The GovData Project

MIT-Harvard Winter Course

2011.01.10-21
Q: Why is Government data interesting?
Why is Government data interesting?

Bureau of Economic Analysis (BEA)

“The Bureau of Economic Analysis (BEA) is an agency of the United States Department of Commerce (DOC) that compiles data on a wide variety of important economic indicators.”

... like the GDP figures you always hear about.
Why is Government data interesting?

... “the commerce department reports that the GDP grew by 1.2 % this quarter, prompting fears that ...”

That stuff refers to this dataset.
Why is Government data interesting?

Table 1.1.1. Percent Change From Preceding Period in Real Gross Domestic Product.

GDP volatility has decreased over time, even including most recent recession.
Why is Government data interesting?

Table 1.1.1. Percent Change From Preceding Period in Real Gross Domestic Product.

Durable goods are more volatile than nondurable goods, ... although this trend has attenuated recently.
Why is Government data interesting?

Table 1.1.5. Gross Domestic Product (absolute numbers)

Units: Billions of dollars
Why is Government data interesting?

Table 1.1.5. Gross Domestic Product (absolute numbers)

Durables roughly kept pace until early 1940s ...
Why is Government data interesting?

Table 1.1.5. Gross Domestic Product (absolute numbers)

Exports held on until Q1 1981, and then, wham ..
Why is Government data interesting?

Table 1.1.5. Gross Domestic Product (absolute numbers)

Defense / nondefense spending gap is at its highest, ever. (including WWII, Iraq II, &c)
Why is Government data interesting?

Table 1.1.10. Percentage Shares of Gross Domestic Product

Clearly a relative scale ...
Why is Government data interesting?

Table 1.1.10. Percentage Shares of Gross Domestic Product

Goods vs. services seems almost like a zero-sum-game ...

and the sides changed Q1 1970.
Why is Government data interesting?

Table 1.1.10. Percentage Shares of Gross Domestic Product

Exports vs. imports:
- until 1950s, exports > imports
- balanced until mid-70s
- significant account deficit thereafter

But its NOT a zero-sum game ... they’re coupled.
Module 0. Introduction

(needed background: None)
Intro: The Two Audiences

Data Professionals

“Front-end users.” (a.k.a. “the public”, “the American people” &c)
Intro: The Two Audiences

Data Professionals
- Academics
- Policy entrepreneurs
- Journalists
- Developers
- Data archives

“Front-end users.” (a.k.a. “the public”, “the American people” &c)
- Students
- Congressional staffers
- Newspaper readers (esp. online)
- Googlers (e.g. everyone)
- Academics & Policy entreprenuers
Intro: Huge Recent Hype

Open Government Initiative | The White House
Dec 29, 2010...Earlier this month, the White House Open Government Initiative and General Services Administration requested your input to help design an...

Open Government Initiative Blog | The White House
Earlier this month, the White House Open Government Initiative and General...www.whitehouse.gov/open/blog - Cached - Similar

Open Government Initiative
Open Government Initiative - Transparency, public participation, and collaboration improvements will strengthen NARA's crucial role in supporting democracy,...www.archives.gov/open/ - Cached - Similar

Open Government Initiative
Date: 04/07/2010 Description: Open Government Plan - State Dept Image The Department of State has a longstanding history and commitment to sharing...

Open Government Initiative
On December 8, 2009, the Office of Management and Budget (OMB) issued an Open Government Directive requiring Federal agencies to take specific actions to...

Sep 20, 2010...Thanks for your comments on NSF's Open Government Directive Plan: On September 20, 2010, ... Federal Open Government Initiatives

Open Government Initiative
A Treasury Open Government Initiative: www.PRACcomment.gov. Treasury launched the first government site to collect public comments on the Paperwork Reduction ...

