Module 10

Data Processing and Text

CS 106 Winter 2019

6 W20

6 W20

Data challenges

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

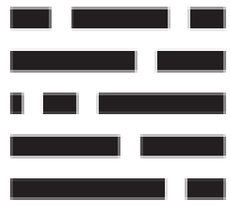
Data challenges

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

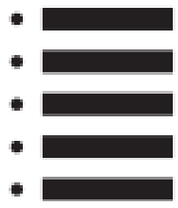
The shape of data

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

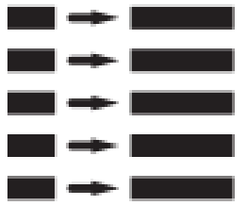
These questions profoundly affect the tools you use and the code you write.



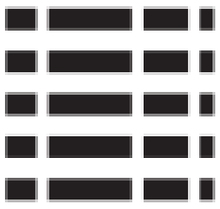
Raw text



Sequence



Dictionary



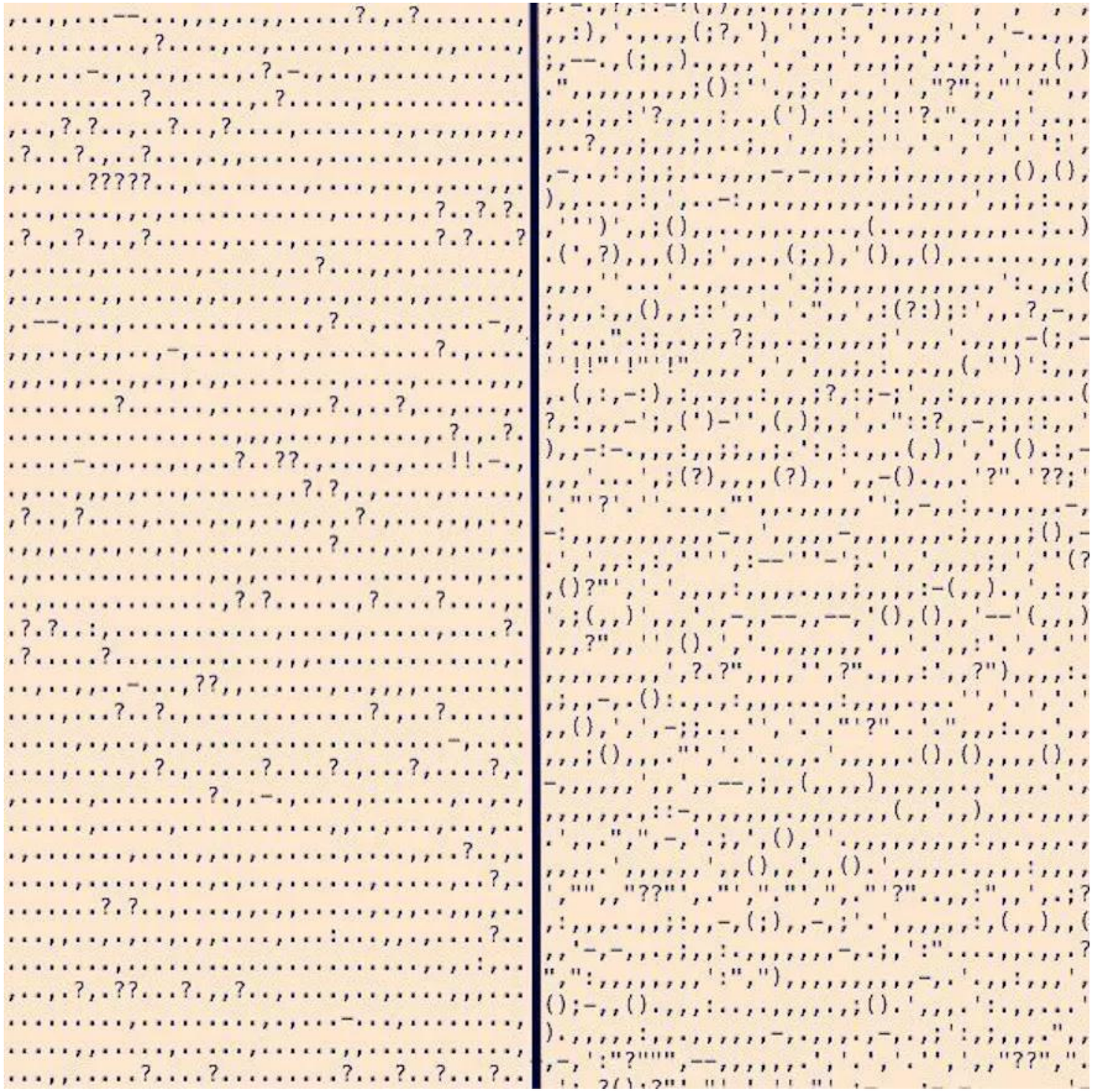
