-Type: multipart/alternative;
  boundary="_000_748888CA42FDF349AF07A8978DDED060281C9EC0connmbx02_"
MIME-Version: 1.0

--_000_748888CA42FDF349AF07A8978DDED060281C9EC0connmbx02_
Content-Type: text/plain; charset="Windows-1252"
Content-Transfer-Encoding: quoted-printable

Hi Craig,
```

Network Working Group
Request for Comments: 2822
Obsoletes: 822
Category: Standards Track

P. Resnick, Editor
QUALCOMM Incorporated

April 2001

Internet Message Format

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

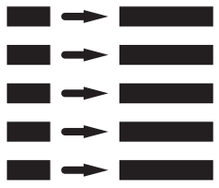
Abstract

This standard specifies a syntax for text messages that are sent between computer users, within the framework of "electronic mail" messages. This standard supersedes the one specified in Request For Comments (RFC) 822, "Standard for the Format of ARPA Internet Text Messages", updating it to reflect current practice and incorporating incremental changes that were specified in other RFCs.

Sequence

```
46.12 47.88 46.32 45.27 44.32 43.87 44.23 42.95 41.74 40.69
41.68 40.73 40.75 40.55 39.39 39.27 40.89 41.22 .      40.57
40.43 40.58 39.93 41.08 40.00 37.64 37.46 37.16 36.76 35.65
36.31 37.32 35.55 34.98 34.72 34.55 36.12 36.76 37.62 .
36.36 37.88 36.59 37.13
```

