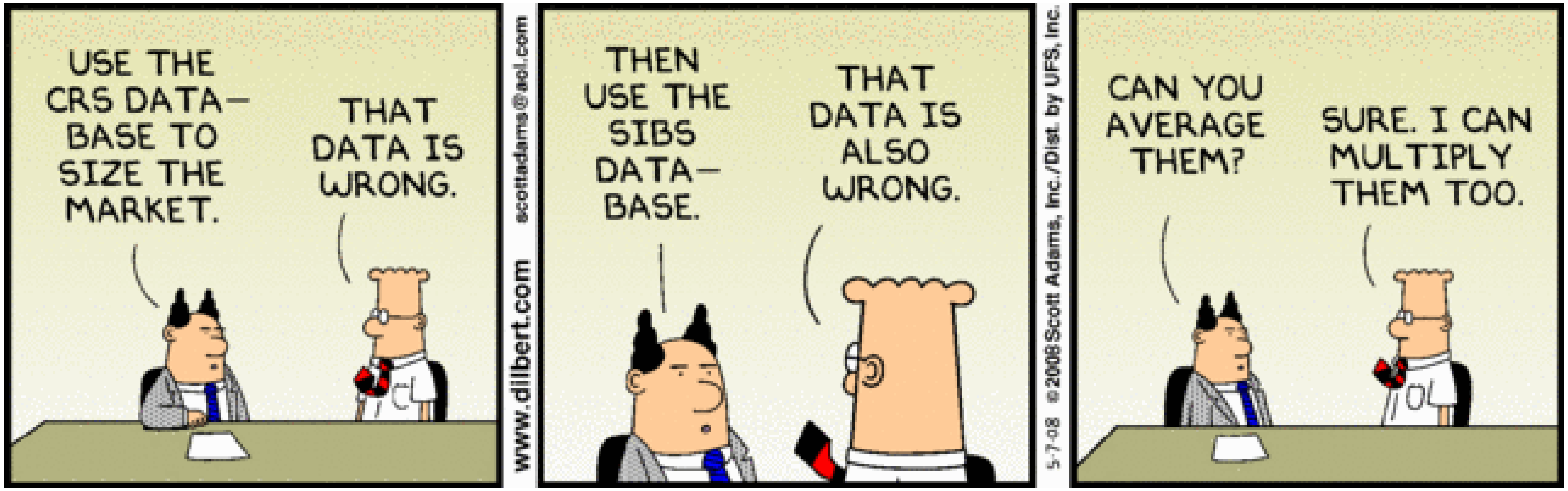
CS109 Data Science

Data Munging

Hanspeter Pfister
pfister@seas.harvard.edu

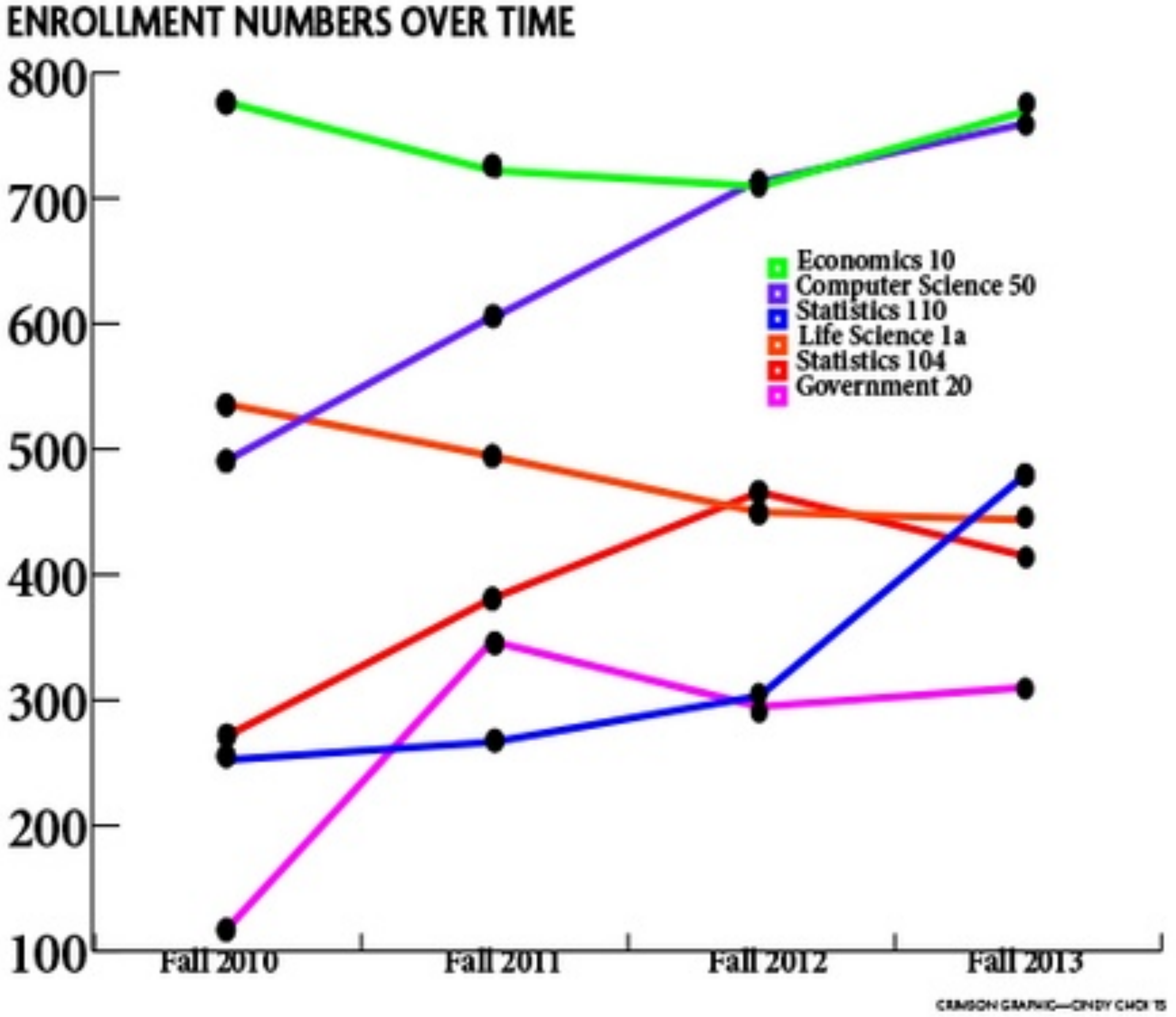

USE THE CRS DATA-BASE TO SIZE THE MARKET.
THAT DATA IS WRONG.
scottadams@aol.com
www.dilbert.com
THEN USE THE SIBS DATA-BASE.
THAT DATA IS ALSO WRONG.
5-7-08 ©2008 Scott Adams, Inc./Dist. by UFS, Inc.
CAN YOU AVERAGE THEM?
SURE. I CAN MULTIPLY THEM TOO.