White House launches open government initiative - Nextgov
Our Theses:

1. Something important has been started.

2. There are some important glaring gaps that are not really being addressed.

3. You don’t need to be the US Federal Government to address them.

4. But you do need a substantial, organized, open source effort.
The point of the Govdata project

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
Course Agenda:

- Give a brief history & overview of government data initiatives
- Explain the Big Gap, as we see it.
- Tell you a bit about our solution.
- Teach you parsing and visualization tools through:
  - boring lectures engaging presentations
  - extended hands-on coding sessions, to really bring it home
- Create the nucleus of an open source community.
Schedule

Day 1:  Government Data 101: It’s power & its flaws
        The GovData Project: Principles
        Web scraping & data cleaning, I

Day 2:  Web scraping & data cleaning, II
        A short introduction to MongoDB & Apache Solr
        The GovData APIs
        Hands-On Coding Project Overview

Day 3:  Hands-on Coding Sessions I & II

Day 4:  Introduction to “Well Structured Javascript”
        Hands-on Coding Sessions III

Day 5:  Hands-on Coding Sessions IV & V
Module 1. Government Data 101

(needed background: None)
The census tells us who we are and where we are going as a nation. The census helps our communities determine where to build everything from schools to superhighways, and it assists states and localities in determining the lines of legislative districts and re-allocating the seats each state holds in Congress.

### 1990 Census of Population

CP-1 - General Population Characteristics

CP-2 - Social and Economic Characteristics

CP-3-1 - The Foreign-Born Population in the United States [PDF, 9.3MB]

CP-3-2 - Ancestry of the Population in the United States [PDF, 13.4MB]

CP-3-3 - Persons of Hispanic Origin in the United States [PDF, 6.7MB]

CP-3-4 - Education in the United States [PDF, 9.5MB]

CP-3-5 - Asians and Pacific Islanders in the United States [PDF, 6.3MB]

CP-3-6 - Characteristics of the Black Population [PDF, 8.2MB]

CP-3-7-1 - Characteristics of American Indians by Tribe and Language: Section 1 [PDF, 14.4MB]

CP-3-7-2 - Characteristics of American Indians by Tribe and Language: Section 2 [PDF, 10.7MB]

CP-S-1-1 - Detailed Occupation and Other Characteristics From the EEO File for the United States [PDF, 2.6MB]

CP-S-1-2 - Detailed Ancestry Groups for States [PDF, 2.5MB]
Collecting Data since ...

U.S. Census Bureau

CENSUS OF POPULATION AND HOUSING
1980 Census

The census assistance...

For most of the United States, "Census Day" for the 1980 enumeration was April 1, 1980. As in past censuses, all questionnaires were to be completed by this date.

The 1980 census also included two small surveys: the Components of Inventory Change Survey, which obtained information on counts and changes of inventories of housing units; and the Housing Survey, requesting data on mortgages, shelter costs, selected housing characteristics, and owner characteristics.

1980 Census of Population

CP-1 - 1980 Census of Population
CP-2 - Vol. I. Characteristics of the Population
CP-3-1 Vol. II. Subject Reports
CP-3-2 Supplementary Reports

1980 Census of Housing

CP-3-3
CP-3-4
CP-3-5
CP-3-6
CP-3-7

NOTE: Currently, none of the 1980 Census of Housing reports are available online. For a listing of reports that were published for the 1980 Census of Housing, please contact your local Federal Depository Library.

1980 Census of Population and Housing

CP-3-7. Preliminary Population and Housing Unit Counts. (PHC80-P-1 to -56). (unavailable)
CP-S-1. Final Population and Housing Unit Counts. (PHC80-V). (unavailable)
CP-S-1. Block Statistics. [Microfiche]. (unavailable)
When planning for the 1970 census, the need for an accurate count of the population was even greater than in the past because of the increasing tendency for government officials and others of the potential effects of census undercount. [More/Less]
The Fourteenth Census Act of July 2, 1909, provided for the 1920 and subsequent censuses; however, numerous minor changes were sought prior to the beginning July 1, 1919. During this 3-year period, the act provided for an increased workforce at the Census Bureau’s headquarters in Washington, DC.

**1920 Census Information**

**Volume 1:** Population, 1920. Number and distribution of inhabitants, 695 pp., plus one 4-color foldout map. * - PDF, ZIP [154.9 MB]

**Volume 2:** Population, 1920. General report and analytical tables. 1410 pp * - PDF, ZIP [283.1 MB]

**Volume 3:** Population, 1920. Composition and characteristics of the population by states. 1253 pp., plus one 4-color foldout map


**Volume 5:** Agriculture. General report and analytical tables. 935 pp * - PDF, ZIP [195.2 MB]

**Volume 6:** Agriculture. Report for the states, with statistics for counties and a summary for the United States and the North, South, and South West. 765 pp. * - PDF, ZIP [154.8 MB]

**Volume 7:** Irrigation and drainage. General report and analytical tables and reports for states, with statistics for counties. 741 pp.,
The sixth census was governed by the same general provisions of law as in 1830. Under the provisions of an act of March 3, 1839 (and amended receipts from enumerators by November 1, 1840, one of which was to be sent to the Secretary of State by December 1, 1840.