```
The Right Honourable Justin Trudeau
The Right Honourable Stephen Harper
The Right Honourable Paul Edgar Philippe Martin
The Right Honourable Joseph Jacques Jean Chrétien
The Right Honourable A. Kim Campbell
The Right Honourable Martin Brian Mulroney
The Right Honourable John Napier Turner
The Right Honourable Pierre Elliott Trudeau
The Right Honourable Charles Joseph Clark
The Right Honourable Pierre Elliott Trudeau
The Right Honourable Lester Bowles Pearson
The Right Honourable John George Diefenbaker
The Right Honourable Louis Stephen St-Laurent
The Right Honourable William Lyon Mackenzie King
The Right Honourable Richard Bedford Bennett
The Right Honourable William Lyon Mackenzie King
The Right Honourable Arthur Meighen
```



Dictionary

Associate a set of *keys* with a set of *values*. Ask for the value associated with any key without examining every other key/value pair.

1896	Athens, Greece
1900	Paris, France
1904	St. Louis, United States
1908	London, United Kingdom
1912	Stockholm, Sweden
1920	Antwerp, Belgium
1924	Paris, France
1928	Amsterdam, Netherlands
1932	Los Angeles, United States
1936	Berlin, Germany
1948	London, United Kingdom
1952	Helsinki, Finland
1956	Melbourne, Australia
1960	Rome, Italy
1964	Tokyo, Japan
1968	Mexico City, Mexico
1972	Munich, West Germany
1976	Montréal, Canada
1980	Moscow, Soviet Union
1984	Los Angeles, United States
1988	Seoul, South Korea
1992	Barcelona, Spain
1996	Atlanta, United States
2000	Sydney, Australia
2004	Athens, Greece
2008	Beijing, China
2012	London, United Kingdom
2016	Rio de Janeiro, Brazil
2020	Tokyo, Japan

Table

Your weekly mixtape of fresh music. Enjoy new discoveries and deep cuts chosen just for you. Updated every Monday, so save your favourites!

Created by: Spotify • 30 songs, 2 hr 36 min

