Module 10

Data Processing and Text

CS 106 Winter 2019

Data challenges

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

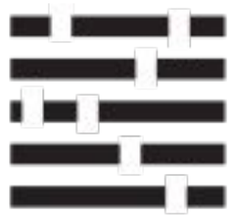
Data challenges

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

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

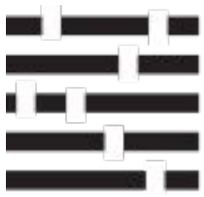
Sequence

Dictionary

Table

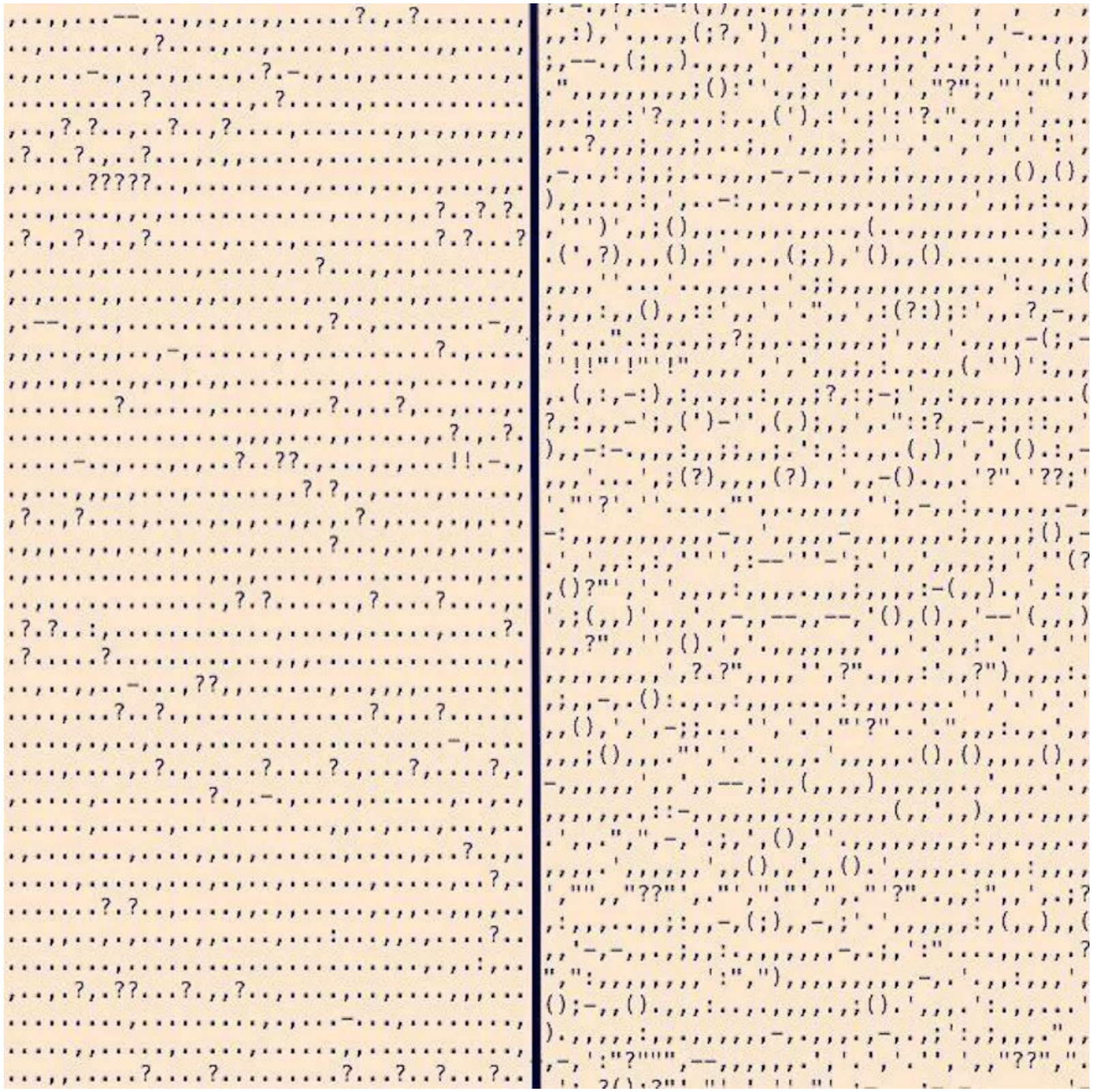
Tree

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 this. If they but knew it, almost all

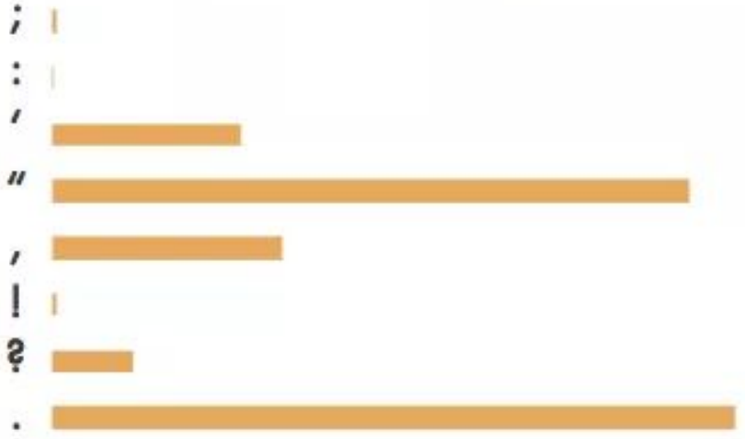


McCarthy Faulkner

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

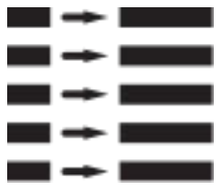
```
 Tue, 17 Jan 2017 15:57:38 -0500
From: Rishabh Moudgil <rishabh.moudgil@uwaterloo.ca>
To: Craig Kaplan <csk@uwaterloo.ca>
CC: Kevin Harrigan <kevinh@uwaterloo.ca>, Kristina Bayda
   <kbayda@uwaterloo.ca>, Travis Bartlett <travis.bartlett@uwaterloo.ca>
Subject: A01 Marking Scheme
Thread-Topic: A01 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
```

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

ixtape of fresh music. Enjoy new discoveries and deep cuts chosen just for you. Updated
so save your favourites!