Table

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



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

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

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

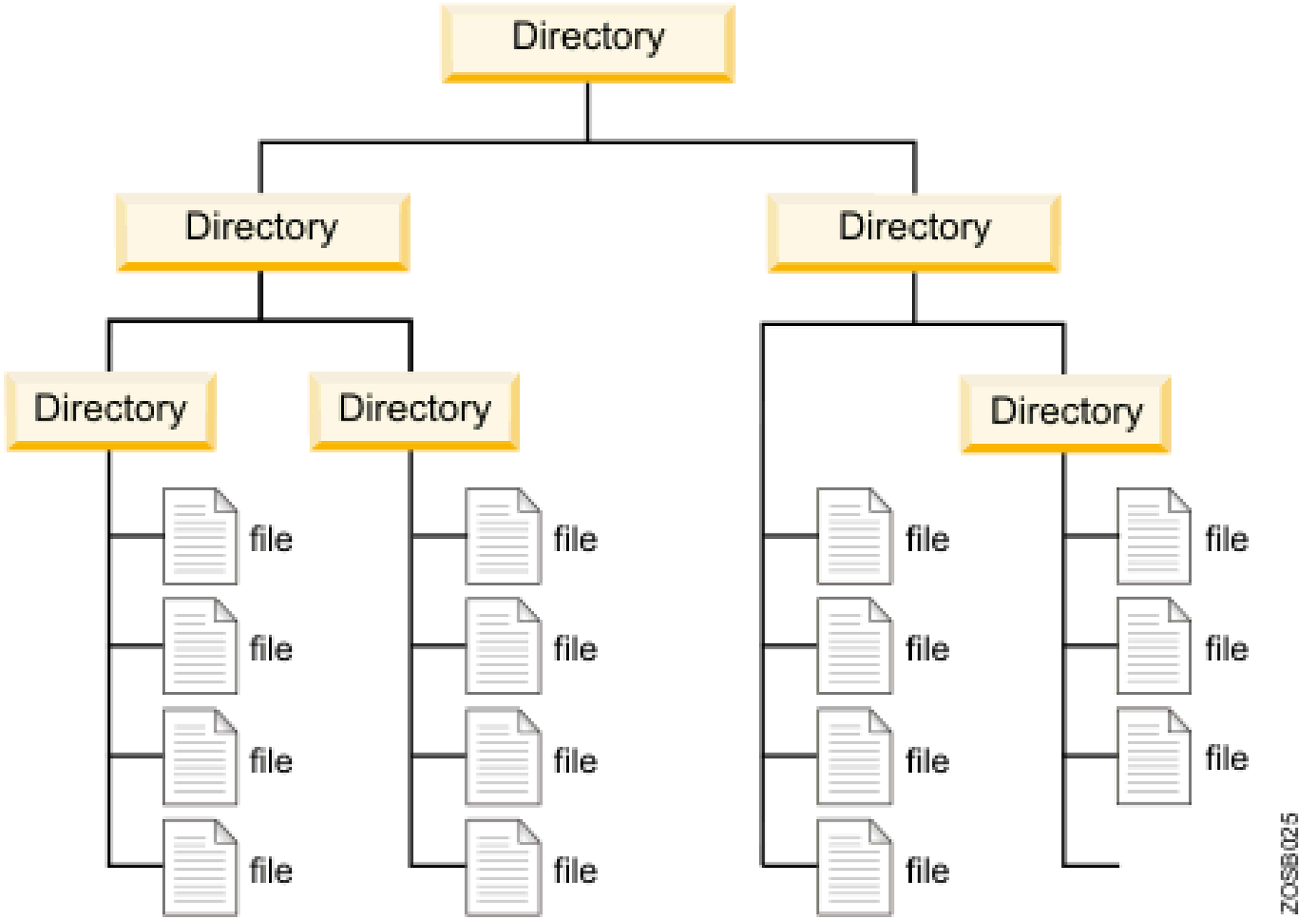
PAUSE FOLLOWING ••• FOLLOWER 1

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

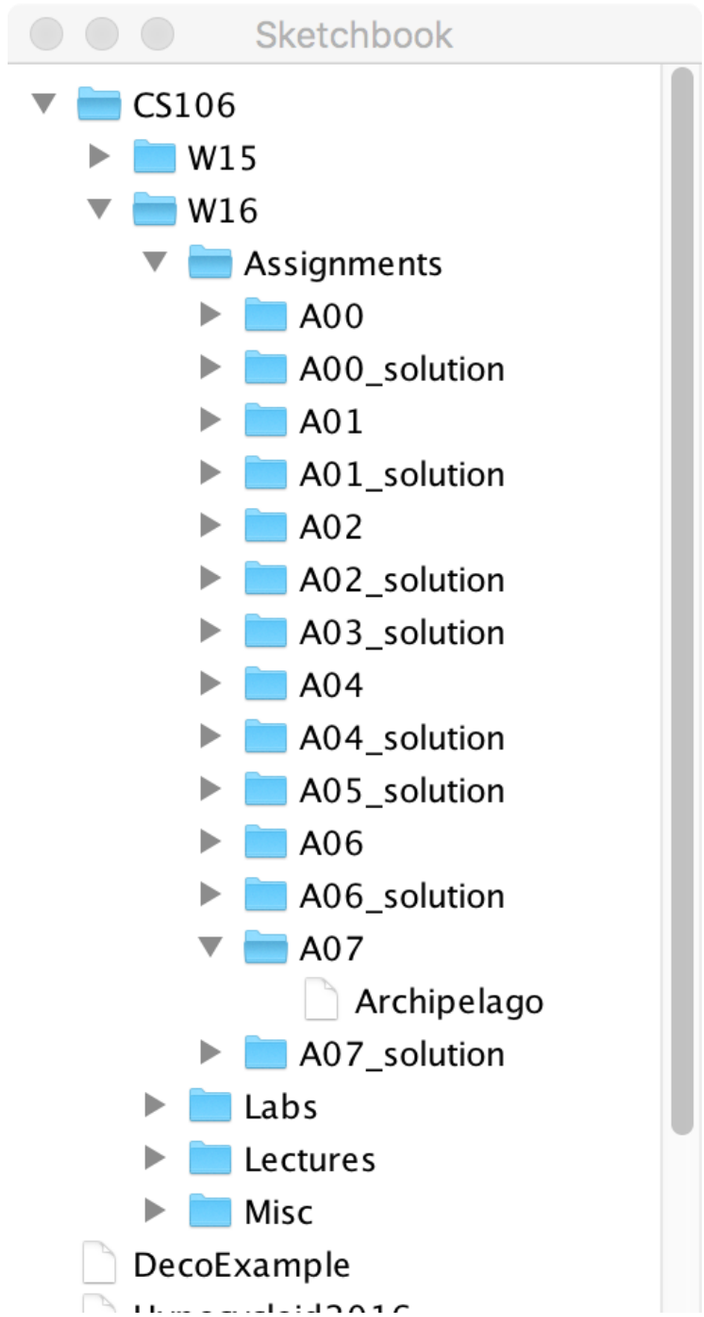

Sketchbook
CS106
W15
W16
Assignments
A00
A00_solution
A01
A01_solution
A02
A02_solution
A03_solution
A04
A04_solution
A05_solution
A06
A06_solution
A07
Archipelago
A07_solution
Labs
Lectures
Misc
DecoExample