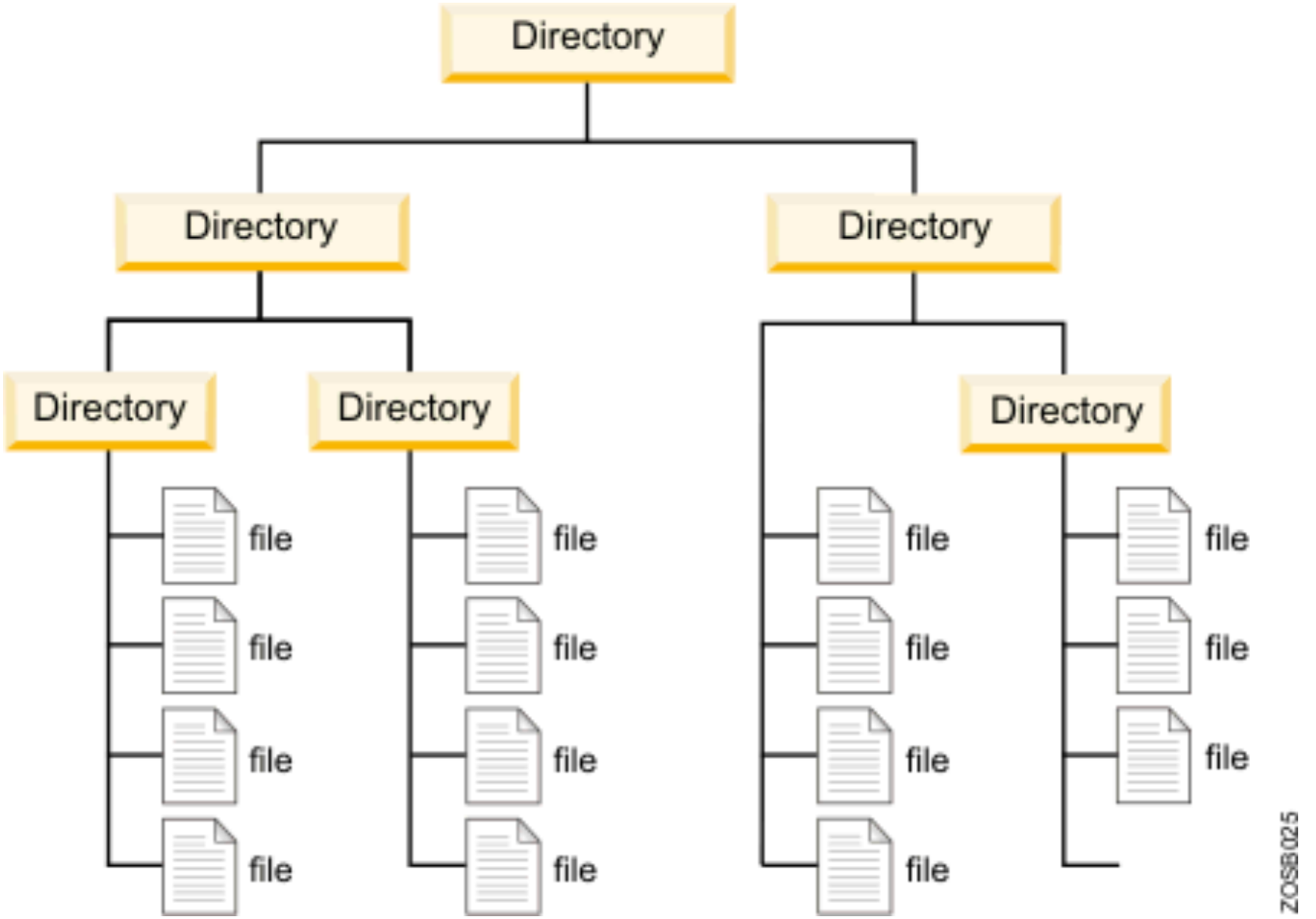
PAUSE FOLLOWING ••• FOLLOWER 1

Filter Download

	SONG	ARTIST	ALBUM		
+	Ways To Go - Margot Mix	Weval, Margot	Weval Remix	11 hours ago	7:11
+	Death Is A Girl	Mini Mansions	The Great Preten...	11 hours ago	4:36
+	Jumbo	Underworld	Beaucoup Fish	11 hours ago	6:58
+	Bug Powder Dust	The Mysterons	Meandering	11 hours ago	4:27
+	...To Have No Answer	Flock of Dimes	If You See Me, Sa...	11 hours ago	3:49
+	I'll Cut You Down	Uncle Acid & The...	Blood Lust	11 hours ago	5:02
+	L'enfer ce n'est pas les autres c'est moi	The Eye Of Time	Myth I: A Last Da...	11 hours ago	5:46
+	Terrain	pg.lost	Key	11 hours ago	5:29

2:46 3:54

Tree


Directory
Directory
Directory
Directory
Directory
Directory
file
file
file
file
file
file
file
file
file
file
file
file
file
file
file
file

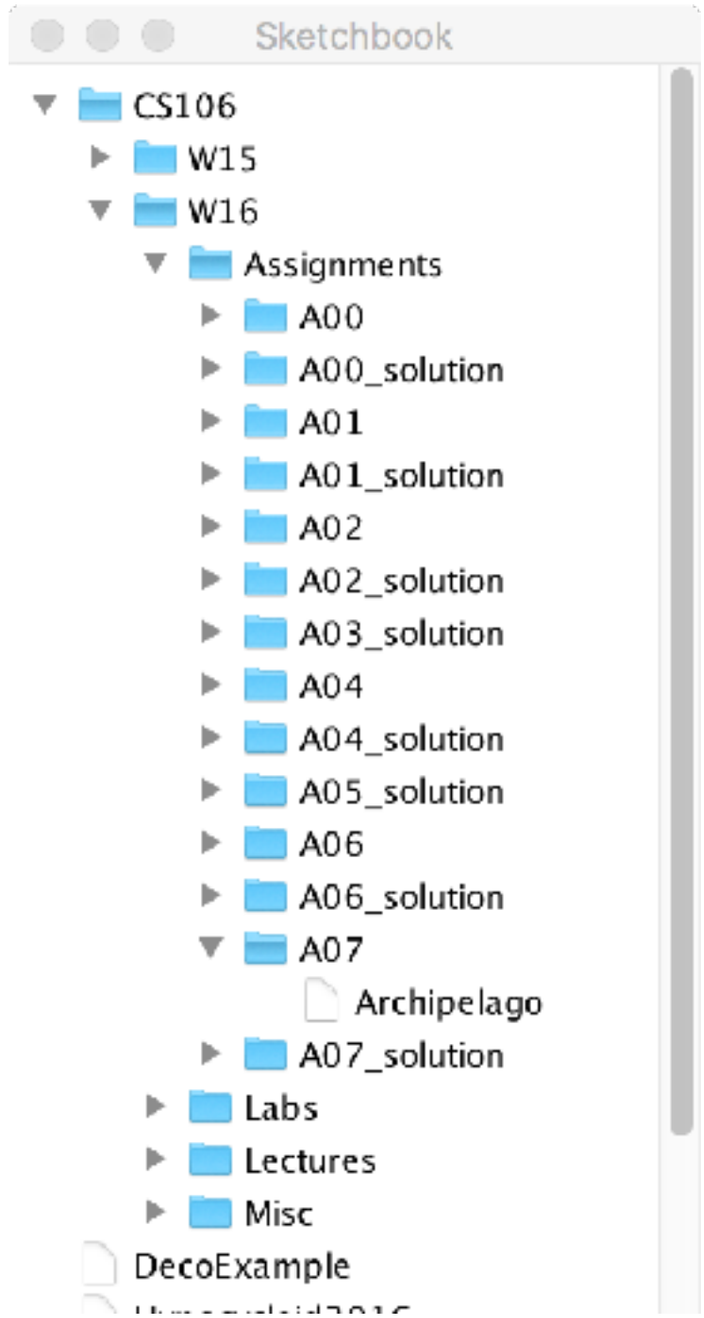

Sketchbook
CS106
W15
W16
Assignments
A00
A00_solution
A01
A01_solution
A02
A02_solution
A03_solution
A04
A04_solution
A05_solution
A06
A06_solution
A07
Archipelago
A07_solution
Labs
Lectures
Misc
DecoExample

Image
save()
saveFrame()

Files
beginRaw()
beginRecord()
createOutput()
createWriter()
endRaw()
endRecord()
PrintWriter
saveBytes()
saveJSONArray()
saveJSONObject()
saveStream()
a.ref-link | 204 × 18
saveXML()
selectOutput()

Calculation
abs()
ceil()
constrain()
dist()
exp()
floor()
lerp()
log()
mag()
map()
max()
min()
norm()
pow()
round()
sq()
sqrt()

Trigonometry
acos()
asin()
atan()
atan2()
cos()
degrees()
radians()
sin()
tan()

Random
noise()
noiseDetail()
noiseSeed()
random()
randomGaussian()
randomSeed()

to)

ual to)

Transform

applyMatrix()
popMatrix()
printMatrix()
pushMatrix()
resetMatrix()
rotate()
rotateX()
rotateY()
rotateZ()
scale()
shearX()
shearY()
translate()

Elements Console Sources Network Timeline » 1