Spotify • 30 songs, 2 hr 36 min

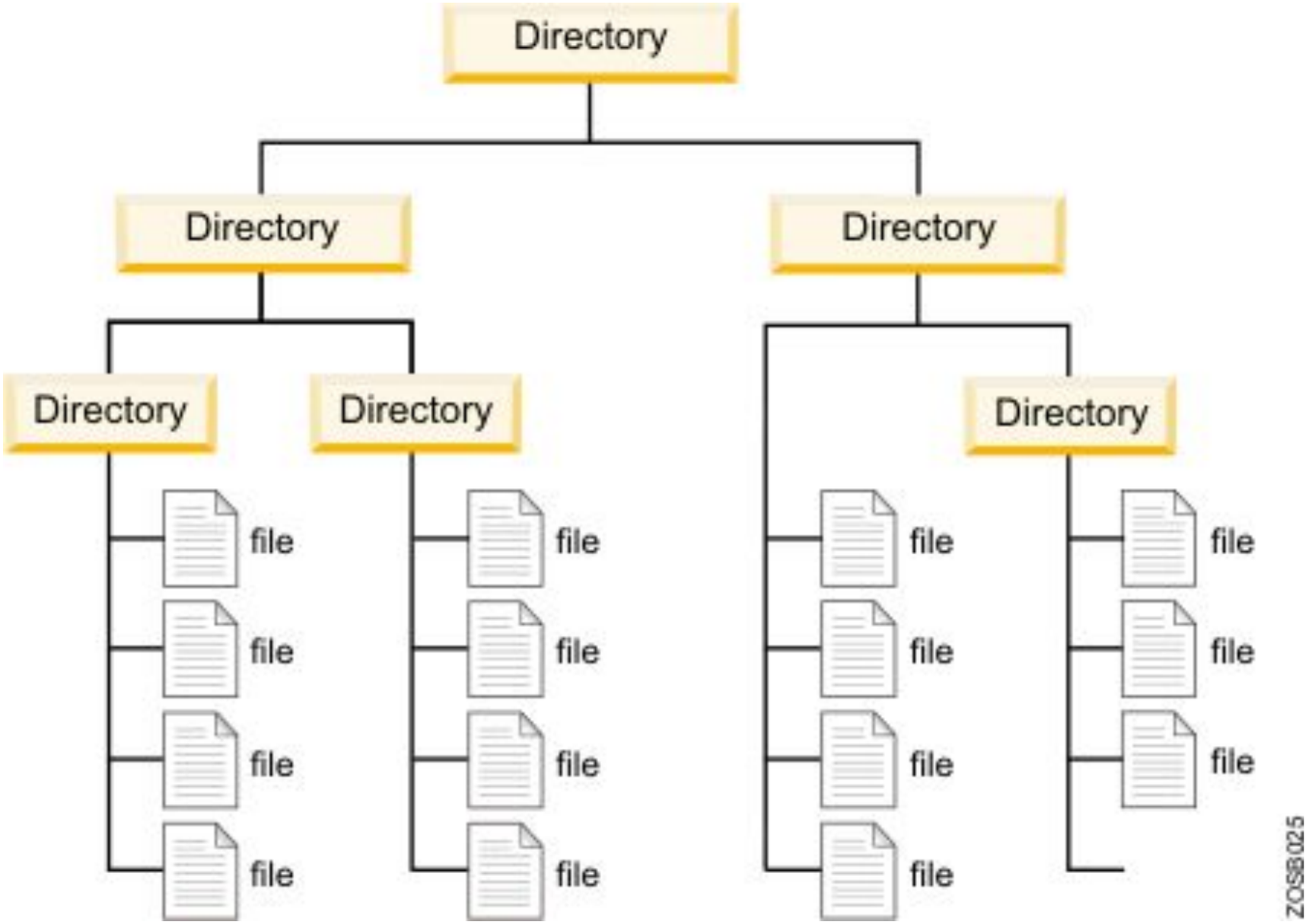
PAUSE FOLLOWING ••• FOLLOWER 1

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
ZOSB025

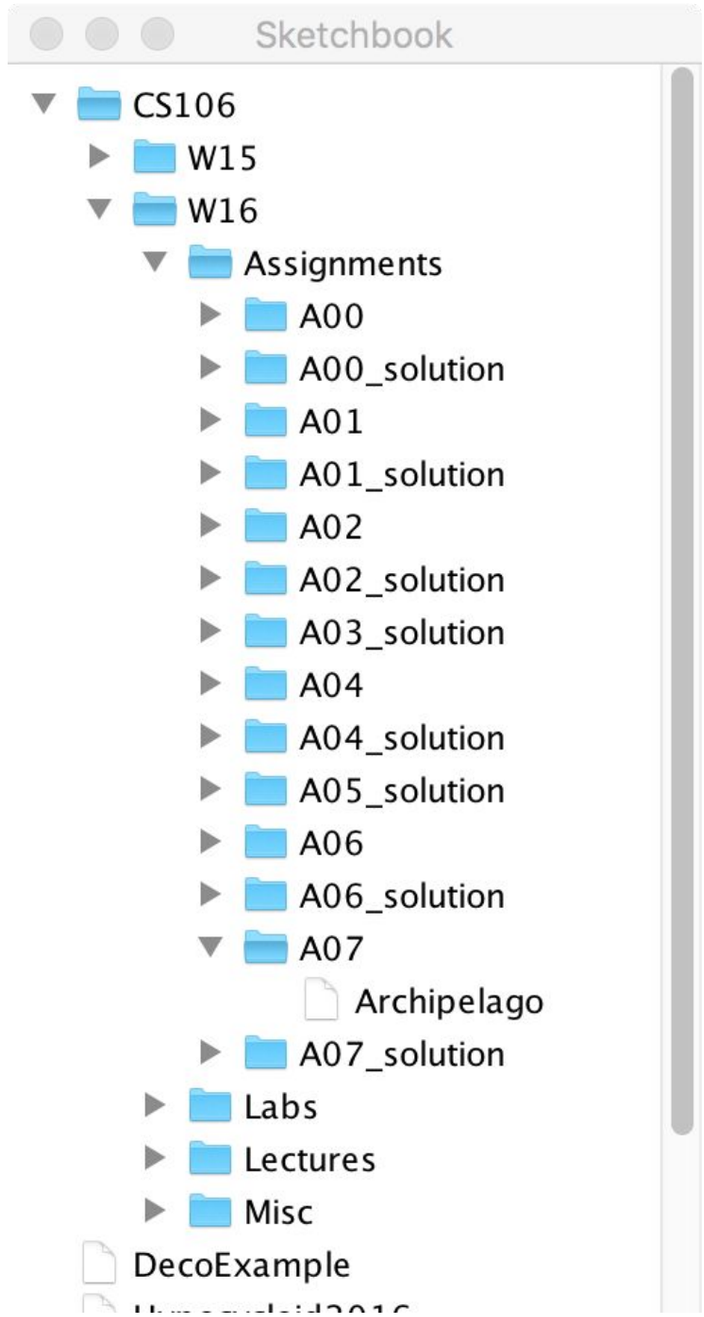

Sketchbook
CS106
W15
W16
Assignments
A00
A00_solution
A01
A01_solution
A02
A02_solution
A03_solution
A04
A04_solution
A05_solution
A06
A06_solution
A07
Archipelago
A07_solution
Labs
Lectures
Misc
DecoExample

Image
save()
saveFrame()

Files
beginRaw()
beginRecord()
createOutput()
createWriter()
endRaw()
endRecord()
PrintWriter
saveBytes()
saveJSONArray()
saveJSONObject()
saveStream()
saveXML()
selectOutput()

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

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()