http://dilbert.com/strips/comic/2008-05-07/

Enrollment Numbers

377 including all 4 course numbers

172 in CS109, 84 in Stat121, 61 in AC209, 60 in E-109

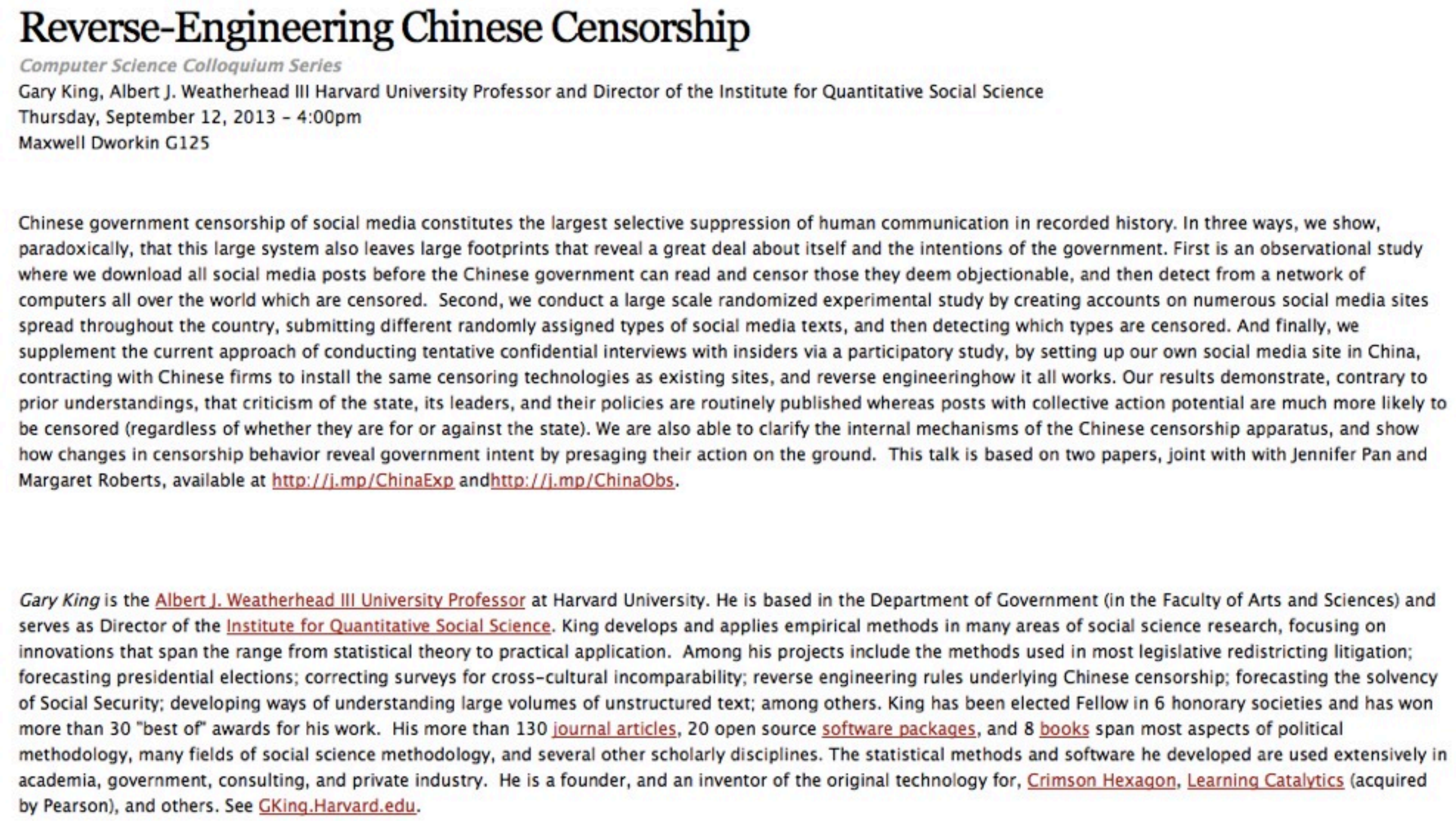


Crimson, Sept 12, 2013

This Week

- HW1 - due Thursday, Sept 19 - you really need to start now!
- Friday lab **10-11:30 am** in MD G115
    - *Data Scraping with Python* with Ray and Johanna
- New classroom, starting Tuesday!
    - Science Center Hall C (Tu / Th)
    - Labs will continue to be in MD G115 (F)

CS Colloquium, Today, 4-5:30 pm, MD G125



Reverse-Engineering Chinese Censorship

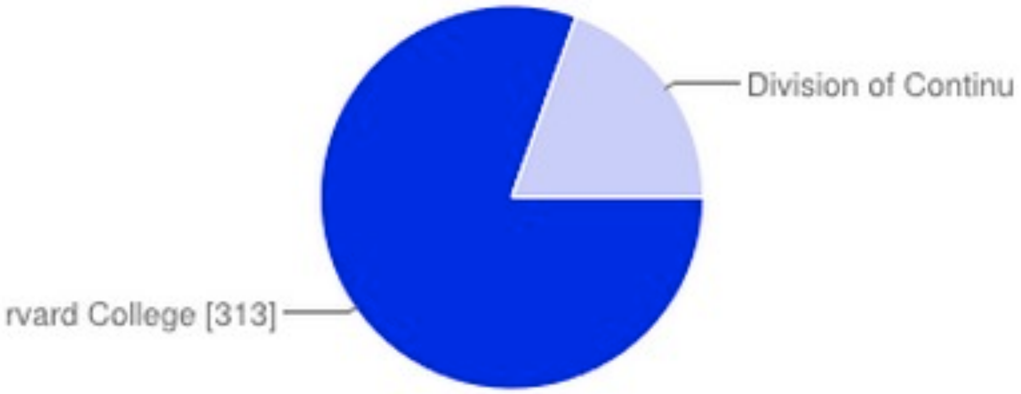
Computer Science Colloquium Series

Gary King, Albert J. Weatherhead III Harvard University Professor and Director of the Institute for Quantitative Social Science
Thursday, September 12, 2013 - 4:00pm
Maxwell Dworkin G125

Chinese government censorship of social media constitutes the largest selective suppression of human communication in recorded history. In three ways, we show, paradoxically, that this large system also leaves large footprints that reveal a great deal about itself and the intentions of the government. First is an observational study where we download all social media posts before the Chinese government can read and censor those they deem objectionable, and then detect from a network of computers all over the world which are censored. Second, we conduct a large scale randomized experimental study by creating accounts on numerous social media sites spread throughout the country, submitting different randomly assigned types of social media texts, and then detecting which types are censored. And finally, we supplement the current approach of conducting tentative confidential interviews with insiders via a participatory study, by setting up our own social media site in China, contracting with Chinese firms to install the same censoring technologies as existing sites, and reverse engineeringhow it all works. Our results demonstrate, contrary to prior understandings, that criticism of the state, its leaders, and their policies are routinely published whereas posts with collective action potential are much more likely to be censored (regardless of whether they are for or against the state). We are also able to clarify the internal mechanisms of the Chinese censorship apparatus, and show how changes in censorship behavior reveal government intent by presaging their action on the ground. This talk is based on two papers, joint with with Jennifer Pan and Margaret Roberts, available at http://j.mp/ChinaExp andhttp://j.mp/ChinaObs.

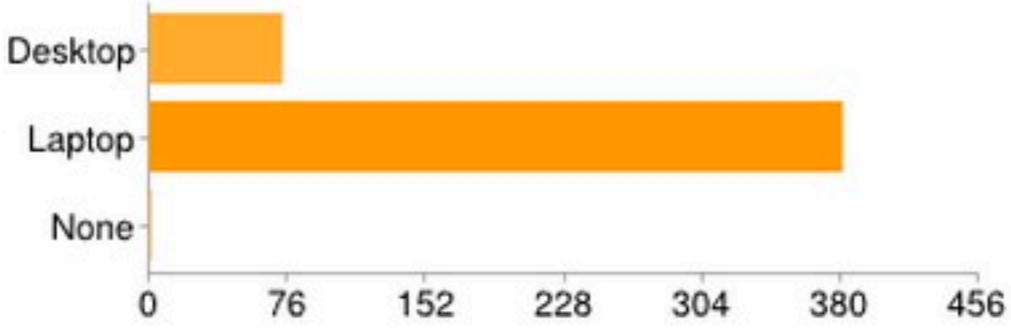
Gary King is the Albert J. Weatherhead III University Professor at Harvard University. He is based in the Department of Government (in the Faculty of Arts and Sciences) and serves as Director of the Institute for Quantitative Social Science. King develops and applies empirical methods in many areas of social science research, focusing on innovations that span the range from statistical theory to practical application. Among his projects include the methods used in most legislative redistricting litigation; forecasting presidential elections; correcting surveys for cross-cultural incomparability; reverse engineering rules underlying Chinese censorship; forecasting the solvency of Social Security; developing ways of understanding large volumes of unstructured text; among others. King has been elected Fellow in 6 honorary societies and has won more than 30 "best of" awards for his work. His more than 130 journal articles, 20 open source software packages, and 8 books span most aspects of political methodology, many fields of social science methodology, and several other scholarly disciplines. The statistical methods and software he developed are used extensively in academia, government, consulting, and private industry. He is a founder, and an inventor of the original technology for, Crimson Hexagon, Learning Catalytics (acquired by Pearson), and others. See GKing.Harvard.edu.

ID Type



Harvard College	**313**	80%
Division of Continuing Education (DCE)	**76**	20%

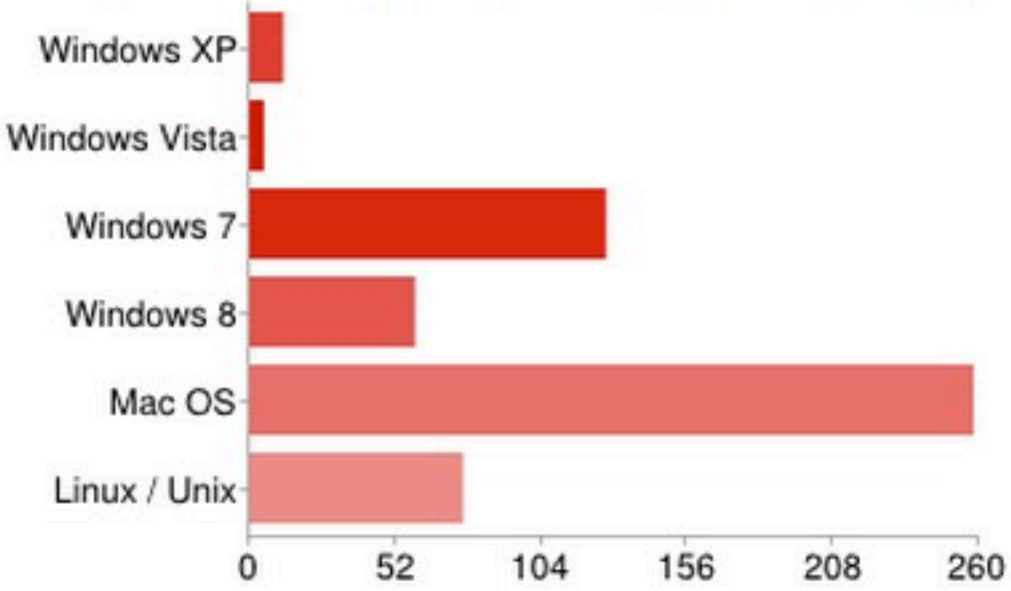
What kind(s) of computer(s) do you own?