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 +

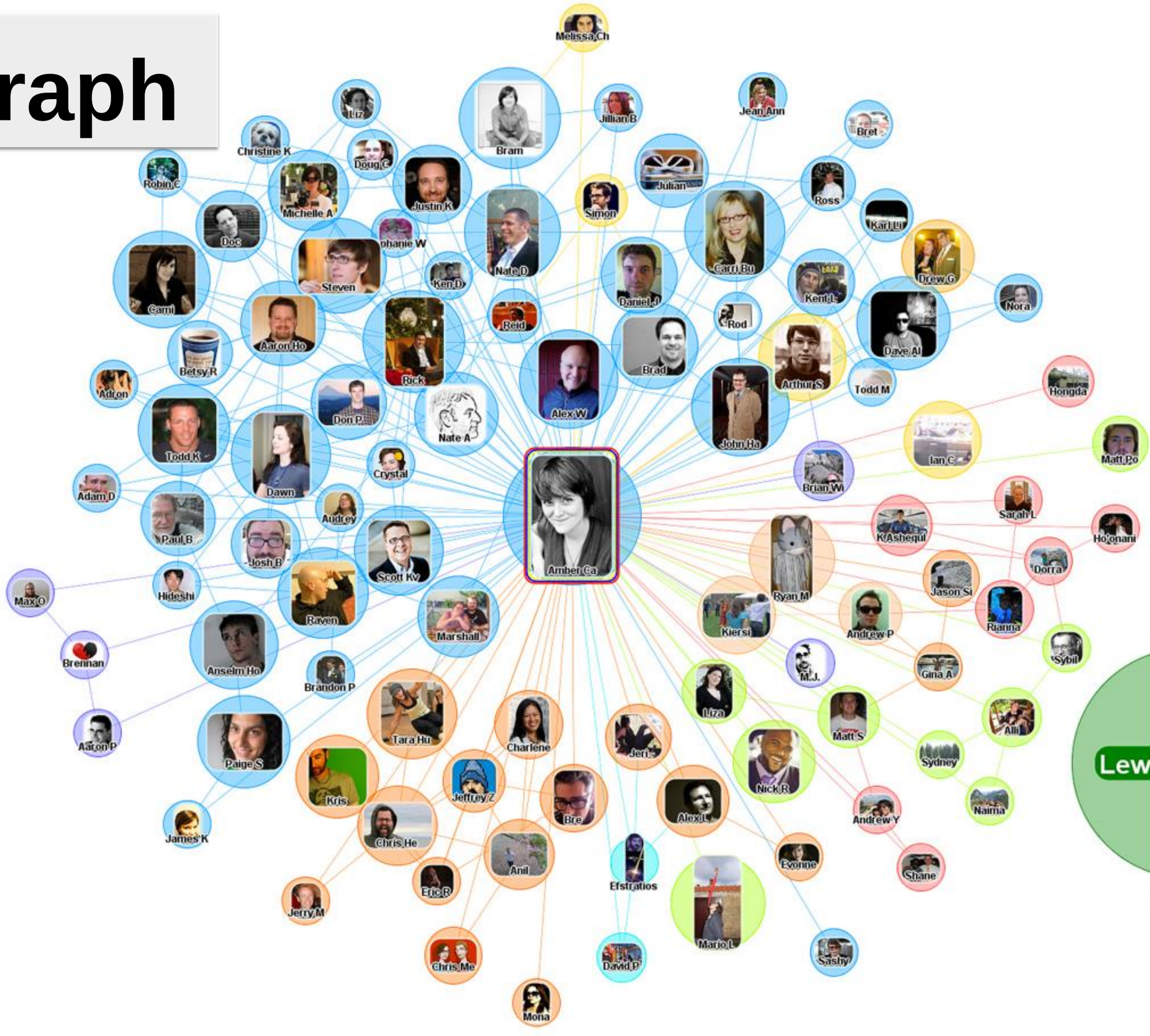
```
element.style {
}
body {                                          style.css:60
  margin: ▶0;
  padding: ▶0;
  overflow-y: scroll;
  background-color: #ddd;
  font-family: 'theSerif', 'Enriqueta',
    georgia, times, serif;
  -webkit-font-smoothing: antialiased;
  -webkit-text-size-adjust: none;
  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

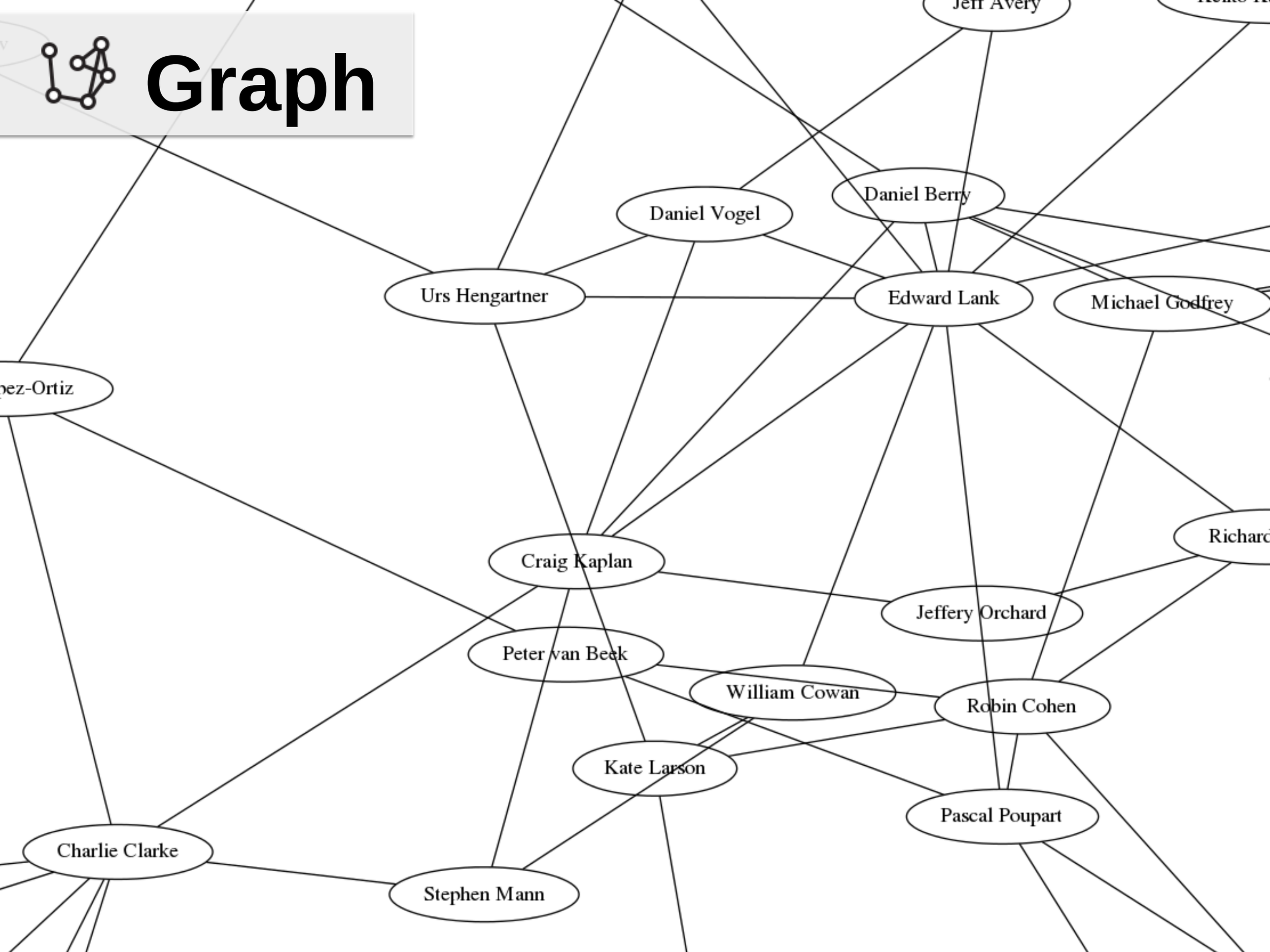
Filter Show all

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

Graph