a.ref-link | 204 × 18

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 +

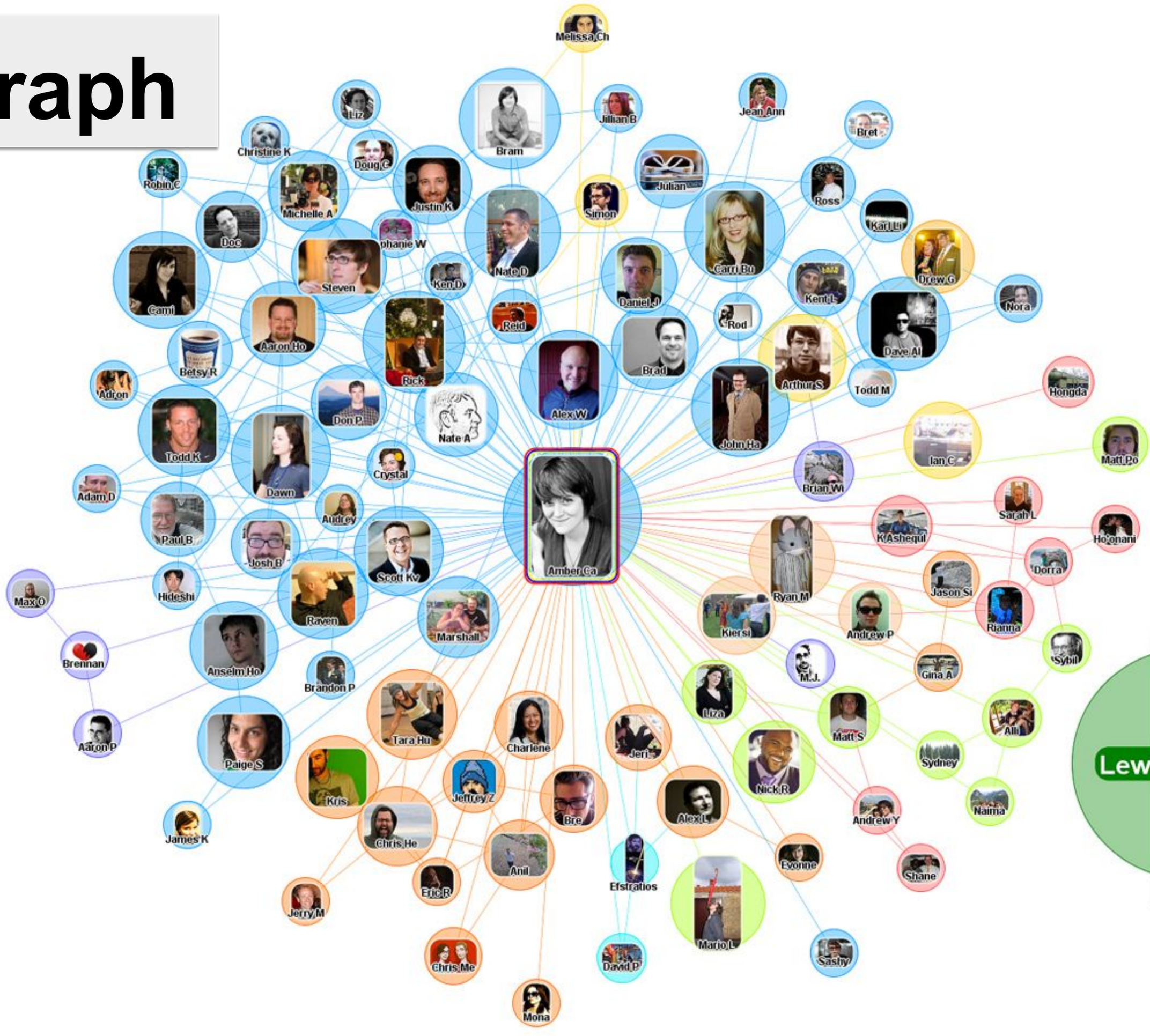
```
element.style {
}
body {                                          style.css:60
  margin: ▶0;
  padding: ▶0;
  overflow-y: scroll;
  background-color: #ddd;
  font-family: 'theSerif', 'Enriqueta',
    georgia, times, serif;
  -webkit-font-smoothing: antialiased;
  -webkit-text-size-adjust: none;
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

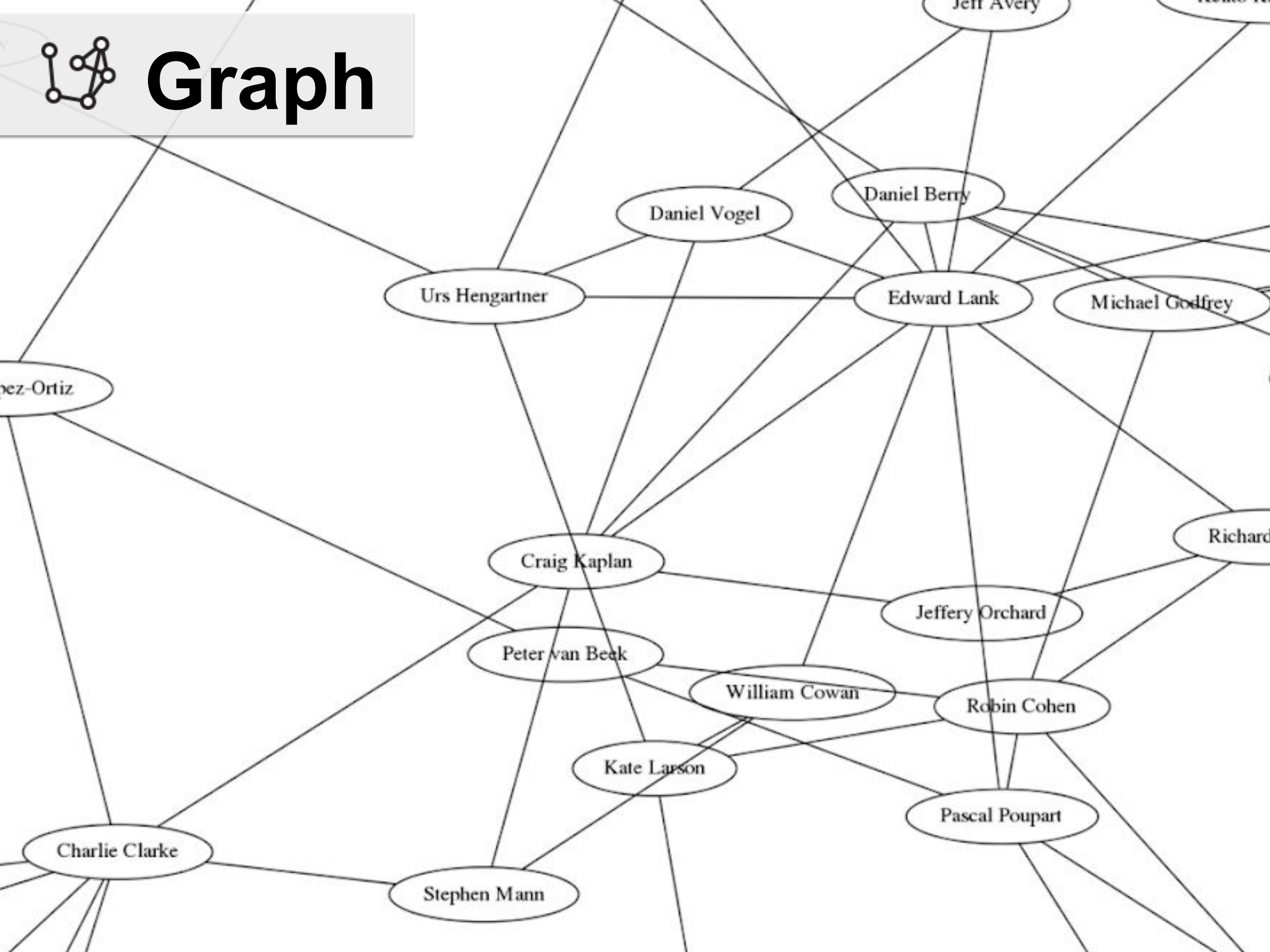
margin - border - padding - 736 × 3824.200

Filter Show all

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

Graph


Melissa Ch
Liz
Bram
Jillian B
Jean Ann
Bret
Christine K
Doug C
Julian
Robin C
Michelle A
Justin K
Simon
Ross
Karl Li
Doc
Stephanie W
Nate D
Carri Bu
Drew G
Cami
Steven
Ken D
Daniel J
Kent L
Nora
Aaron Ho
Reid
Rod
Dave Al
Betsy R
Rick
Brad
Arthur S
Todd M
Hongda
Adron
Don P
Alex W
Nate A
John Ha
Ian C
Matt Po
Todd K
Crystal
Brian Wi
Adam D
Dawn
Audrey
Sarah L
Paul B
Amber Ca
Ryan M
K'Ashequl
Ho'onani
Josh B
Scott Kv
Dorra
Jason Si
Hideshi
Raven
Kiersi
Rianna
Max O
Marshall
Andrew P
Brennan
Anselm Ho
Brandon P
M.J.
Gina A
Sybil
Liza
Aaron P
Tara Hu
Charlene
Jeri
Matt S
Alli
Paige S
Kris
Jeffrey Z
Nick R
Sydney
Lew
Bre
Alex L
Nairna
James K
Chris He
Andrew Y
Anil
Evonne
Efstratios
Shane
Eric R
Jerry M
Mario L
Chris Me
David P
Sashy
Mona


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);
```