```
<a href="beginRaw_.html" class=" ref-link">beginRaw()</a>
<a href="beginRecord_.html" class=" ref-link">beginRecord()</a>
<a href="createOutput_.html" class=" ref-link">createOutput()</a>
<a href="createWriter_.html" class=" ref-link">createWriter()</a>
<a href="endRaw_.html" class=" ref-link">endRaw()</a>
<a href="endRecord_.html" class=" ref-link">endRecord()</a>
<a href="PrintWriter.html" class=" ref-link">PrintWriter</a>
<a href="saveBytes_.html" class=" ref-link">saveBytes()</a>
<a href="saveJSONArray_.html" class=" ref-
link">saveJSONArray()</a>
<a href="saveJSONObject_.html" class=" ref-
link">saveJSONObject()</a>
<a href="saveStream_.html" class=" ref-link">saveStream()</a>
<a href="saveStrings_.html" class=" ref-link">saveStrings()</a>
<a href="saveTable_.html" class=" ref-link">saveTable()</a>
<a href="saveXML_.html" class=" ref-link">saveXML()</a>
<a href="selectOutp                                    ut()</a>
</div>
▶<div class="category">…</div>
▶<div class="category">…</div>
</div>
▶<div class="ref-col">…</div>
</div>
<!-- ================================== FOOTER
```

https://processing.org/reference/saveXML_.html

html.js.no-touch body#language

Styles Event Listeners DOM Breakpoints Properties

Filter :hov .cls +

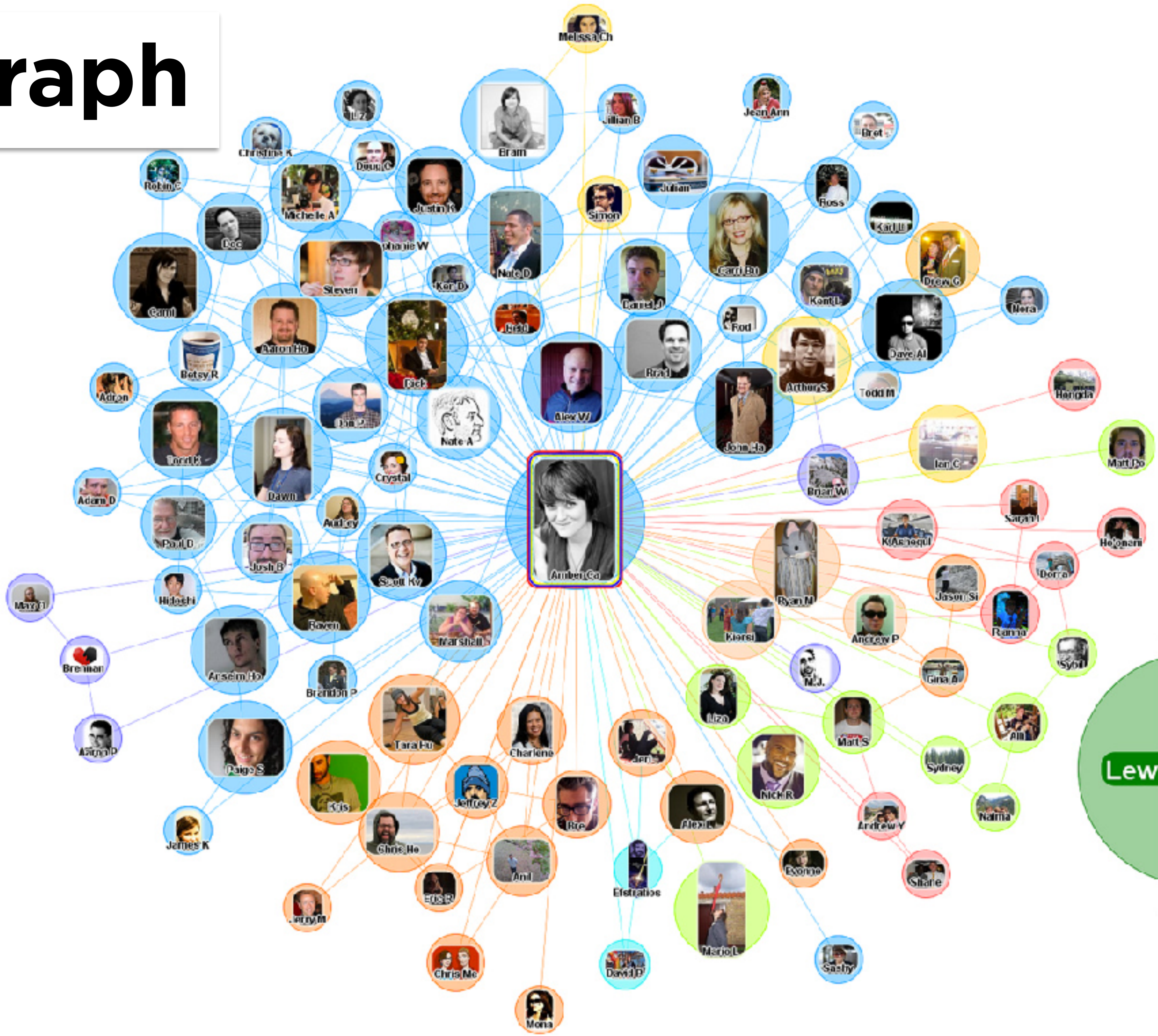
```
element.style {
}
body {                                          style.css:60
  margin:▶0;
  padding:▶0;
  overflow-y: scroll;
  background-color: #ddd;
  font-family: 'theSerif', 'Enriqueta',
    georgia, times, serif;
  -webkit-font-smoothing: antialiased;
  -webkit-text-size-adjust: none;
  font-size: 100%;
  font-size: 0.79em;
  font-weight: normal;
  line-height: 1.5em;
  color: #252525;
}
body {                          user agent stylesheet
```

margin -
border -
padding -
736 × 3824.200

Filter Show all

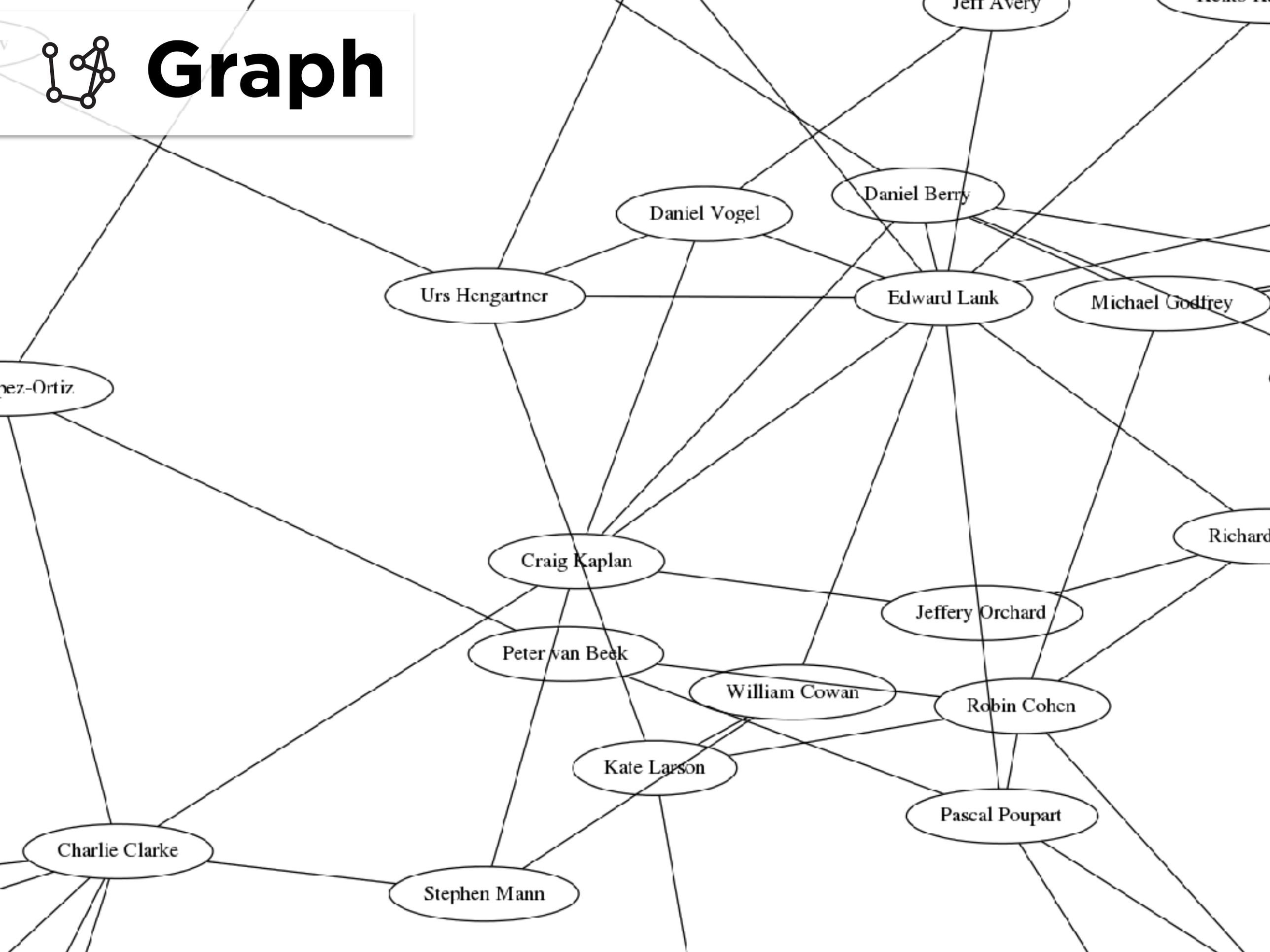
▶ background-colo…	rgb(2…
▶ color	rgb(3…
▶ display	block
▶ font-family	theSeri…
▶ font-size	12.64px

Graph



Melissa Ch
Bram
Jillian B
Jean Ann
Julian
Simon
Ross
Doc
Steven
Aaron Ho
Betsy R
Dick
Nate D
Alex W
Brad
John Ha
Arthur S
Todd M
Dave Al
Drew G
Nora
Nate A
Crystal
Dawn
Paul D
Josh B
Scott Ky
Raven
Marshall
Anselm Ho
Brandon P
Paige S
James K
Amber Ca
Ryan N
Kiersi
Andrew P
Ian C
Brian W
Tara Hu
Charlene
Kris
Jeffrey Z
Chris He
Mona
Liza
Matt S
Sydney
Nick R
Alex L
Efstratios
David P
Mario L
Andrew Y
Shane

Graph


Jeff Avery
Daniel Vogel
Daniel Berry
Urs Hengartner
Edward Lank
Michael Godfrey
pez-Ortiz
Craig Kaplan
Richard
Jeffery Orchard
Peter van Beek
William Cowan
Robin Cohen
Kate Larson
Pascal Poupart
Charlie Clarke
Stephen Mann

String operations