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

Graph


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

Raw Text

- Web pages are raw text
  - index.html is a web page
  - index.html is raw text
- Let’s create an eReader on a web page
  - Read in text files
  - Display on index.html as raw text

HTML

- The tag defines a division or a section in an HTML document.

- The element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.

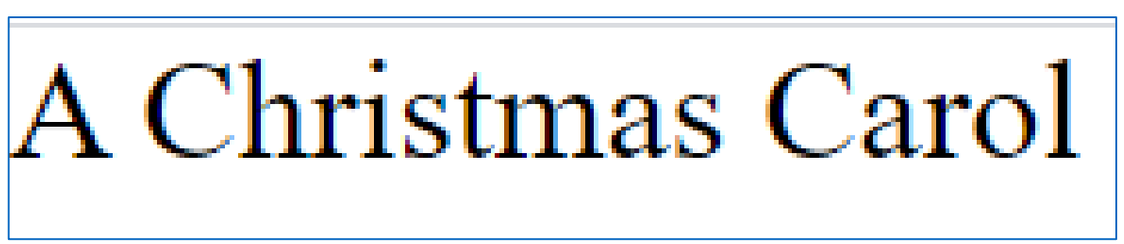
- createDiv() in JavaScript P5 creates a element in the DOM with given inner HTML.

HTML div
in JavaScript p5