Initialize a variable from a string literal

```
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);
```

Count the number of characters in a string

```
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 );
```

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;

if( str1.equals( str2 ) ) { … }

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

Check if two strings have the same characters

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

Abitibi	Cambridge
Able	Cambridge
Abram Clemens St	Kitchener
Accobee	Cambridge
Adair	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
```

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

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, " " ) );
        ⇒ 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

```
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

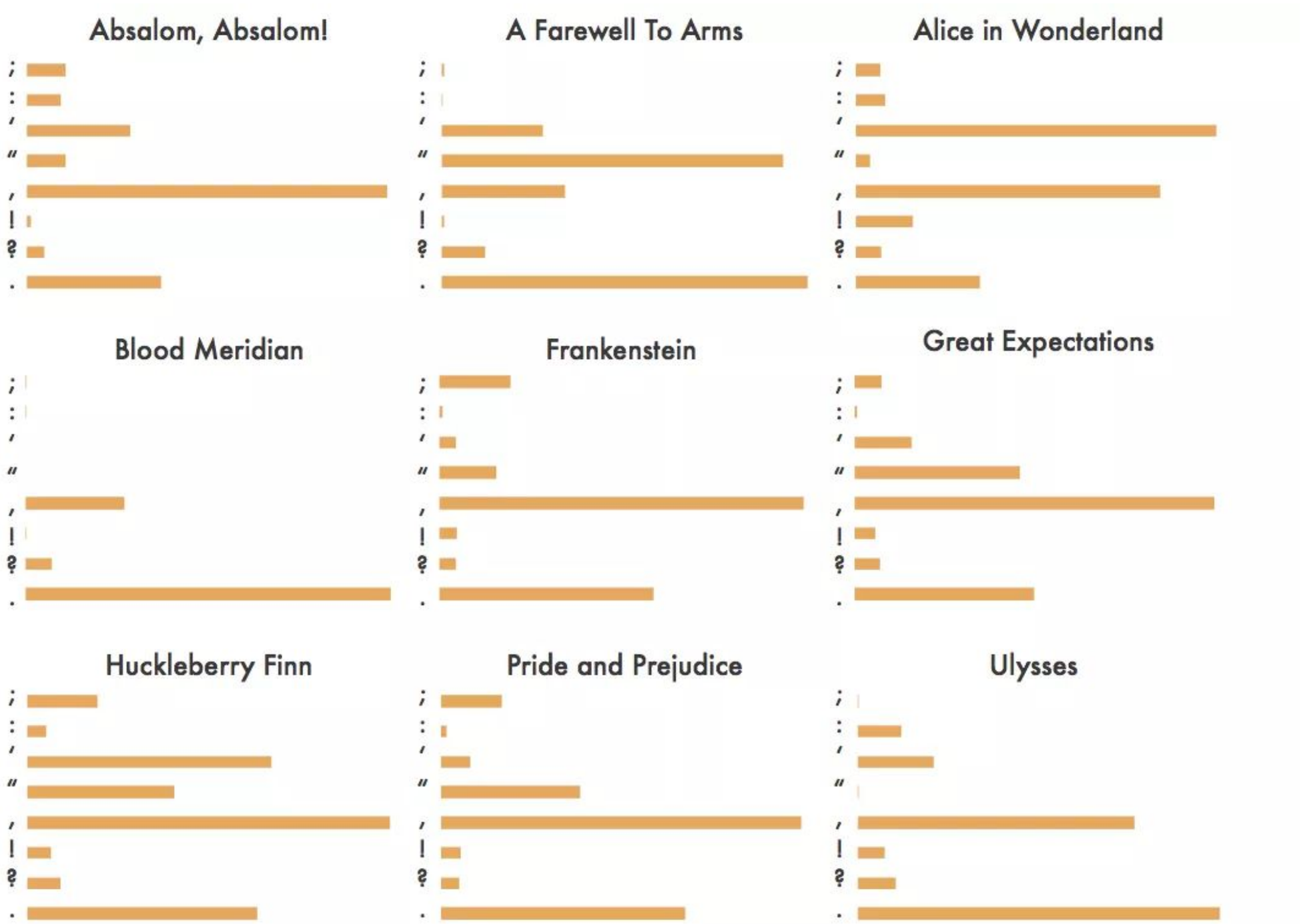
Dict is guaranteed to be **fast**!

```
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 );
}
```