Desktop	**73**	19%
Laptop	**381**	98%
None	**1**	0%

People may select more than one checkbox, so percentages may add up to more than 100%.

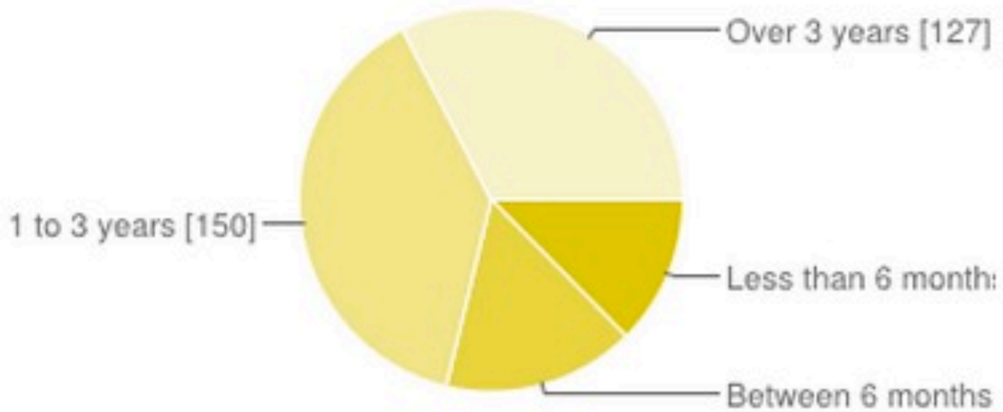
What operating system(s) do you run on your computer(s)?



Windows XP	**12**	3%
Windows Vista	**5**	1%
Windows 7	**127**	33%
Windows 8	**59**	15%
Mac OS	**258**	66%
Linux / Unix	**76**	20%

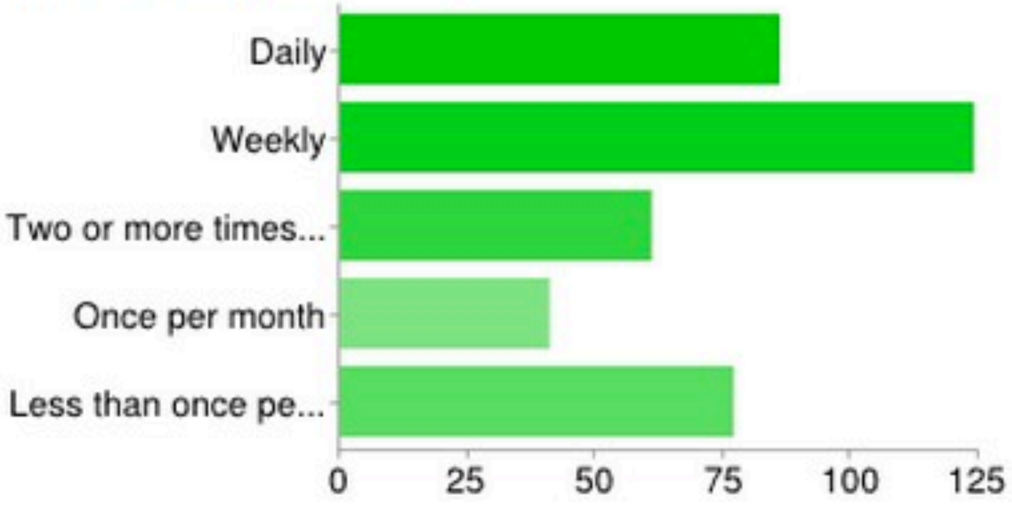
People may select more than one checkbox, so percentages may add up to more than 100%.

How long have you been programming?



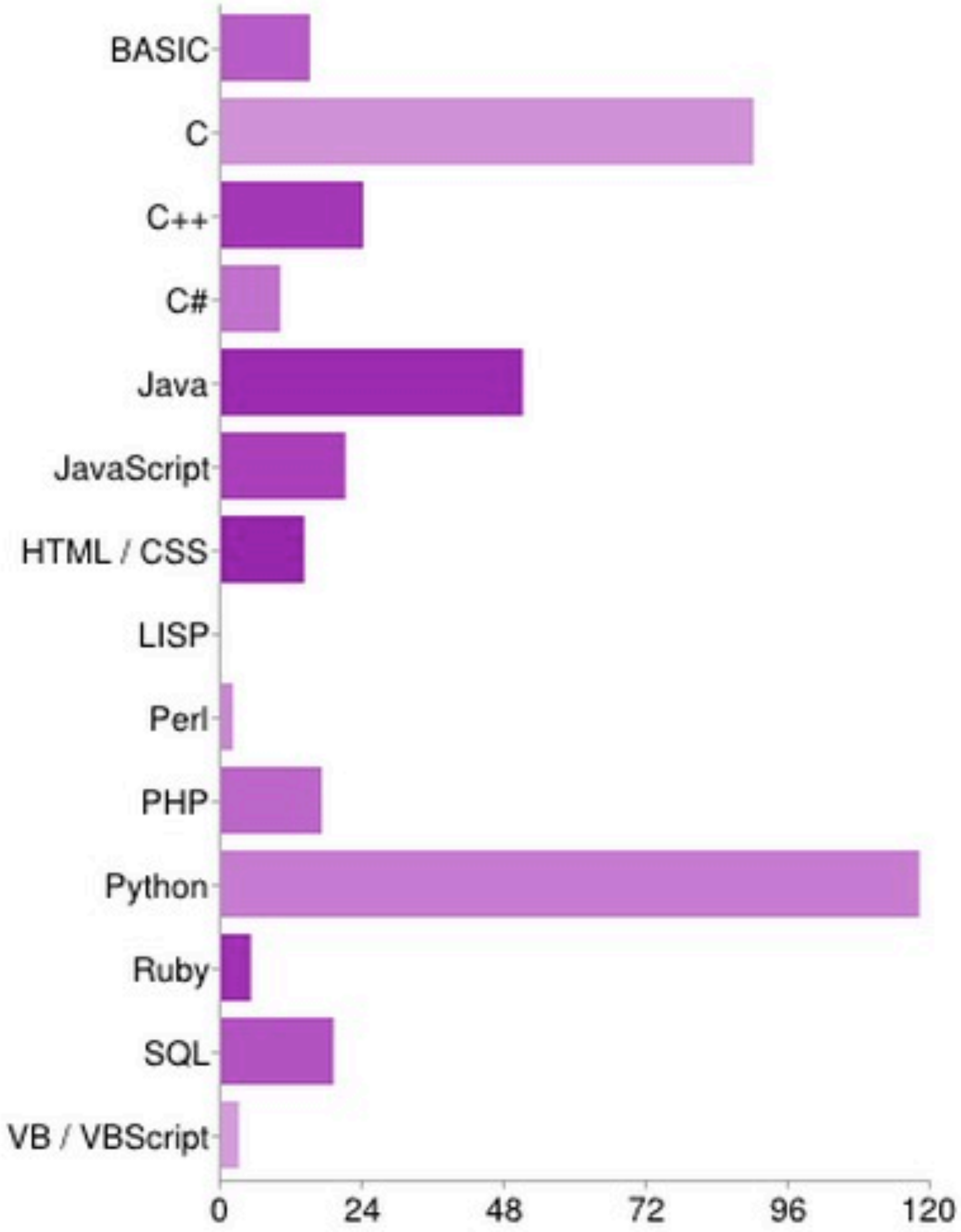
Less than 6 months	**49**	13%
Between 6 months and one year	**63**	16%
1 to 3 years	**150**	39%
Over 3 years	**127**	33%

How often do you write code?



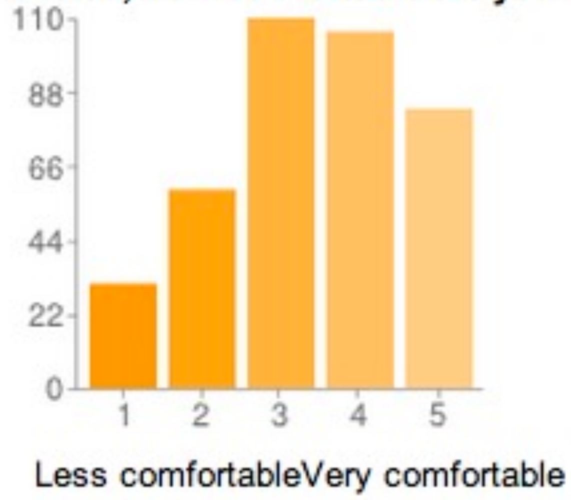
Daily	**86**	22%
Weekly	**124**	32%
Two or more times per month	**61**	16%
Once per month	**41**	11%
Less than once per month	**77**	20%

What is your primary programming language?