```
let myDiv;
function setup() {
  createCanvas(100, 100);
  background(220);
  let myText = "A Christmas Carol";
  myDiv = createDiv(myText);
  myDiv.style("font-size", "48");
}
```

noCanvas()

```
let myDiv;
function setup() {
  noCanvas();
  let myText = "A Christmas Carol";
  myDiv = createDiv(myText);
  myDiv.style("font-size", "48");
}
```



Multiple Lines (1 of 2)

```
let myDiv;
let myText;
function setup() {
  noCanvas();
  myText = getText();
  myDiv = createDiv(myText);
  myDiv.style("font-size", "48");
}
```

Multiple Lines (2 of 3)

```
function getText() {
  let myText = "Books by Charles Dickens:" +
    "A Christmas Carol" +
    "Great Expectations" +
    "Oliver Twist" +
    "Hard Times";
    return myText;
}
```

Books by Charles Dickens:A Christmas CarolGreat ExpectationsOliver TwistHard Times

Multiple Lines (3 of 3)


```
function getText() {
  let myText = "Books by Charles Dickens:" +
    "<br>" + "A Christmas Carol" +
    "<br>" + "Great Expectations" +
    "<br>" + "Oliver Twist" +
    "<br>" + "Hard Times";
    return myText;
}
```

Books by Charles Dickens:
A Christmas Carol
Great Expectations
Oliver Twist
Hard Times

A primitive eReader

Displays all of “A Christmas Carol” on one page

```
let aChristmasCarol = [];
let myDiv;
let myText;

function preload() {
  aChristmasCarol =
loadStrings("/data/AChristmasCarol.txt");
}

function setup() {
  noCanvas();
  myText = join(aChristmasCarol, "<br>");
  myDiv = createDiv(myText);
  myDiv.style("font-size", "48");
}
```

Add Functionality to the eReader

- 30 lines per page
  - Will need page forward/back buttons
- Allow user to choose font size
- Allow user to select from multiple books by Dickens

30 Lines per Page (1 of 3)

```
let aChristmasCarol = [];
let myDiv;
let allLines;
let linesPerPage = 30;
let currentLine = 0;

function preload() {
  aChristmasCarol =
loadStrings("/data/AChristmasCarol.txt");
}
```

30 Lines per Page (2 of 3)

```
function setup() {
  noCanvas();

  pageForward = createButton("Forward");
  pageForward.mouseClicked(pageForwardFunc);
  pageForward.style('width', '100px');
  pageForward.style('height', '50px');

  myDiv = createDiv();
  myDiv.style("font-size", "48");
}
```

30 Lines per Page (3 of 3)