No population questionnaire was prescribed by the Congress—the design of the questionnaire was left to the discretion of the Secretary of State.

**1840 Census Information**

*Volume 1. Sixth Census or enumeration of the inhabitants of the United States, as corrected at the Department of State, in* 

*Volume 2. Statistics of the United States of America, as collected and returned by the marshals of the several judicial districts* 

*Volume 3. Compendium of the Enumeration of the inhabitants and statistics of the United States, as obtained at the Department of State.*

*Volume 4. A census of pensioners for revolutionary or military services; with their names, ages, and places of residence, at* 

For online viewing, each book with a total file size over 10 MB has been divided into several internally linked PDF files. These same files have been formatted for online viewing. The total file size of the online PDF files is similar to that of their ZIP counterparts.

For access to volumes not available, please contact your local Federal Depository Library.
Collecting Data since ...

U.S. Census Bureau

CENSUS OF POPULATION AND HOUSING
1790 Census

The first enumeration began on Monday, August 2, 1790, little more than a year after the inauguration of President Washington. The U.S. judicial districts under an act that, with minor modifications and extensions, governed census-taking through 1840 [the district was established by a decision of the U.S. Supreme Court], there to remain for the inspection of all concerned...” and that “the aggregate amount of each description of persons in each household of the following descriptions: Free White males of 16 years and upward (to assess the country...

1790 Census Information

Return of the whole number of persons within the several districts of the United States..." - PDF, ZIP

Heads of Families at the First Census of the United States in the year 1790

Connecticut - PDF, ZIP [60.6 MB]
Maine - PDF, ZIP [20.3 MB]
Maryland - PDF, ZIP [38.2 MB]
Massachusetts - PDF, ZIP [81.4 MB]
New Hampshire - PDF, ZIP [30.5 MB]
New York - PDF, ZIP [74.0 MB]
North Carolina - PDF, ZIP [107.4 MB]
Pennsylvania - PDF, ZIP [94.8 MB]
Rhode Island - PDF, ZIP [14.3 MB]
South Carolina - PDF, ZIP [54.2 MB]
Vermont - PDF, ZIP [21.9 MB]
Virginia - PDF, ZIP [67.7 MB]

For online viewing, each book with a total file size over 10 MB has been divided into several internally linked PDF files for viewing. The total file size of the online PDF files is similar to that of their ZIP counterparts.

Users with visual impairments who have difficulty accessing PDF documents may access technical support or call 301-718-8000.

[PDF] or [ZIP] denotes a file in Adobe's Portable Document Format. To view the file, you will need the Adobe® Reader.
Fedstats in late 1990s

Celebrating over 10 years of making statistics from more than 100 agencies available to citizens everywhere

Links to statistics

- **Topic Links - A To Z** - Direct access to statistical data on topics of your choice.
- **MapStats** - Statistical profiles of States, counties, cities, Congressional Districts, and Federal judicial districts.
- **Statistics By Geography From U.S. Agencies** - International comparisons, national, State, county, and local.
- **Statistical Reference Shelf** - Published collections of statistics available online including the Statistical Abstract of the United States.
- **Search** across agency websites.

Links to statistical agencies

- **Agencies Listed Alphabetically** with descriptions of the statistics they provide and links to their websites, contact information, and key statistics.
- **Agencies by subject** - Select a subject:
  - [Agriculture](#)
  - [Submit](#)
- **Press Releases** - The latest news and announcements from individual agencies.
- **Kids' Pages** on agency websites.
- **Data Access Tools** - Selected agency online databases.