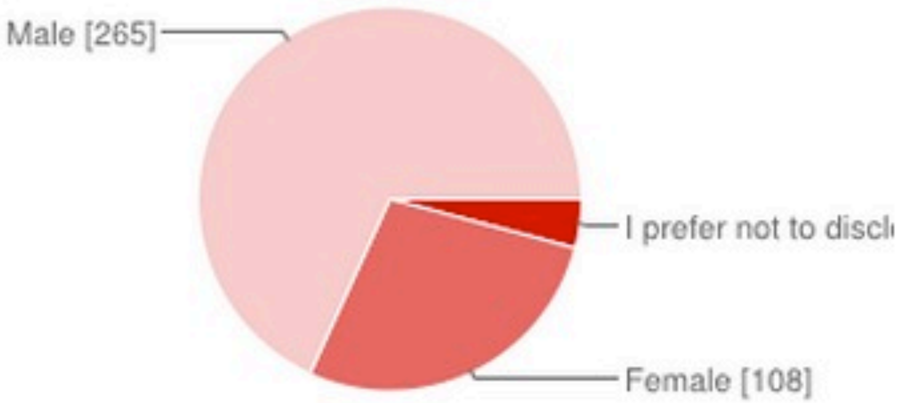
BASIC	**15**	4%
C	**90**	23%
C++	**24**	6%
C#	**10**	3%
Java	**51**	13%
JavaScript	**21**	5%
HTML / CSS	**14**	4%
LISP	**0**	0%
Perl	**2**	1%
PHP	**17**	4%
Python	**118**	30%
Ruby	**5**	1%
SQL	**19**	5%
VB / VBScript	**3**	1%

Overall, how comfortable are you with programming?



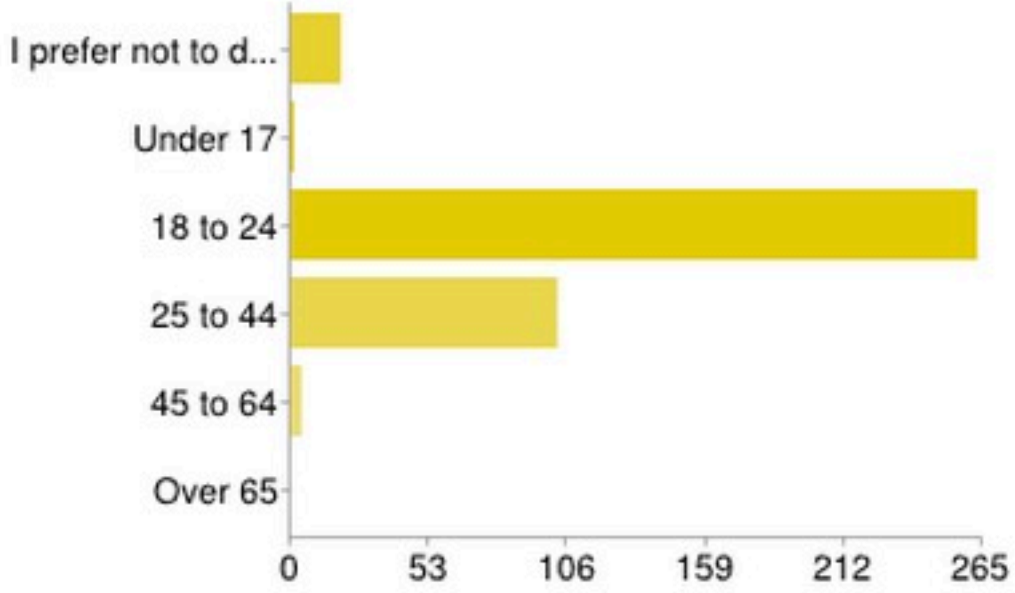
1 - Less comfortable	**31**	8%
2	**59**	15%
3	**110**	28%
4	**106**	27%
5 - Very comfortable	**83**	21%

Gender



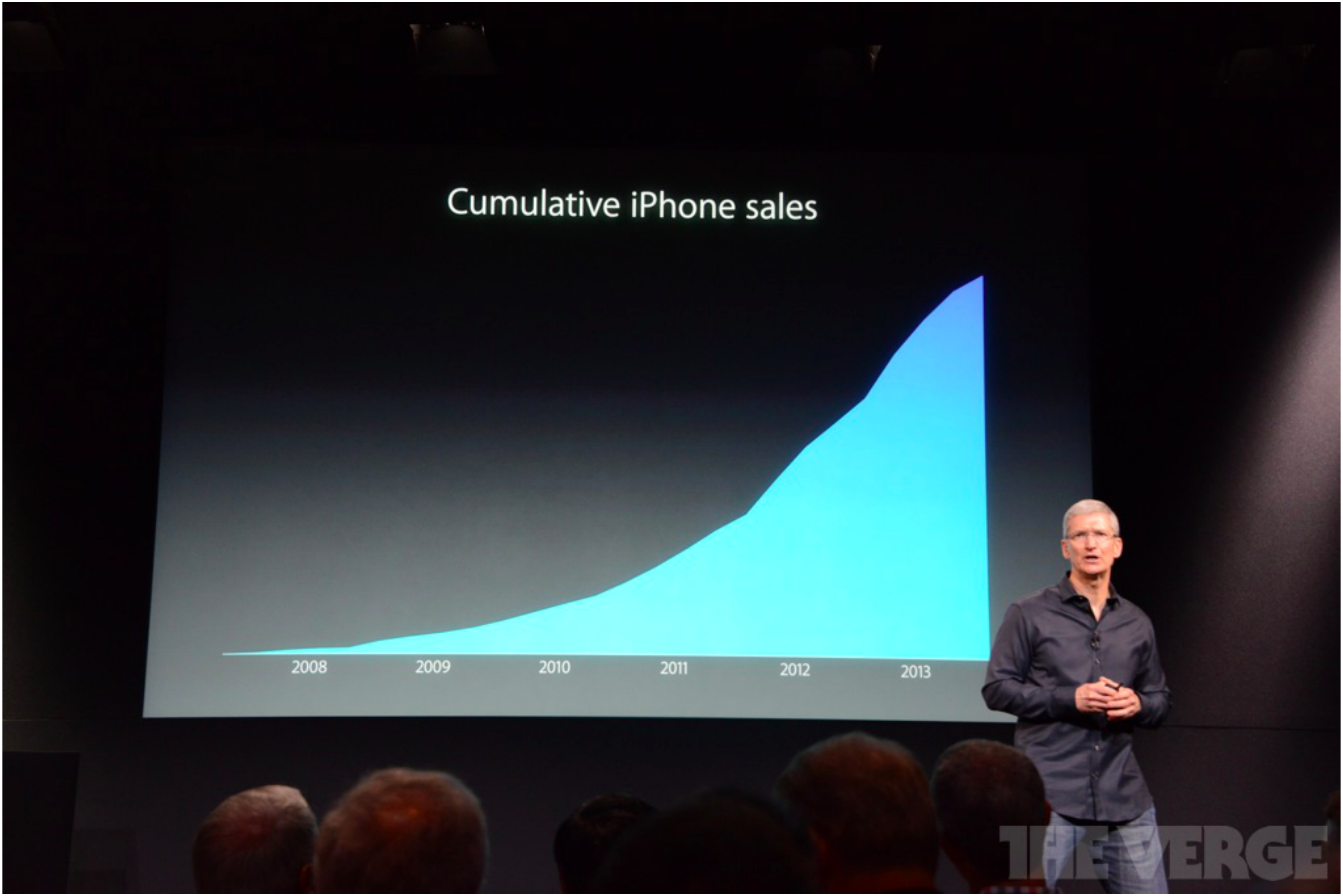
I prefer not to disclose	**16**	4%
Female	**108**	28%
Male	**265**	68%