```
String wd = "…";
int len = wd.length();
char c = wd.charAt(2);
String str3 = str1 + str2;
if( str1.equals( str2 ) ) { … }
String[] words = splitTokens( str1 );
```

String operations

```
String wd = "…";
int len = wd.length();
char c = wd.charAt(2);
String str3 = str1 + str2;
if( str1.equals( str2 ) ) { … }
String[] words = splitTokens( str1 );
```

Initialize a variable from a string literal

String operations

```
String wd = "…";
int len = wd.length();
char c = wd.charAt(2);
String str3 = str1 + str2;
```

Count the number of characters in a string

```
if( str1.equals( str2 ) ) { … }
String[] words = splitTokens( str1 );
```

String operations

```
String wd = "…";

int len = wd.length();

char c = wd.charAt(2);

String str3 = str1 + str2;

if( str1.equals( str2 ) ) { … }

String[] words = splitTokens( str1 );
```

Extract a character from a string. Like accessing an array

String operations

```
String wd = "…";

int len = wd.length();

char c = wd.charAt(2);

String str3 = str1 + str2;

if( str1.equals( str2 ) ) { … }

String[] words = splitTokens( str1 );
```

Glue two strings together

String operations

```
String wd = "…";
int len = wd.length();
char c = wd.charAt(2);
String str3 = str1 + str2;
```

Check if two strings have the same characters

```
if( str1.equals( str2 ) ) { … }
String[] words = splitTokens( str1 );
```

String operations

```
String wd = "…";

int len = wd.length();

char c = wd.charAt(2);

String str3 = str1 + str2;

if( str1.equals( str2 ) ) { … }

String[] words = splitTokens( str1 );
```

Break a string into words
by looking for whitespace

Messier text

```
String[] splitTokens( String text, String delims ) { … }
```

Break the long string `text` into “words”, where the characters in `delims` (and not whitespace) are treated as breakpoints.

```
String trim( String text ) { … }
```

Return a copy of `text` with any excess whitespace removed from the start and end.

Example: the Region of Waterloo's list of reserved street names

FullStreetName	Municipality
Abbey Glen	Kitchener
Aberle	Woolwich
Abeth	Kitchener
Abitibi	Cambridge
Able	Cambridge
Abram Clemens St	Kitchener
Accobee	Cambridge
Adair	Cambridge
Adcock	Region of Waterloo
Addley	Cambridge

www.regionofwaterloo.ca/en/regionalgovernment/reservedstreetnamesdataset.asp

Reading the dictionary

```
A
a
aa
aal
aalii
aam
Aani
aardvark
aardwolf
Aaron
Aaronic
Aaronical
Aaronite
Aaronitic
Aaru
Ab
aba
Ababdeh
Ababua
abac
abaca
abacate
abacay
abacinate
abacination
abaciscus
```

Reading the dictionary

```
A
a
aa
aal
aalii
aam
Aani
aardvark
aardwolf
Aaron
Aaronic
Aaronical
Aaronite
Aaronitic
Aaru
Ab
aba
Ababdeh
Ababua
abac
abaca
abacate
abacay
abacinate
abacination
abaciscus
```

Find the longest word

Find all words with three or more Ys

Find all words ending with MT

Find all words starting with TM

Find all words ending with DOUS

Find all words containing UFA

Find all words ending in GRY

Find all palindromes

Find words with three consecutive double letters

Find the longest word whose letters are in alphabetical order

Find the longest word with no

Finding things in strings

```
if( str.contains( "abc" ) ) { … }
```

Check if the string `str` has the substring "abc" anywhere inside of it.