+ uncountably many other comparatively low-profile initiatives ...
MEMORANDUM FOR THE HEADS OF EXECUTIVE DEPARTMENTS AND AGENCIES

SUBJECT: Transparency and Open Government

My Administration is committed to creating an unprecedented level of openness in Government. We will work together to ensure the public trust and establish a system of transparency, public participation, and collaboration. Openness will strengthen our democracy and promote efficiency and effectiveness in Government.
December 8, 2009

MEMORANDUM FOR THE HEADS OF EXECUTIVE DEPARTMENTS AND AGENCIES

FROM: Peter R. Orszag
       Director

SUBJECT: Open Government Directive

In the Memorandum on Transparency and Open Government, issued on January 21, 2009, the President instructed the Director of the Office of Management and Budget (OMB) to issue an Open Government Directive. Responding to that instruction, this memorandum is intended to direct executive departments and agencies to take specific actions to implement the principles of transparency, participation, and collaboration set forth in the President’s Memorandum. This Directive was informed by recommendations from the Federal Chief Technology Officer, who solicited public comment through the White House Open Government Initiative.

The three principles of transparency, participation, and collaboration form the...
Due to efforts and passions at various offices in the federal government, there has been an explosion of interest in “open data.”

The Open Government Initiative

Vivek Kundra
Federal CIO

Aneesh Chopra
Federal CTO
The Open Government Initiative

Due to efforts and passions at various offices in the federal government, there has been an explosion of interest in “open data”

Holy alliance of lawyers, economists, and programmers ...
The Open Government Initiative

CIO Council Members

By Department

Department of Agriculture

Mr. Chris Smith
Chief Information Officer
1400 Independence Avenue, SW
Room 414W
Washington, DC 20250
chris.smith@usa.gov
Office Phone: 202-720-1031
Fax: 202-720-1031

Mr. Charles McClain
Deputy Chief Information Officer
1400 Independence Avenue
432W Whitten Building
Washington, DC 20416
charles.mcclain@usda.gov
Office Phone: 202-720-5865
Fax: 202-720-3733

Department of the Air Force

Lt. Gen. William T. Lord
Chief of Warfighting Integration and Chief Information Officer
1800 Air Force Pentagon
Room 4E1050
Washington, DC 20330-1800
workforce@pentagon.af.mil
Office Phone: 703-695-6829
Fax: 703-692-7512

Mr. Daniel F. McMillin
Deputy Chief of Warfighting Integration & Deputy Chief Information Officer
1800 Air Force Pentagon
Room 4E1050
Washington, DC 20330-1800
daniel.mcmillin@pentagon.af.mil
Office Phone: 703-697-1605
Fax: 703-692-7512

Department of Health & Human Services

Mr. Michael W. Carleton
Chief Information Officer
200 Independence Avenue, SW
Hubert Humphrey Building, Room 537H
Washington, DC 20201
michael.carleton@hhs.gov
Office Phone: 202-690-5162
Fax: 202-690-5715

Mr. John Teeter
Deputy Chief Information Officer
200 Independence Avenue, SW
Hubert Humphrey Building, Room 537H
Washington, DC 20201
john.teeter@hhs.gov
Office Phone: 202-690-5162
Fax: 202-690-5715

Department of Homeland Security

Mr. Richard Spies
Chief Information Officer
301 7th Street
Washington, DC 20528
cio@dhs.gov
Office Phone: 202-447-3735

Ms. Margie Graves
Deputy Chief Information Officer
Department of Homeland Security
301 7th Street
Washington, DC 20528
cio@dhs.gov
Office Phone: 202-447-3735
The Open Government Initiative

data.gov
data.gov.uk
datasf.org
data.dc.gov
Mass ODI
NYC Datamine
toronto.ca/open
socrata.com
sunlight labs
The Open Government Initiative

Data.gov Catalogs
Use the Data.gov catalog below to access U.S. Federal Executive Branch datasets. Click on the name of a dataset to view additional metadata for that dataset. By accessing the data catalogs, you agree to the Data Policy. Data.gov offers data in three ways: through the "raw" data catalog, using tools and through the geodata catalog. The "Raw" Data Catalog provides an instant download of machine readable, platform-independent datasets while the Tools Catalog provides hyperlinks which may lead to agency tools or agency web pages that allow you to mine datasets.

data.gov ---> the professionals ---> front-end users
The Open Government Initiative

data.gov.uk
Opening up government

Looking for something specific, or just want to know more about how Government spends your money? You'll find 5,300 datasets to help you get answers.