Age

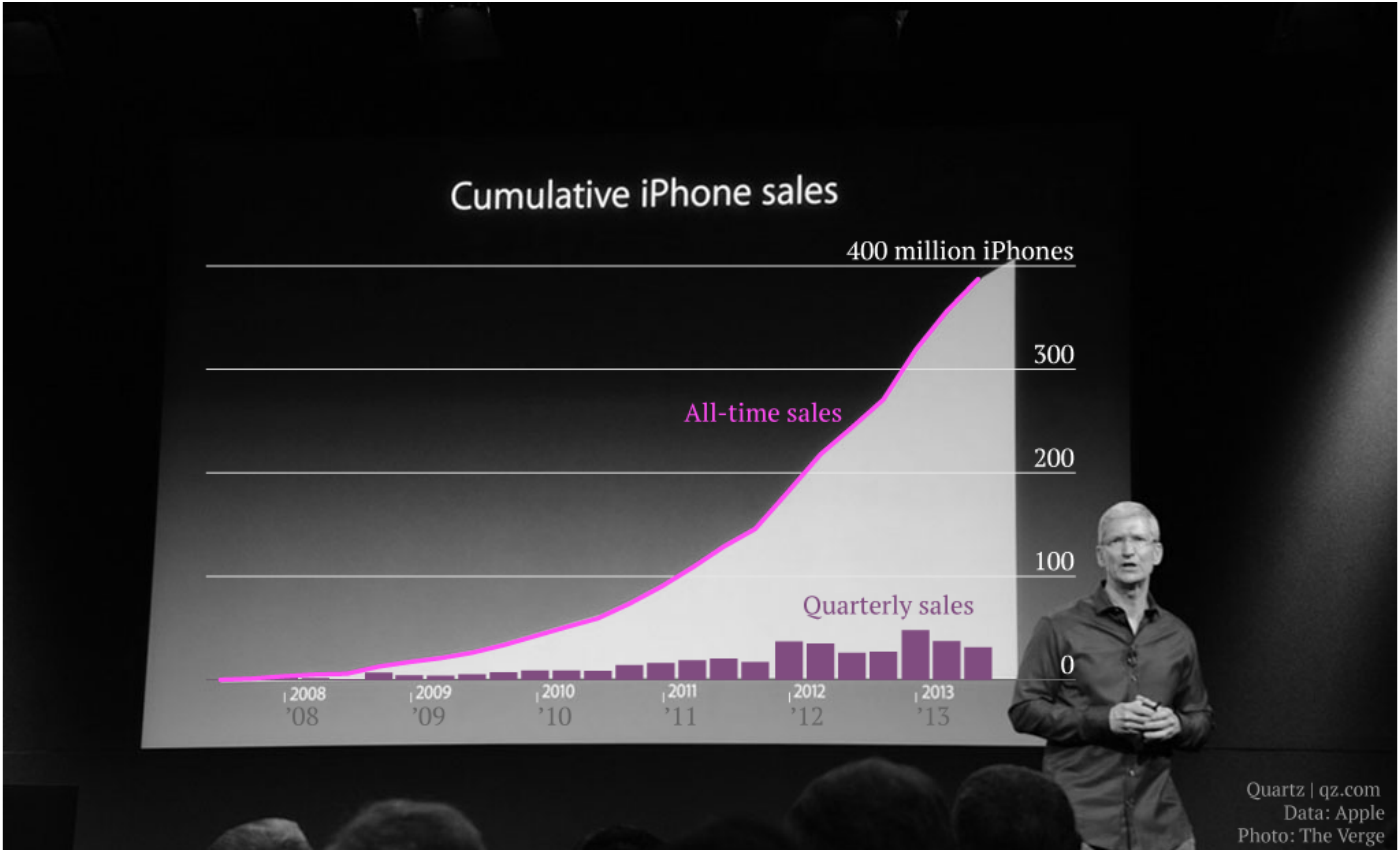


I prefer not to disclose	**19**	5%
Under 17	**1**	0%
18 to 24	**263**	68%
25 to 44	**102**	26%
45 to 64	**4**	1%
Over 65	**0**	0%

Vis of the Week



David Yanofsky, Quartz



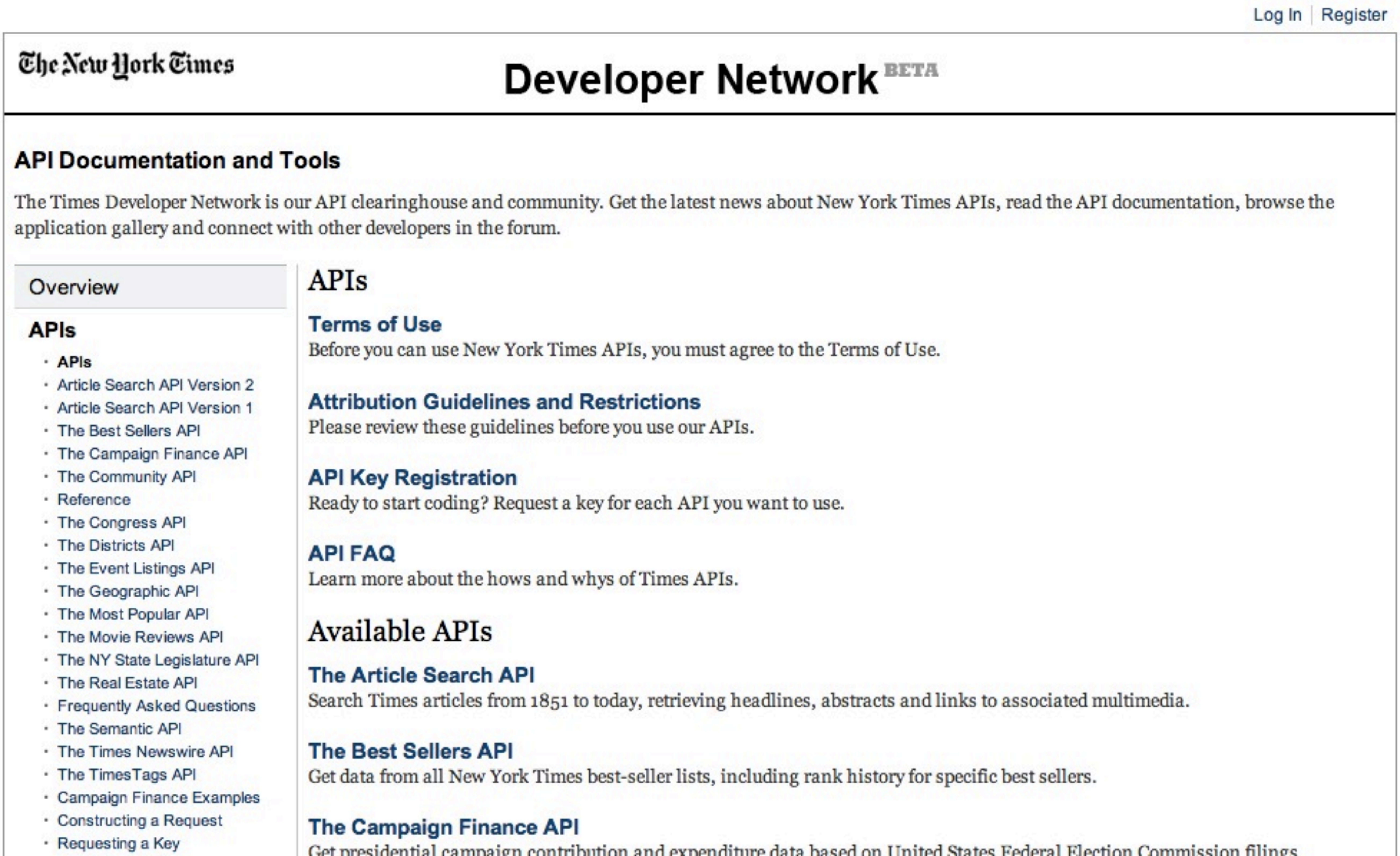
David Yanofsky, Quartz

Getting Data from the WWW

Data Access Schemes

- Bulk downloads
  - Wikipedia, IMDB, Million Song Database, etc.
  - See list of data web sites on the Resources page
- API access
  - NY Times, Twitter, Facebook, Foursquare, Google, ...
- Web scraping

NY Times API



The New York Times

Developer Network BETA

API Documentation and Tools

The Times Developer Network is our API clearinghouse and community. Get the latest news about New York Times APIs, read the API documentation, browse the application gallery and connect with other developers in the forum.

APIs

- **APIs**
- Article Search API Version 2
- Article Search API Version 1
- The Best Sellers API
- The Campaign Finance API
- The Community API
- Reference
- The Congress API
- The Districts API
- The Event Listings API
- The Geographic API
- The Most Popular API
- The Movie Reviews API
- The NY State Legislature API
- The Real Estate API
- Frequently Asked Questions
- The Semantic API
- The Times Newswire API
- The TimesTags API
- Campaign Finance Examples
- Constructing a Request
- Requesting a Key

APIs

Terms of Use

Before you can use New York Times APIs, you must agree to the Terms of Use.

Attribution Guidelines and Restrictions

Please review these guidelines before you use our APIs.

API Key Registration

Ready to start coding? Request a key for each API you want to use.

API FAQ

Learn more about the hows and whys of Times APIs.

Available APIs

The Article Search API

Search Times articles from 1851 to today, retrieving headlines, abstracts and links to associated multimedia.