```
function draw() {
   let myText = "";
   for (let i = currentLine; i < currentLine +
      linesPerPage; i++) {
     myText = myText + aChristmasCarol[i]+"<br>";
   }
   myDiv.html(myText);
}

function pageForwardFunc() {
   currentLine = currentLine + linesPerPage;
}
```

Change Font Size Slider

- Add a global variable
```
let myFontSizeSlide;
```
- Add to setup()
```
createP("Font Size: ");
myFontSizeSlider = createSlider(20, 96, 48);
```
- Add to draw()
```
myDiv.style("font-size",
    myFontSizeSlider.value());
```

Add radio for Books (1 of 5)

- Add a global variable
  ```
  let bookSelectorRadio;
  ```

Add radio for Books (2 of 5) add to preload()

```
function preload() {
  aChristmasCarol =
loadStrings("/data/AChristmasCarol.txt");
  greatExpectations =
loadStrings("/data/GreatExpectations.txt");
  oliverTwist =
loadStrings("/data/OliverTwist.txt");
  hardTimes =
loadStrings("/data/HardTimes.txt");
}
```

Add radio for Books (3 of 5)
Add to setup()

```
bookSelectorRadio = createRadio();
 bookSelectorRadio.option("Carol");
 bookSelectorRadio.option("Expectations");
 bookSelectorRadio.option("Twist");
 bookSelectorRadio.option("Times");
 bookSelectorRadio.value("Carol");
 bookSelectorRadio.style("font-size", "48");
```

Add radio for Books (4 of 5)
Add to draw()

```
let currentBook = [];
 if (bookSelectorRadio.value() === "Carol") {
   currentBook = aChristmasCarol;
 } else if (bookSelectorRadio.value( ==="Expectations") {
   currentBook = greatExpectations;
} else if (bookSelectorRadio.value() === "Twist") {
   currentBook = oliverTwist;
} else if (bookSelectorRadio.value() === "Times") {
   currentBook = hardTimes;
}
```

Add radio for Books (5 of 5)
Add to draw()

```
if (previousBook != currentBook) {
  currentLine = 0;
  previousBook = currentBook;
}

for (let i=currentLine; i<currentLine+rowsPerPage; i++){
    tempText = tempText + currentBook[i] + "<br>";
}
  myDiv.html(tempText);
}
```

Fully implemented eReader is in demo code

“eReaderOnTheDOM”

What will this program display?

6 W20

What will this program display?

(

Review from Week 3 (I/O)

- Two examples for quick review
  - Shopping List
  - Speed Reader

Total KG

```
let lines = [];
let words = [];
let nextI;
function preload() {
  lines = loadStrings("data/shoppinglist.txt");
}
function setup() {
  createCanvas(600, 600);
  textSize(24);
}
function draw() {
  background(220);
  let totalKG = 0;
  for (let i = 0; i < lines.length; i++) {
    words = splitTokens(lines[i], " ");
    text(lines[i], 10, 30 + (i * 30));
    totalKG = totalKG + int(words[1]);
    nextI = i + 1;
    }
    text("Total KG: " + totalKG, 10, 30+(nextI * 30));
}
```

apple 2 kg
banana 5 kg
potato 4 kg
onion 2 kg

Total KG: 13

SpeedReader Example

- Read in a text file.
- Make one big long list (array) of “words”
  - Words may contain punctuation in this example
- Display one word at a time

SpeedReader

```
let lines = [];
let words = [];
let index = 0;
function preload() {
  lines = loadStrings("data/marley.txt");
}
function setup(){
  createCanvas(400, 200);
  textSize(50);
  textAlign(CENTER);
  fill(255);
  let allLines = join( lines, " ");
  words = splitTokens(allLines);
  frameRate(1);
}
function draw() {
  background( 80 );
  text(words[index], width/2, height/2);
  index = (index + 1) % words.length;
}
```

Messier text

```
function splitTokens(text, delims) { … }
```

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

```
function trim(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

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

Messy Text (1 of 2)

```
let lines;
let myDiv;

function preload() {
  lines = loadStrings( "/data/ReservedStreetnames.csv" );
}
```

Messy Text (2 of 2)

```
function setup() {
  noCanvas();
  myDiv = createDiv();
  myDiv.style("font-size", 36);
  let list = "";
  for ( let idx = 0; idx < lines.length; ++idx ) {
    let line = lines[idx];
    if ( line.charAt( 0 ) != '-' ) {
      let words = splitTokens( line, "|" );
      let street = trim( words[0] );
      let municipality = trim( words[1] );
      list = list + street + "---" + municipality +"<br>";
    }
  }
  myDiv.html(list);
}
```

Fully implemented reserved street names in demo code

“ReservedStreetNamesP5”

Time permitting:

There is one more example of using JavaScript p5 and HTML in the demo code. It builds an unordered list of Canadian Prime Ministers.