Browse by publisher
- Advisory, Conciliation and Arbitration Service (0)
- Arts and Humanities Research Council (0)
- Arts Council England (1)
- Ashfield District Council (1)
- Ashton, Leigh and Wigan Community Healthcare NHS Trust (0)
- Attorney General’s Office (3)
- Audit Commission (0)

Browse by nation
- England
- Northern Ireland
- Scotland
- Wales

EDITOR’S PICK
COINS – HM Treasury’s database of public spending

Popular tags
- health (1,988)
- care (1,324)
- communities (1,306)
- child (1,130)
- health-and-social-care (1,104)
- transparency (1,062)
- children (1,058)
- local-government (1,058)
The Open Government Initiative

What is DataSF?

DataSF is a clearinghouse of datasets available from the City & County of San Francisco. Our goal in releasing this site is: (1) improve access to data (2) help our community create innovative apps (3) understand what datasets you'd like to see (4) get feedback on the quality of our datasets.

Datasets

Film Locations in San Francisco
Categorized under Admin & Finance Views: 1123 (http://www.filmsf.org)

Description: If you love movies, and you love San Francisco, you’re bound to love this -- a listing of filming locations of movies shot in San Francisco from 1924 - 2010. You’ll find the titles, locations, fun facts, names of the director, writer, actors, and studio for most of these films.

Agency Name: San Francisco Film Commission
Date Released: 12/21/2010
Time Period: 1924 - 2010
Frequency: Annual
Location of dataset: Film Locations in SF Dataset
Format: excel
Location of Data Dictionary: www.filmsf.org

Tags: film locations movies actors directors

Read More | 1 Comment  ShareThis

San Francisco Wind Monitoring Data
Categorized under Environment Views: 275 (Editoral)
The Open Government Initiative

National Data Catalog

This project is funded by Sunlight. Became a project on August 12, 2009.

The National Data Catalog aims to be a complete catalog of all data sets and APIs that are either put out by the government or are derived from the government. Scoped to all government levels (federal, state, and local), and all branches (executive, legislative, judicial), NDC will be the one-stop shop for developers, researchers, and investigative journalists interested in government data.

NDC will tap into the social benefits of having users come together around common interests. More than just a catalog, it will be a place for community-supported documentation about government data.

Tags
- data
- ruby

Project Participants

Lead
- David James

Team
- Luis Montaner

Join This Project?

If you're interested in helping out on
The Open Government Initiative

Code for America Fellowship PSA: "What if...?"

Tim O'Reilly
Founder and CEO, O'Reilly Media

Code for America was founded to help the brightest minds of the Web 2.0 generation transform city governments. Cities are under greater pressure than ever, struggling with budget cuts and outdated technology. What if, instead of cutting services or raising taxes, cities could leverage the power of the web to become more efficient, transparent, and participatory?

We believe there is a wealth of talent in the web industry eager to contribute to the rebuilding of America. Code for America gives them the means.

City governments, learn more.
The Open Government Initiative

API = “application programming interface”
= URL access scheme, in web context
http://mysite.com/?title=MyLife&author=Me

Catalogs = organized metadata, searchable

Apps = cool iPad stuff
The Datasets Themselves

There used to be this great page on data.gov ...

Listing all the participating agencies -- that is, all the actual data sources.

Unfortunately, it’s no longer accessible.

Hi Dan,

No problem! Glad I could help. Keep me in the loop about how things progress with NTIA.

Not sure where the page you’re looking for went – I did a quick search myself and didn't see anything...sorry!

Best,

Advisor to the Chief Technology Officer
Executive Office of the President | Office of Science and Technology Policy
202.456.6028
ostp.eop.gov
The Datasets Themselves

Let’s explore it a bit ..
The Datasets Themselves

Data has a natural **source tree hierarchy:**

Agency  Subagency  Program  Topic  ...  Dataset

Depending on how you count .... 2 ~ 10 TB
Claim:
The existing situation has a problem fundamental flaw.
The Fundamental Flaw

Catalogs:

NOT

Data collections:
The Fundamental Flaw

Catalogs: centralized formatting/storage of supplied metadata

NOT

Data collections: centralized storage of the data itself
The Fundamental Flaw

Employment, Hours, and Earnings from the Current Employment Statistics survey (National)