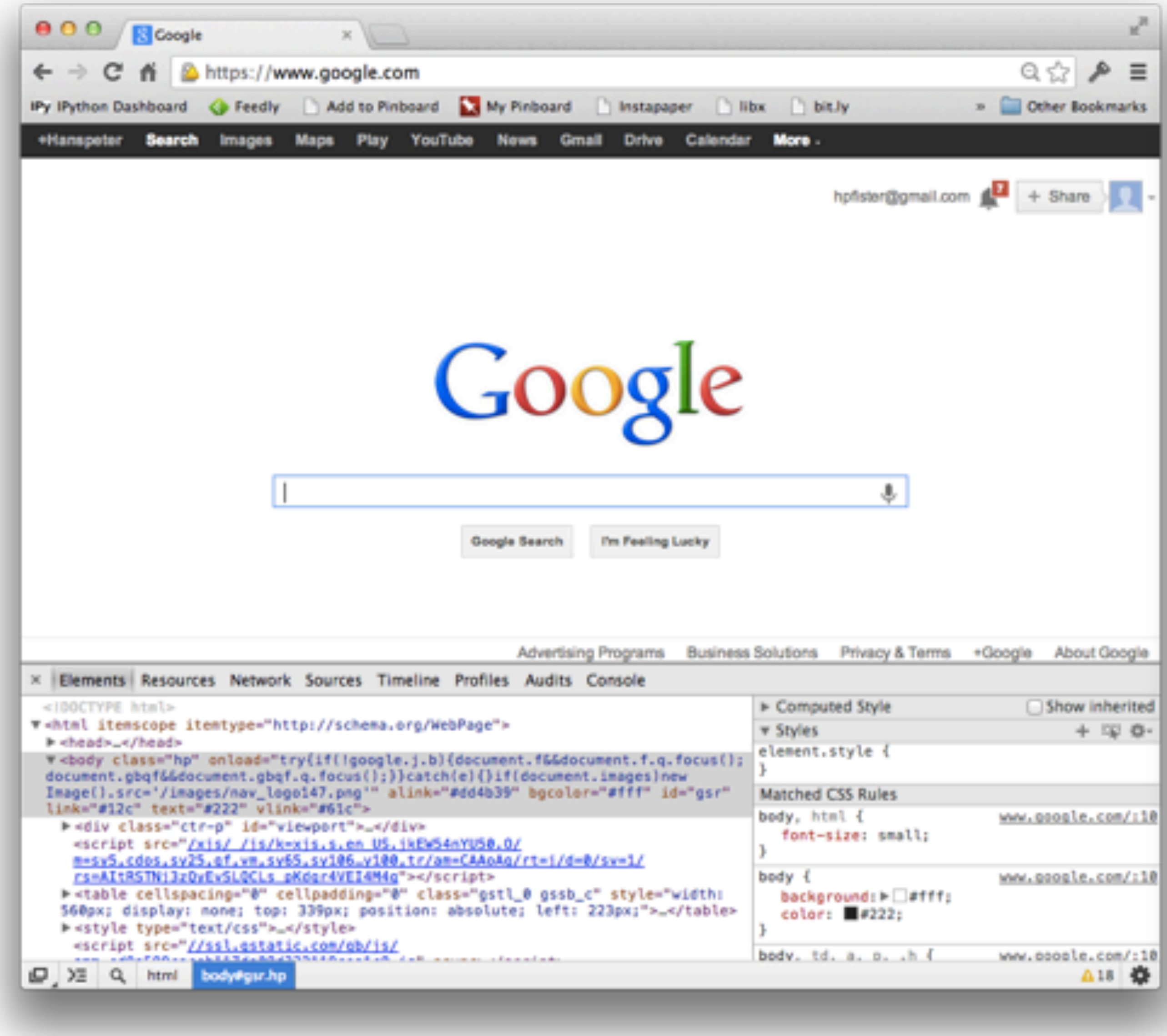
The Best Sellers API

Get data from all New York Times best-seller lists, including rank history for specific best sellers.

The Campaign Finance API

Get presidential campaign contribution and expenditure data based on United States Federal Election Commission filings

Developer Tools



```
<!DOCTYPE html>
▼<html itemscope itemtype="http://schema.org/WebPage">
  ▶<head>…</head>
  ▼<body class="hp" onload="try{if(!google.j.b){document.f&&document.f.q.focus();
  document.gbqf&&document.gbqf.q.focus();}}catch(e){}if(document.images)new
  Image().src='/images/nav_logo147.png'" alink="#dd4b39" bgcolor="#fff" id="gsr"
  link="#12c" text="#222" vlink="#61c">
    ▶<div class="ctr-p" id="viewport">…</div>
     <script src="/xjs/_/js/k=xjs.s.en_US.jkEW54nYU50.O/
     m=sy5,cdos,sy25,gf,vm,sy65,sy106,v100,tr/am=CAAoAg/rt=j/d=0/sv=1/
     rs=AItRSTNj3zQyEvSLQCLs_pKdgr4VEI4M4g"></script>
    ▶<table cellspacing="0" cellpadding="0" class="gstl_0 gssb_c" style="width:
    560px; display: none; top: 339px; position: absolute; left: 223px;">…</table>
    ▶<style type="text/css">…</style>
     <script src="//ssl.gstatic.com/gb/js/
```

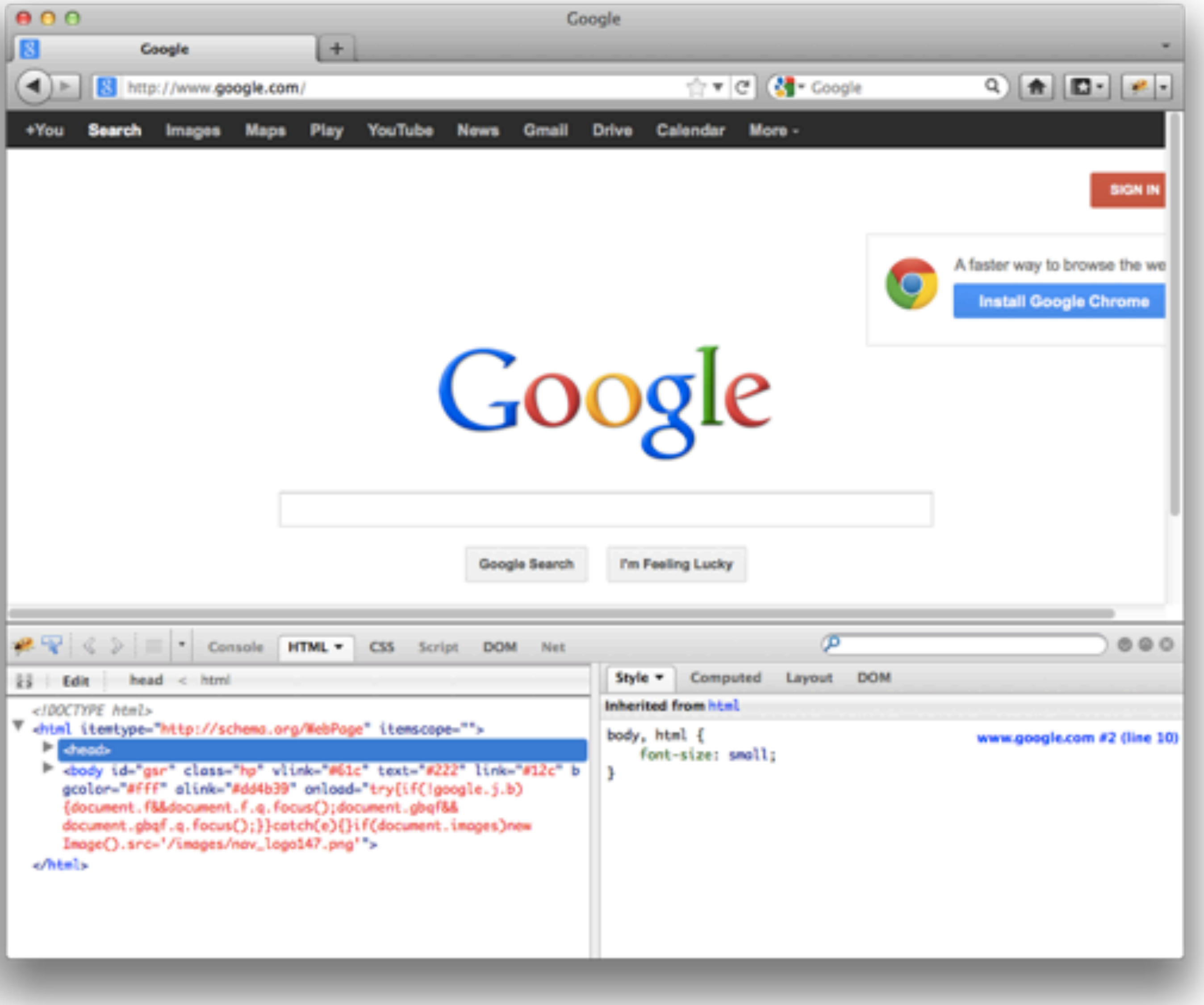
```
element.style {
}

body, html {                         www.google.com/:10
   font-size: small;
}

body {                               www.google.com/:10
   background: ▶ □ #fff;
   color: ■ #222;
}

body, td, a, p, .h {                 www.google.com/:10
```

Chrome



```
<!DOCTYPE html>
▼<html itemtype="http://schema.org/WebPage" itemscope="">
  ▶ <head>
  ▶ <body id="gsr" class="hp" vlink="#61c" text="#222" link="#12c" b
    gcolor="#fff" alink="#dd4b39" onload="try{if(!google.j.b)
    {document.f&&document.f.q.focus();document.gbqf&&
    document.gbqf.q.focus();}}catch(e){}if(document.images)new
    Image().src='/images/nav_logo147.png'">
</html>
```

```
Inherited from html
body, html {                         www.google.com #2 (line 10)
    font-size: small;
}
```

Firebug

JSON

- Looks like Python dictionaries and arrays

```
{
    "kind": "grape",
    "color": "red",
    "quantity": 12,
    "tasty": true
}
```

- Easy to parse in JS and Python

HTTP: Hypertext Transfer Protocol

- A request-response protocol for web servers
- Dynamic web pages respond to the data and parameters the user requests
- Two main request methods:

  GET: most data is transmitted in the URL

  POST: most data is sent in the invisible header
- POST is used to hide sensitive information or to transmit lots of data

Data Formats

- Delimited values

  Comma Separated Values (CSV)

  Tab Separated Values (TSV)

- Markup languages

  Hypertext Markup Language (HTML5 / XML)