“UnorderedList”

Reading the dictionary to solve Dictionary “problems”

A
a
aa
aal
aalii
aam
Aani
aardvark
aardwolf
Aaron
Aaronic
Aaronical
Aaronite
Aaronitic
Aaru
Ab
aba
Ababdeh
Ababua
abac

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 all Palindromes (1 of 2)

```
let words = [];
let palindromes = [];
let myDiv;

function preload() {
  words = loadStrings("/data/H_words.txt");
}
```

Find all Palindromes (2 of 2)

```
function setup() {

  myDiv = createDiv();
  myDiv.style("font-size", "48");

  for (let i = 0; i < words.length; i++) {

    let pal = true;
    let wordLength = words[i].length;
    let halfWordLength = int(words[i].length / 2);

    for (let j = 0; j < halfWordLength; j++) {
      if (words[i][j] != words[i][wordLength - j - 1]) {
        pal = false;
      }
    }
    if (pal === true) {
      palindromes.push(words[i]);
    }
  }
  myDiv.html(join(palindromes, "<br>"));
}
```

Fully implemented palindrome is in the demo code

“Palindromes”

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. Here are 5 examples.

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

Writing an inefficient spellchecker (1 of 2)

```
let words;
let myDiv;
let carolLines;

function preload() {
  words = loadStrings( "/data/H_words.txt" );
  carolLines = loadStrings( "/data/AChristmascarol.txt" );
}
```

Writing an inefficient spellchecker (2 of 2)

```
function setup()
{
  myDiv = createDiv();
  myDiv.style("font-size", 48);

  let text = join( carolLines, ' ' );
  let carolWords = splitTokens( text, " ,.!0123456789[]*:!" );

  let badWords = [];
  for ( let idx = 0; idx < carolWords.length; ++idx ) {
    let  word = carolWords[idx].toLowerCase();
    let wordFound = isWord(word);
    if (!wordFound) {
      badWords.push(word);
    }
  }
  myDiv.html(join(badWords, "<br>"));
}

function isWord(wd) {
  for (let j = 0; j < words.length; j++) {
    if (wd === words[j]) {
      return true;
    }
  }
  return false;
}
```

Fully implemented inefficient spell checker is in the demo code

"ArraySpellCheckerP5"

Writing an efficient spellchecker (1 of 2)

```
let dictionary = {};
let words;
let myDiv;
let carolLines;

function preload() {
  words = loadStrings( "/data/H words.txt" );
  carolLines = loadStrings(
"/data/AChristmascarol.txt" );
}
```

Writing an efficient spellchecker (2 of 2)

```
function setup()
{
  myDiv = createDiv();
  myDiv.style("font-size", 48);
  for ( let idx = 0; idx < words.length; ++idx ) {
    dictionary[words[idx]] = 1;
  }

  let text = join( carolLines, ' ' );
  let carolWords=splitTokens(text, " ,.!0123456789[]*:!");

  let badWords = [];
  for ( let idx = 0; idx < carolWords.length; ++idx ) {
    let word = carolWords[idx].toLowerCase();
    if ( !dictionary[word] ) {
      badWords.push(word);
    }
  }
  myDiv.html(join(badWords, "<br>"));
}
```

Fully implemented efficient spell checker is in the demo code

"DictSpellCheckerP5"

Regular Expressions: match()

To help us search through data, p5 provides us with the “match” function.

```
function match( text,  pattern ) { ... }
```

Reference

Search the API

match()

Example

p5js*

```
var string = 'Hello p5js*!';
var regexp = 'p5js\\*';
var m = match(string, regexp);
text(m, 5, 50);
```

edit reset copy

Description

This function is used to apply a regular expression to a piece of text, and return matching groups (elements found inside parentheses) as a String array. If there are no matches, a null value will be returned. If no groups are specified in the regular expression, but the sequence matches, an array

Reference

Search the API

match()

Example

p5js*

```
var string = 'Hello p5js*!';
var regexp = 'p5js\\*';
var m = match(string, regexp);
text(m, 5, 50);
```

edit reset copy

Description

This function is used to apply a regular expression to a piece of text, and return matching groups (elements found inside parentheses) as a String array. If there are no matches, a null value will be returned. If no groups are specified in the regular expression, but the sequence matches, an array

```
function match( text,  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.

Finding patterns

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