<table>
<thead>
<tr>
<th>DATASET SUMMARY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Agency:</strong></td>
<td>Department of Labor</td>
</tr>
<tr>
<td><strong>Sub-Agency:</strong></td>
<td>US Bureau of Labor Statistics</td>
</tr>
<tr>
<td><strong>Category:</strong></td>
<td>Labor Force, Employment, and Earnings</td>
</tr>
<tr>
<td><strong>Date Released:</strong></td>
<td>1909</td>
</tr>
<tr>
<td><strong>Date Updated:</strong></td>
<td>Monthly</td>
</tr>
<tr>
<td><strong>Time Period:</strong></td>
<td>Employment and wages for the pay period including the 12th of the month</td>
</tr>
<tr>
<td><strong>Frequency:</strong></td>
<td>Monthly</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
<td>Each month the Current Employment Statistics (CES) program surveys about 150,000 businesses and government agencies, representing approximately 390,000 individual worksites, in order to provide detailed industry data on employment, hours, and earnings of workers on nonfarm payrolls.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATASET RATINGS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall</strong></td>
<td>★★★★★★ (0 votes)</td>
</tr>
<tr>
<td><strong>Data Utility</strong></td>
<td>★★★★★☆ (0 votes)</td>
</tr>
<tr>
<td><strong>Usefulness</strong></td>
<td>★★★★★☆ (0 votes)</td>
</tr>
<tr>
<td><strong>Ease of Access</strong></td>
<td>★★★★★☆ (0 votes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DOWNLOAD INFORMATION</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>CSV/TXT</td>
</tr>
<tr>
<td>XLS</td>
<td>KML/KMZ</td>
</tr>
<tr>
<td>Shapefile</td>
<td>Maps</td>
</tr>
</tbody>
</table>

Comment on these data: [Submit] (Privacy Policy)
The Fundamental Flaw
### Gross Domestic Product (GDP)

- **News Release**: [Gross Domestic Product](#) | [PDF version of the Gross Domestic Product release](#) includes highlights, technical note, and associated tables
- **Current-dollar and "real" GDP** (Excel + 25KB)
- **Percent change from preceding period** (Excel + 24KB)
- **Interactive Tables**: [GDP and the National Income and Product Account (NIPA) Historical](#)
- **Selected NIPA Tables**:
  - [Text format](#) (Text + 227KB)
  - [Comma delimited format](#) (CSV + 2230KB)
  - [Portable document format](#) (PDF + 608KB)

**Now Available**: [Information on the comprehensive revision of the GDP accounts](#)

### Personal Income and Outlays

- **News Release**: [Personal Income and Outlays](#) includes highlights and associated tables
- **Interactive Tables**: [National Income and Product Accounts Tables](#)
  - Charts and interactive tables:
    - [Comparison of Personal Saving in the NIPAs with Personal Saving in the Flow of Funds](#)
    - Comparison of the Personal Consumption Expenditures (PCE) Price Index with the Consumer Price Index (CPI)
      - *Note*: This table is being redesigned as a result of changes in the presentation of the PCE estimates of the 2009 comprehensive NIPA revision. The PCE/CPI reconciliation table will be available by information on the changes in the presentation of the PCE estimates.

**Now Available**: [Information on the comprehensive revision of the GDP accounts](#)
Interative Access To National Income and Product Accounts Tables

To display and download a NIPA table, select one of the methods below or go directly to our Advanced... 

- Choose one of the Frequently Requested NIPA Tables
- Choose a table from a list of Selected NIPA Tables
- Choose a table from a list of All NIPA Tables
- Choose a table from the Keyword Index

What's New

February 1, 2010
Monthly NIPA tables have been updated to include the release of December 2009

January 29, 2010
Selected Tables have been updated to include the advance estimate of the 4th quarter of 2009.

December 23, 2009
Monthly NIPA tables have been updated to include the release of November 2009.

December 22, 2009
Selected Tables have been updated to include the third estimate of the 3rd quarter of 2009.