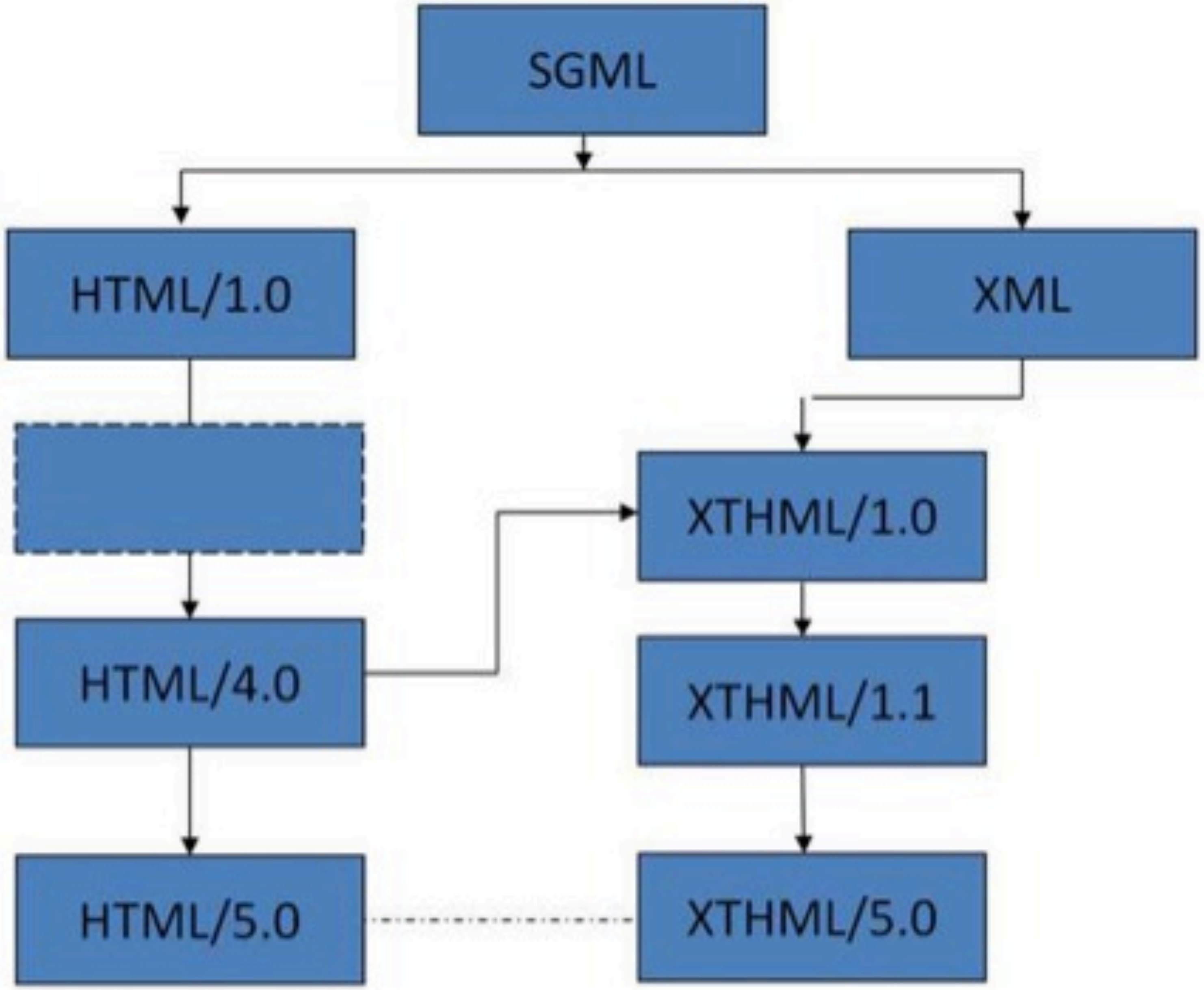
  JavaScript Object Notation (JSON)

  Hierarchical Data Format (HDF5)

- Ad hoc formats

  Graph edge lists, voting records, fixed width files, ...

HTML5



(C) 2013 Technologeeks.com

XML

- Tree structured

  Multiple field values, complex structure

- “Self-describing”: schema is part of the record

```
<menu id=”file” value=”File”>
    <popup>
        <menuitem value=”New” onclick=”CreateNewDoc()” />
        <menuitem value=”Open” onclick=”OpenDoc()” />
        <menuitem value=”Close” onclick=”CloseDoc()” />
    </popup>
</menu>
```

HTML

- A more relaxed version of XML for web pages

```
<!DOCTYPE html>
<html>
    <head>
        <title>My beautiful web page!</title>
    </head>
    <body>
        Here is my content.
    </body>
</html>
```

Tags & Elements

- Tags begin with < and end with >
- Usually occur in pairs

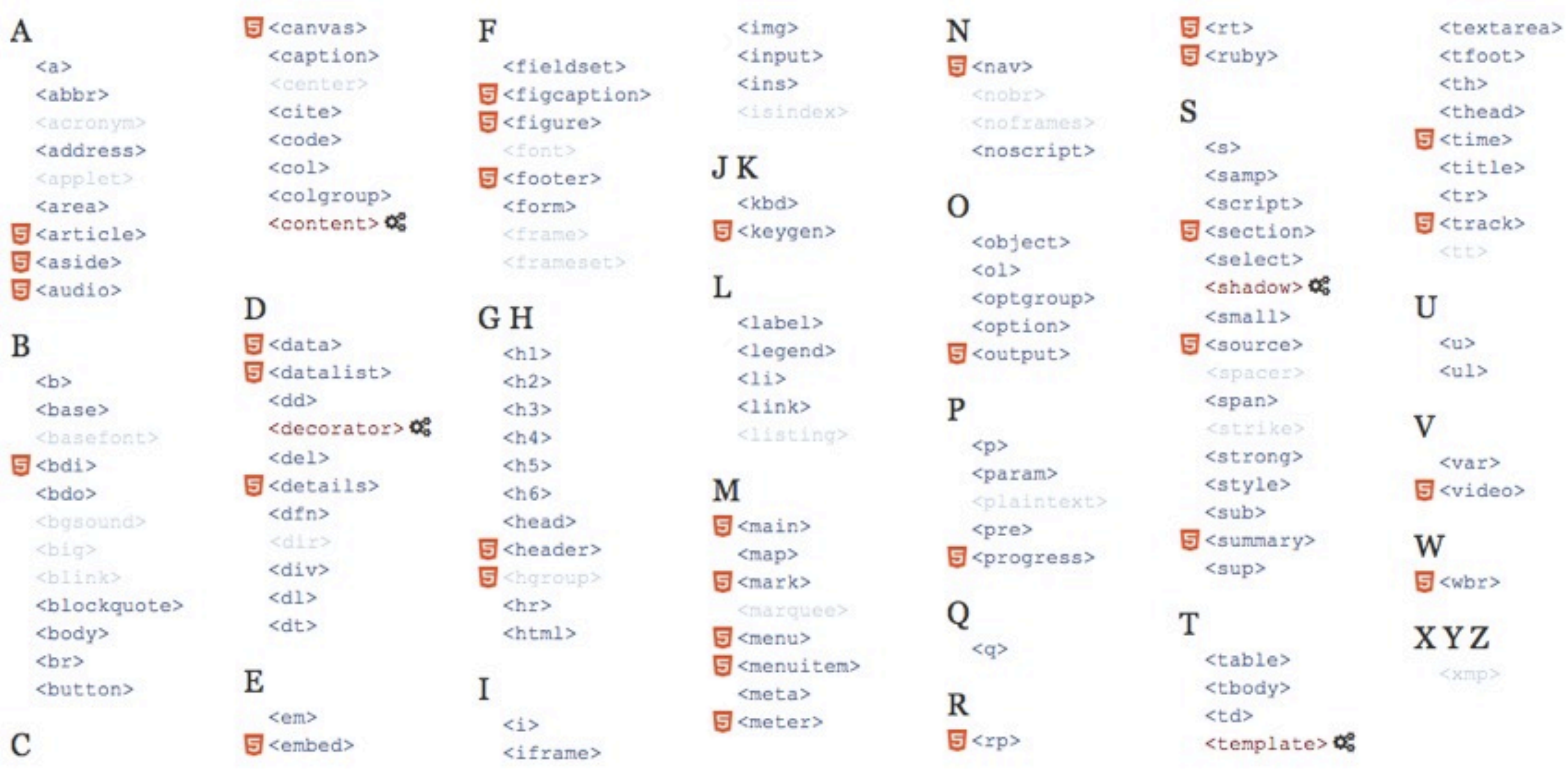
  <p> Wow! </p> creates a new element in the structure
- Elements can be nested

  <p>This is a <em>really</em> interesting paragraph.</p>
- Some tags never occur in pairs