```
if (match(myString, "(519) 888-4567") != null) { … }
```

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

Finding patterns

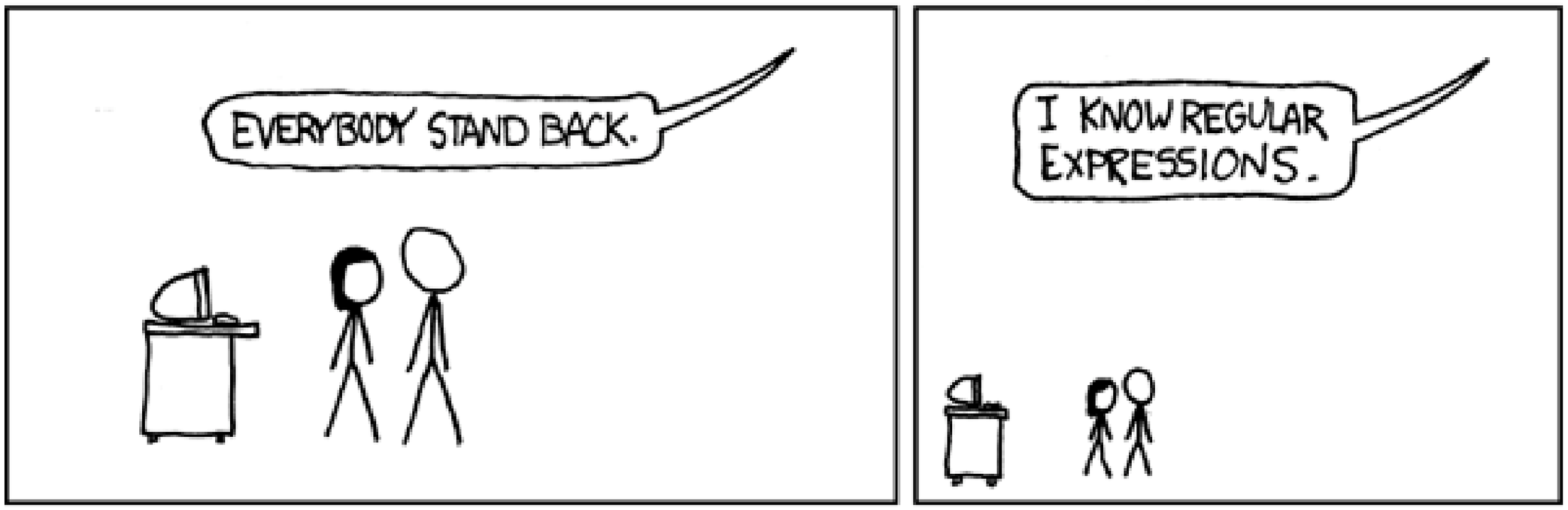
Regular Expressions are a general tool for finding patterns in strings.

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/

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:

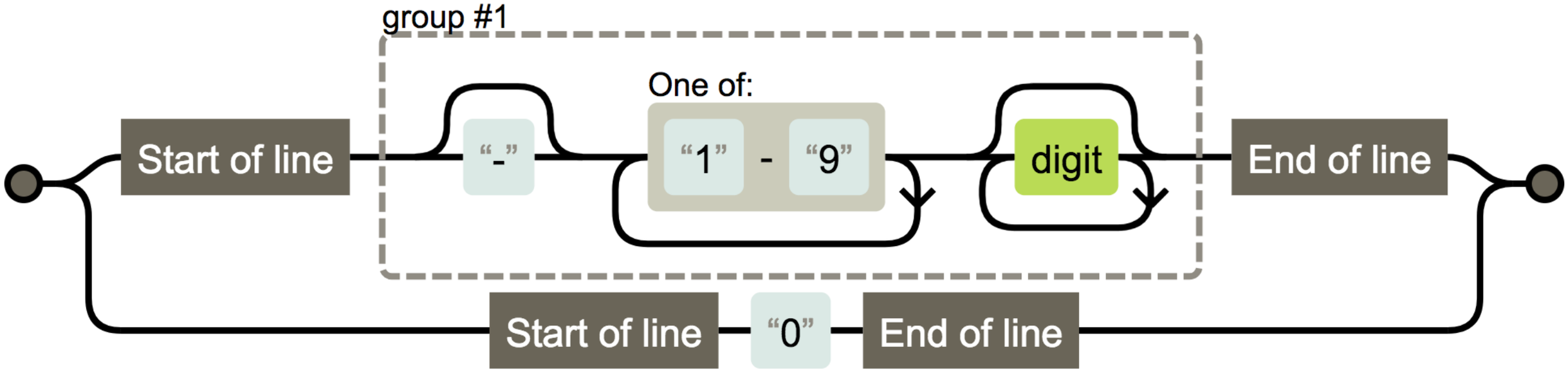
\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b

A URL:

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

A regular expression is like a little “machine”:

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



Phone Number Patterns (1 of 2)

```
let p = "(?:(?:\\((\\d\\d\\d)\\) )|(\\d\\d\\d)[ -
])(\\d\\d\\d)-(\\d\\d\\d\\d)";
let lines = [];
let myDiv;
let phoneNumbers = [];

function preload() {
  print("preload");
  lines = loadStrings( "/data/input.txt" );
}
```

Phone Number Patterns (2 of 2)

```
function setup()
{
  noCanvas();
  myDiv = createDiv();
  myDiv.style("font-size", "48");

  for ( let idx = 0; idx < lines.length; ++idx ) {
    let m = match( lines[idx], p );
    if ( m != null ) {
      if ( m[1] == null ) {
        phoneNumbers = phoneNumbers + "<br>" + "(" + m[2] + ") " +
m[3] + "-" + m[4];
      } else {
        phoneNumbers = phoneNumbers + "<br>" + "(" + m[1] + ") " +
m[3] + "-" + m[4];
      }
    }
  }
  myDiv.html(phoneNumbers);
}
```

Fully implement of finding phone number patterns is in the demo code

“FindPhoneNumbersP5”