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

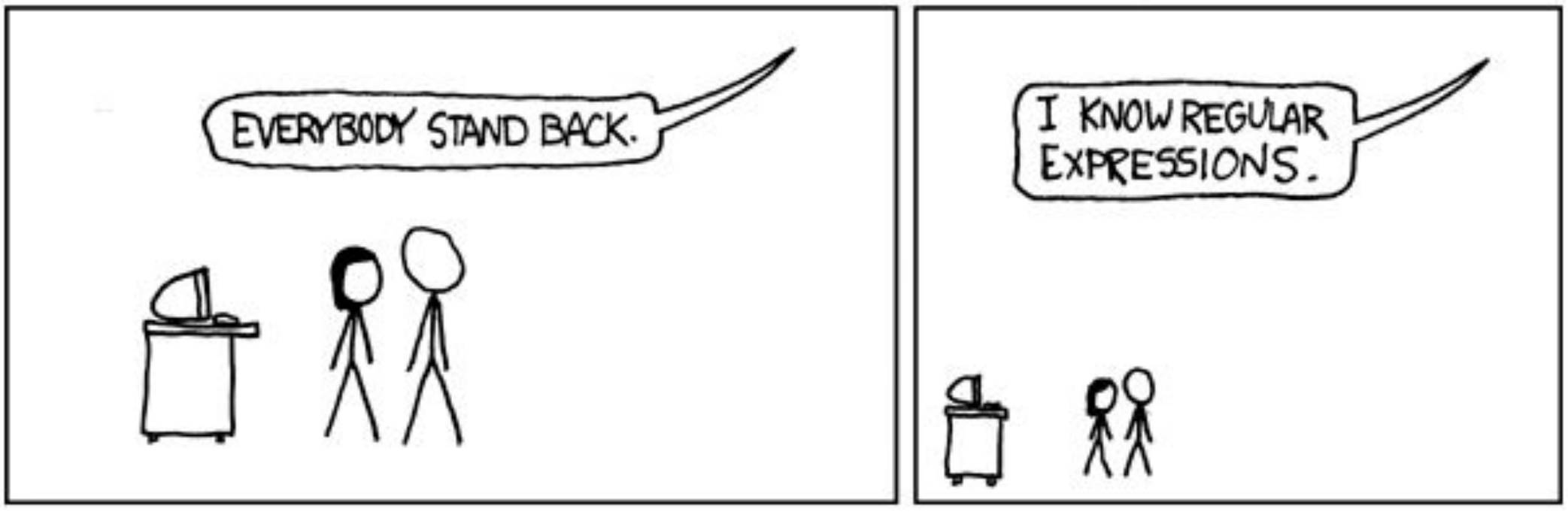
Regular Expressions are a general tool for finding patterns in strings.

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 OD)
\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*)$|^0$
```