  Usual to use trailing slash, but not necessary

  <img src="photo.jpg" />

HTML Element Reference



A
<a>
<abbr>
<acronym>
<address>
<applet>
<area>
<article>
<aside>
<audio>

B
<b>
<base>
<basefont>
<bdi>
<bdo>
<bgsound>
<big>
<blink>
<blockquote>
<body>

<button>

C
<canvas>
<caption>
<center>
<cite>
<code>
<col>
<colgroup>
<content>

D
<data>
<datalist>
<dd>
<decorator>
<del>
<details>
<dfn>
<dir>
<div>
<dl>
<dt>

E
<em>
<embed>

F
<fieldset>
<figcaption>
<figure>
<font>
<footer>
<form>
<frame>
<frameset>

G H
<h1>
<h2>
<h3>
<h4>
<h5>
<h6>
<head>
<header>
<hgroup>
<hr>
<html>

I
<i>
<iframe>
<img>
<input>
<ins>
<isindex>

J K
<kbd>
<keygen>

L
<label>
<legend>
<li>
<link>
<listing>

M
<main>
<map>
<mark>
<marquee>
<menu>
<menuitem>
<meta>
<meter>

N
<nav>
<nobr>
<noframes>
<noscript>

O
<object>
<ol>
<optgroup>
<option>
<output>

P
<p>
<param>
<plaintext>
<pre>
<progress>

Q
<q>

R
<rp>
<rt>
<ruby>

S
<s>
<samp>
<script>
<section>
<select>
<shadow>
<small>
<source>
<spacer>
<span>
<strike>
<strong>
<style>
<sub>
<summary>
<sup>

T
<table>
<tbody>
<td>
<template>
<textarea>
<tfoot>
<th>
<thead>
<time>
<title>
<tr>
<track>
<tt>

U
<u>
<ul>

V
<var>
<video>

W
<wbr>

XYZ
<xmp>

https://developer.mozilla.org/en/HTML/Element

Noteworthy Tags

- <p> Paragraph of text
- <a> Link, typically displayed as underlined, blue text
- <span> Arbitrary span of text, typically within a larger containing element like <p>
- <div> Arbitrary division within the document used for grouping and containing related elements
- Lists

  <ul> Unordered lists (e.g., bulleted lists)

  <ol> Ordered lists (often numbered)

  <li> List items within <ul> and <ol>

Attributes

- HTML elements can be assigned attributes by including property/value pairs in the opening tag

  ```
  <tagname property="value"></tagname>
  ```

- E.g., a link can be given an href attribute, whose value specifies the URL for that link

  ```
  <a href="http://d3js.org/">The D3 website</a>
  ```

Classes and IDs

- Attributes that can be referenced later to identify specific pieces of content

```
<p class="awesome">Awe-inspiring paragraph</p>
```

- Elements can be assigned to multiple classes

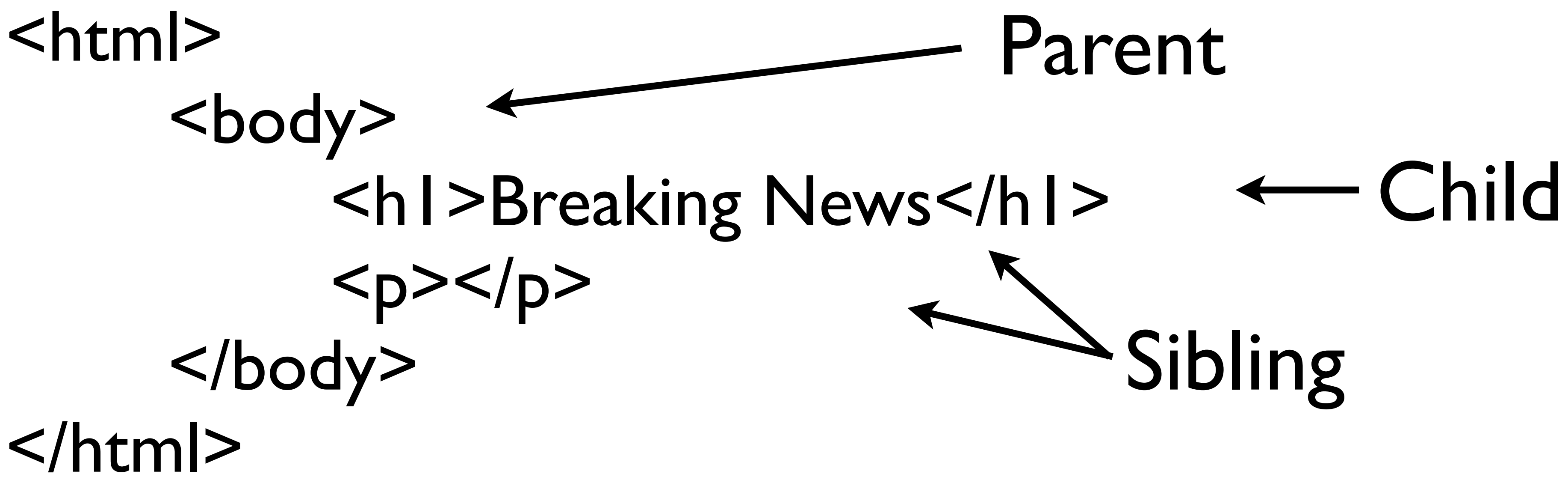
```
<p class="uplifting awesome">Awe-inspiring paragraph</p>
```

- IDs are similar, but only one ID per element, and each ID value only once per page

```
<div id="content">
	<div id="button"></div>
</div>
```

DOM

Document Object Model: the hierarchical structure of HTML



```
<html>
        <body>
                <h1>Breaking News</h1>
                <p></p>
        </body>
</html>
```

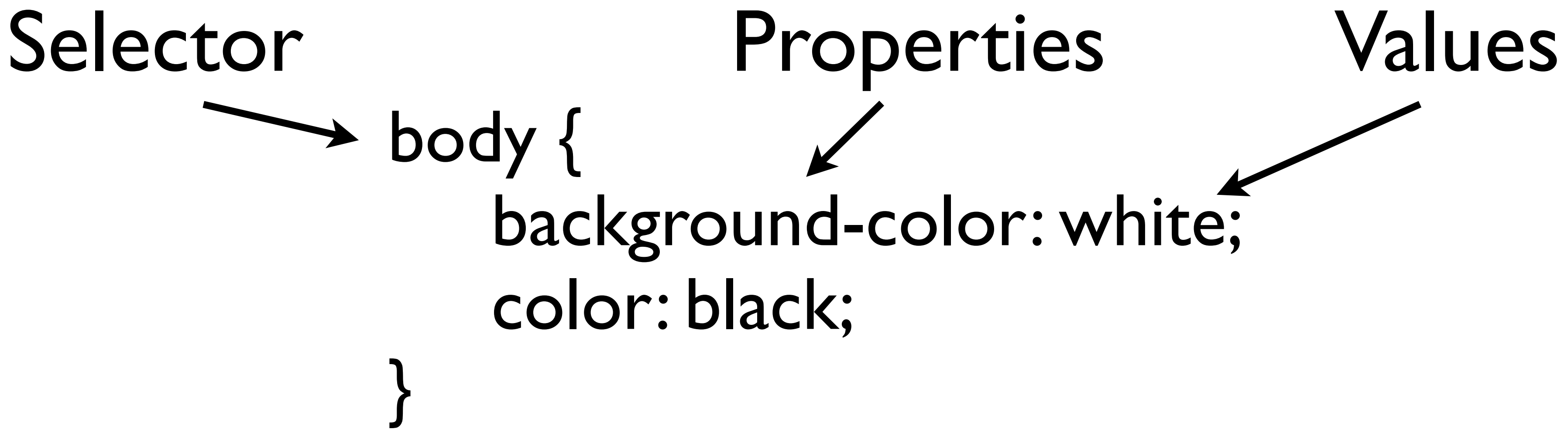
Parent

Child

Sibling

Cascading Style Sheets (CSS)

- Style the visual presentation of DOM elements



Selector

Properties

Values

```
body {
    background-color: white;
    color: black;
}
```

- Selectors come in different flavors

  Type selectors: h1, em, div, ...

  Class selectors: .caption, .label, .axis.x, ...

  ID selectors: #header, #nav, #button, ...

Referencing CSS

- Link to external CSS style sheets

```
<html>
    <head>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <p>Would you say I have style?</p>
    </body>
</html>
```

Web Scraping

Web Scraping

"Data scraping is a technique in which a computer program extracts data from human-readable output coming from another program.

Web pages are built using text-based mark-up languages (HTML and XHTML), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human end-users and not for ease of automated use. Because of this, tool kits that scrape web content were created. A web scraper is an API to extract data from a web site."

wikipedia

What data is OK to scrape?

Public, non-sensitive, anonymized, fully referenced information (always cite sources)

Data Cleanup

Sources of Error in Data

- Data entry errors (e.g., telephone call centers)
- Measurement errors (e.g., improper sampling)
- Distillation errors (e.g., smoothing due to noise)
- Data integration errors (e.g., multiple databases)

Data Cleaning

"In our experience, the tasks of exploratory data mining and data cleaning constitute 80% of the effort that determines 80% of the value of the ultimate data."

T. Dasu and T. Johnson
Authors of *Exploratory Data Mining and Data Cleaning*

Data Manipulations

- Filtering, or subsetting: Remove observations based on some condition
- Transforming: add new variables or modify existing variables (e.g., log-transforming)
- Aggregating: collapse multiple values into a single value (e.g., summing or taking means)
- Sorting: change the order of values