```
if( str.startsWith( "def" ) ) { … }
if( str.endsWith( "ghi" ) ) { … }
```

Look for a substring specifically at the start or end of a string.

Writing a spellchecker

With the dictionary at our disposal, it's easy to check if a given string is a word.

```
String[] dict;

void setup() {
  dict = loadStrings( “words.txt” );
}

boolean isWord( String word ) {




}
```

Writing a spellchecker

With the dictionary at our disposal, it's easy to check if a given string is a word.

```
String[] dict;

void setup() {
  dict = loadStrings( “words.txt” );
}

boolean isWord( String word ) {
  for ( int idx = 0; idx < dict.length; ++idx ) {
    if ( dict[idx].equals( word ) ) {
      return true;
    }
  }
  return false;
}
```

```
boolean isWord( String word ) {
  for ( int idx = 0; idx < dict.length; ++idx ) {
    if ( dict[idx].equals( word ) ) {
      return true;
    }
  }
  return false;
}
```

Looping over every word works, but it's painfully slow, *especially* when the word isn't there!

The function `join()` is like the reverse of `splitTokens()`: it turns an array of strings into one long string, using a given delimiter string.

```
String[] things = {
  "Kumquat", "Durian", "Rambutan", "Lychee" };

println( join( things, " " ) );


println( join( things, " and " ) );
```

The function `join()` is like the reverse of `splitTokens()`: it turns an array of strings into one long string, using a given delimiter string.

```
String[] things = {
  "Kumquat", "Durian", "Rambutan", "Lychee" };

println( join( things, " " ) );

    ⇒ Kumquat Durian Rambutan Lychee

println( join( things, " and " ) );
```

The function `join()` is like the reverse of `splitTokens()`: it turns an array of strings into one long string, using a given delimiter string.

```
String[] things = {
  "Kumquat", "Durian", "Rambutan", "Lychee" };

println( join( things, " " ) );

   ⇒ Kumquat Durian Rambutan Lychee

println( join( things, " and " ) );

   ⇒ Kumquat and Durian and Rambutan and Lychee
```

Dictionaries

In programming, a *dictionary* is a mapping from a set of *keys* to a set of *values*. Any given key may have at most one associated value.

Year	→	Olympic host city
Name	→	Phone number
Student ID number	→	Exam seating code
Clicker ID	→	Student ID number
Server name	→	IP address

Dictionaries

Dictionary operations we might care about:

- Look up the value associated with a given key
- Ask if the dictionary has a given key
- Add a new key to the dictionary, with its associated value
- Remove a key and its value from the dictionary

Processing includes a few handy dictionary classes, where the keys are `String`s:

- `IntDict`: **map** `String`**s to** `int`**s**
- `FloatDict`: **map** `String`**s to** `float`**s**
- `StringDict`: **map** `String`**s to** `String`**s**

Processing includes a few handy dictionary classes, where the keys are `String`s:

- `IntDict`: **map** `String`**s to** `int`**s**
- `FloatDict`: **map** `String`**s to** `float`**s**
- `StringDict`: **map** `String`**s to** `String`**s**

Java allows more-or-less arbitrary mappings between keys and values:

- `java.util.TreeMap<K,V>`: **map any key type** `K` **to any value type** `V`**.**

```
IntDict myDict = new IntDict();
```

Create a new, empty dictionary

```
IntDict myDict = new IntDict();
```

Create a new, empty dictionary

```
myDict.set( "Kumquat", 13 )
myDict.set( "Durian", 19 );
```

Add a new key to the dictionary, with its associated value

```
IntDict myDict = new IntDict();
```

Create a new, empty dictionary

```
myDict.set( "Kumquat", 13 )
myDict.set( "Durian", 19 );
```

Add a new key to the dictionary, with its associated value

```
println( myDict.get( "Kumquat" ) );
```

Look up the value associated with a given key

```
IntDict myDict = new IntDict();
```

Create a new, empty dictionary

```
myDict.set( "Kumquat", 13 )
myDict.set( "Durian", 19 );
```

Add a new key to the dictionary, with its associated value

```
println( myDict.get( "Kumquat" ) );
```

Look up the value associated with a given key

```
if( myDict.hasKey( "Rambutan" ) ) { ... }
```

Ask if the dictionary has a given key

```
IntDict myDict = new IntDict();
```

Create a new, empty dictionary

```
myDict.set( "Kumquat", 13 )
myDict.set( "Durian", 19 );
```

Add a new key to the dictionary, with its associated value

```
println( myDict.get( "Kumquat" ) );
```

Look up the value associated with a given key

```
if( myDict.hasKey( "Rambutan" ) ) { ... }
```

Ask if the dictionary has a given key

```
myDict.remove( "Durian" );
```

Remove a key and its value from the dictionary

Writing a spellchecker

```
String[] dict;

void setup() {
  dict = loadStrings( “words.txt” );
}
```

```
boolean isWord( String word ) {
  for ( int idx = 0; idx < dict.length; ++idx ) {
    if ( dict[idx].equals( word ) ) {
      return true;
    }
  }
  return false;
}
```

Writing a spellchecker

```
IntDict myDict;

void setup()
{
  String[] words = loadStrings( “words.txt” );
  for( int idx = 0; idx < words.length; ++idx ) {
    myDict.set( words[idx], 1 );
  }
}

boolean isWord( String word )
{
  return myDict.hasKey( word );
}
```

Writing a spellchecker

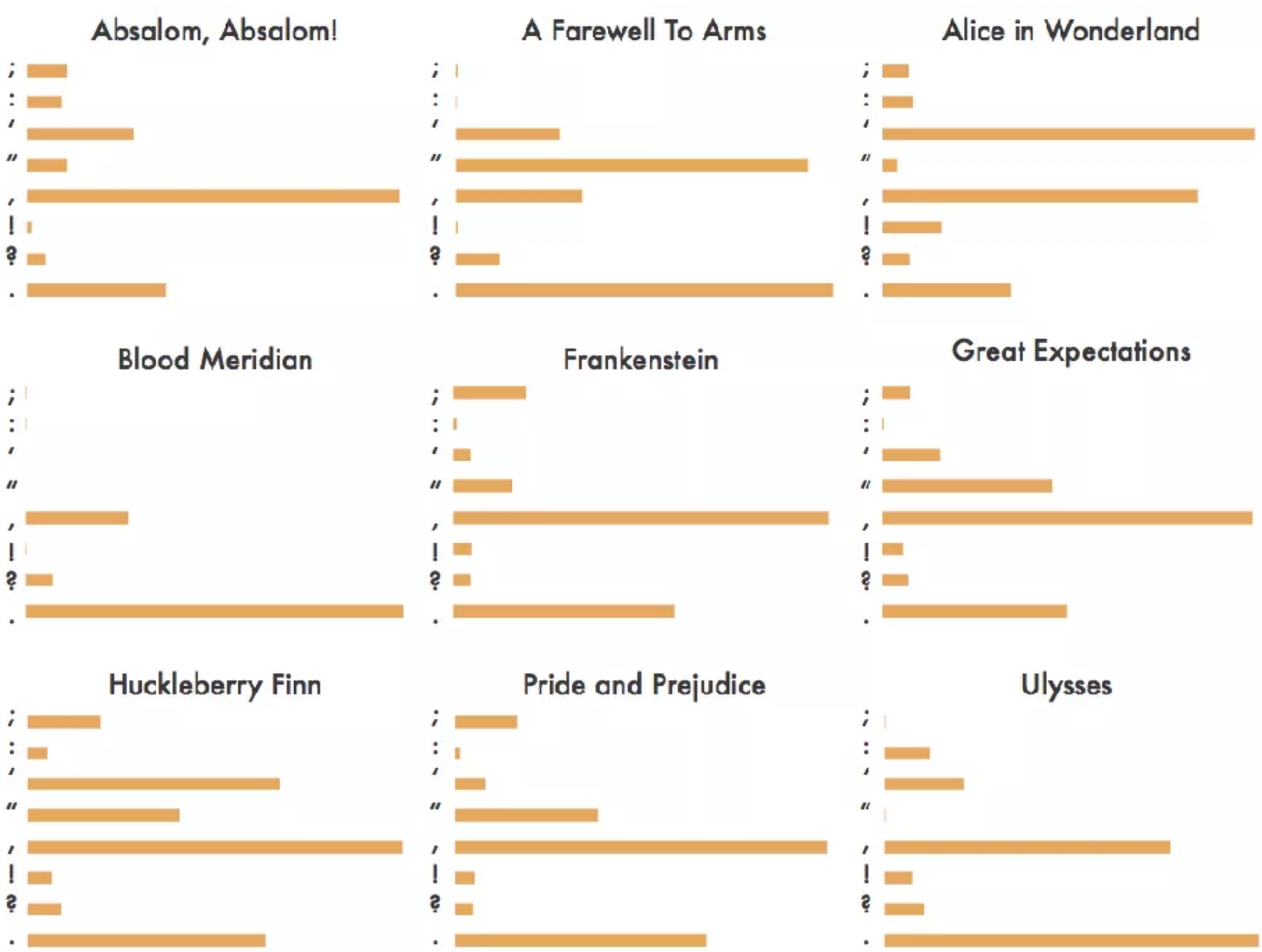
```
IntDict myDict;

void setup()
{
  String[] words =
  for( int idx = 0; idx < words.length; ++idx ) {
    myDict.set( words[idx], 1 );
  }
}

boolean isWord( String word )
{
  return myDict.hasKey( word );
}
```

These are guaranteed to be **fast!**

Counting things



Finding patterns

It’s easy to search a string for a given phone number:

```
if( myString.contains( "(519) 888-4567" ) ) { … }
```

But what if we wanted to find all the phone numbers in a string?

Finding patterns

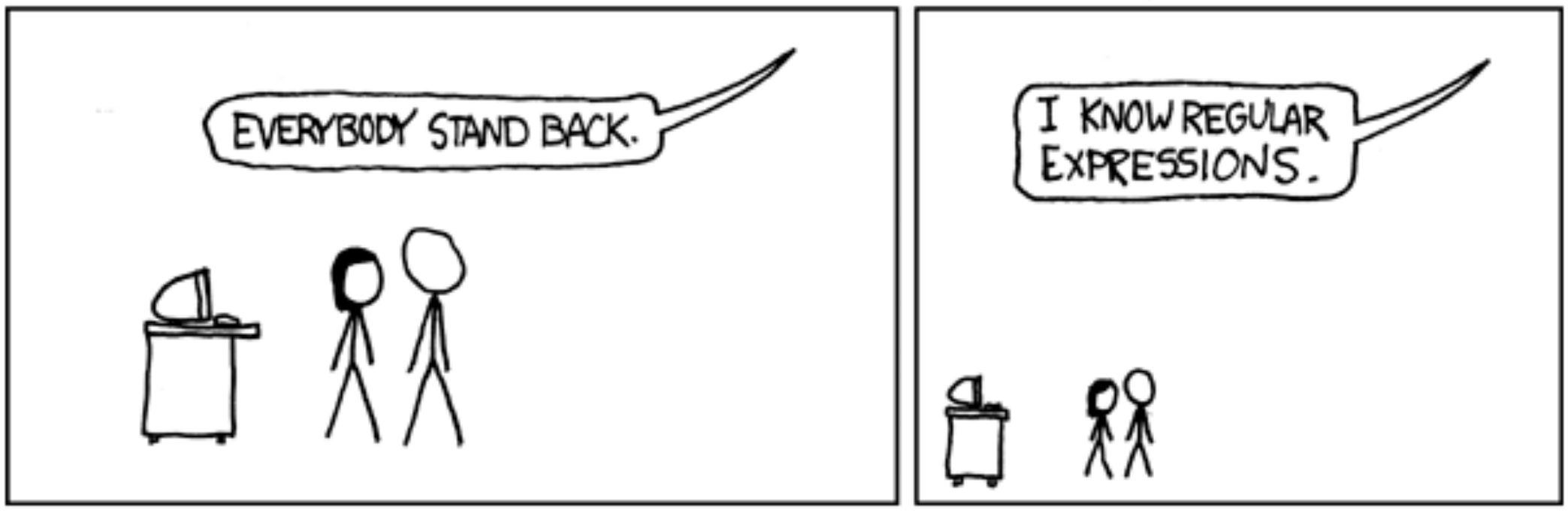
Regular Expressions are a general tool for finding patterns in strings.

Finding patterns

Regular Expressions are a **programming language** for finding patterns in strings.

Finding patterns

Regular Expressions are a **cryptic programming language** for finding patterns in strings.



xkcd.com/208/

```
String[] match( String text, String pattern ) { ... }
```

Look for an instance of the regular expression `pattern` **inside of the string** `text`**. If the answer is not** `null`**, the pattern was found.**

Regular Expressions - Quick Reference Guide

Anchors

^	start of line
$	end of line
\b	word boundary
\B	not at word boundary
\A	start of subject
\G	first match in subject
\z	end of subject
\Z	end of subject or before newline at end

Non-printing characters

\a	alarm (BEL, hex 07)
\cx	"control-x"
\e	escape (hex 1B)
\f	formfeed (hex 0C)
\n	newline (hex 0A)
\r	carriage return (hex 0D)
\t	tab (hex 09)
\ddd	octal code ddd
\xhh	hex code hh
\x{hhh..}	hex code hhh..

Generic character types

\d	decimal digit
\D	not a decimal digit
\s	whitespace character
\S	not a whitespace char
\w	"word" character
\W	"non-word" character

POSIX character classes

alnum	letters and digits
alpha	letters
ascii	character codes 0-127
blank	space or tab only
cntrl	control characters
digit	decimal digits
graph	printing chars -space
lower	lower case letters
print	printing chars +space
punct	printing chars -alnum
space	white space
upper	upper case letters
word	"word" characters
xdigit	hexadecimal digits

Literal Characters

Letters and digits match exactly	a x B 7 0
Some special characters match exactly	@ - = %
Escape other specials with backslash	\. \\ \$ \[

Character Groups

Almost any character (usually not newline)	.
Lists and ranges of characters	[ ]
Any character except those listed	[^]

Counts (add ? for non-greedy)

0 or more ("perhaps some")	*
0 or 1 ("perhaps a")	?
1 or more ("some")	+
Between "n" and "m" of	{n,m}
Exactly "n", "n" or more	{n}, {n,}

Alternation

Either/or	|

Lookahead and Lookbehind

Followed by	(?=)
NOT followed by	(?!)
Following	(?<=)
NOT following	(?<!)

Grouping

For capture and counts	( )
Non-capturing	(?:)
Named captures	(?<name>)

Back references

Numbered	\n \gn \g{n}
Relative	\g{-n}
Named	\k<name>

Character group contents

x	individual chars
x-y	character range
[:class:]	posix char class
[^:class:]	negated class

Examples

[a-zA-Z0-9_]
[[:alnum:]_]

Comments

(?#comment)

Conditional subpatterns

(?(condition)yes-pattern)
(?(condition)yes|no-pattern)

Recursive patterns

(?n)	Numbered
(?0) (?R)	Entire regex
(?&name)	Named

Replacements

$n	reference capture

Case foldings

\u	upper case next char
\U	upper case following
\l	lower case next char
\L	lower case following
\E	end case folding

Conditional insertions

(?n:insertion)
(?n:insertion:otherwise)

Substring “ufa” anywhere in a word:

```
ufa
```

Word ending in “mt”:

```
mt$
```

Word with three or more “y”s, on a line by itself:

```
y.*y.*y
```

An integer:

```
^(-?[1-9]+\d*)$|^0$
```

An email address:

```
\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b
```

A URL:

```
^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?$
```

A regular expression is like a little “machine”:

```
^(-?[1-9]+\d*)$|^0$
```