November 25, 2009
Monthly NIPA tables have been updated to include the release of October 2009.
Section 1 - Domestic Product and Income

Table 1.1.1. Percent Change From Preceding Period in Real Gross Domestic Product (A) (Q)
Table 1.1.2. Contributions to Percent Change in Real Gross Domestic Product (A) (Q)
Table 1.1.3. Real Gross Domestic Product, Quantity Indexes (A) (Q)
Table 1.1.4. Price Indexes for Gross Domestic Product (A) (Q)
Table 1.1.5. Gross Domestic Product (A) (Q)
Table 1.1.6. Real Gross Domestic Product, Chained Dollars (A) (Q)
Table 1.1.6A. Real Gross Domestic Product, Chained (1937) Dollars (A)
Table 1.1.6B. Real Gross Domestic Product, Chained (1952) Dollars (A) (Q)
Table 1.1.6C. Real Gross Domestic Product, Chained (1972) Dollars (A) (Q)
Table 1.1.6D. Real Gross Domestic Product, Chained (1987) Dollars (A) (Q)
Table 1.1.7. Percent Change From Preceding Period in Prices for Gross Domestic Product (A) (Q)
Table 1.1.8. Contributions to Percent Change in the Gross Domestic Product Price Index (A) (Q)
Table 1.1.9. Implicit Price Deflators for Gross Domestic Product (A) (Q)
Table 1.1.10. Percentage Shares of Gross Domestic Product (A) (Q)
Table 1.2.1. Percent Change From Preceding Period in Real Gross Domestic Product by Major Type of
Table 1.2.2. Contributions to Percent Change in Real Gross Domestic Product by Major Type of Produ
Table 1.2.3. Real Gross Domestic Product by Major Type of Product, Quantity Indexes (A) (Q)
Table 1.2.4. Price Indexes for Gross Domestic Product by Major Type of Product (A) (Q)
Table 1.2.5. Gross Domestic Product by Major Type of Product (A) (Q)
Table 1.2.6. Real Gross Domestic Product by Major Type of Product, Chained Dollars (A) (Q)
Table 1.3.1. Percent Change From Preceding Period in Real Gross Value Added by Sector (A) (Q)
Table 1.3.3. Real Gross Value Added by Sector, Quantity Indexes (A) (Q)

Monthly NIPA tables have been updated to include the release of October 2009.
### National Income and Product Accounts Table

**Table 1.1.4. Price Indexes for Gross Domestic Product**

[Index numbers, 2005=100]  Seasonally adjusted

Today is: 2/3/2010  Last Revised on January 29, 2010  Next Release Date February 26, 2010

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gross domestic product</strong></td>
<td><strong>1947</strong></td>
<td><strong>1948</strong></td>
<td><strong>1949</strong></td>
<td><strong>1950</strong></td>
<td><strong>1951</strong></td>
<td><strong>1952</strong></td>
<td><strong>1953</strong></td>
<td><strong>1954</strong></td>
<td><strong>1955</strong></td>
<td><strong>1956</strong></td>
<td><strong>1957</strong></td>
<td><strong>1958</strong></td>
<td><strong>1959</strong></td>
<td><strong>1960</strong></td>
<td><strong>1961</strong></td>
<td><strong>1962</strong></td>
<td><strong>1963</strong></td>
<td><strong>1964</strong></td>
</tr>
<tr>
<td><strong>Durable goods</strong></td>
<td>41.332</td>
<td>41.712</td>
<td>41.968</td>
<td>42.491</td>
<td>42.815</td>
<td>43.377</td>
<td>44.886</td>
<td>44.967</td>
<td>44.981</td>
<td>43.929</td>
<td>43.795</td>
<td>44.131</td>
<td>44.743</td>
<td>44.981</td>
<td>43.929</td>
<td>43.795</td>
<td>44.131</td>
<td>44.743</td>
</tr>
<tr>
<td><strong>Net exports of goods and services</strong></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Services</strong></td>
<td>16.207</td>
<td>15.503</td>
<td>15.142</td>
<td>15.020</td>
<td>15.141</td>
<td>15.234</td>
<td>15.357</td>
<td>15.468</td>
<td>15.533</td>
<td>15.620</td>
<td>15.671</td>
<td>15.726</td>
<td>15.681</td>
<td>15.750</td>
<td>15.823</td>
<td>15.823</td>
<td>15.823</td>
<td>15.823</td>
</tr>
<tr>
<td><strong>Net exports</strong></td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
### The Fundamental Flaw

#### Table 1.1.4. Price Indexes for Gross Domestic Product

[Index numbers, 2005=100]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td>Durable goods</td>
<td>41.332</td>
<td>41.712</td>
<td>41.945</td>
<td>42.491</td>
<td>42.815</td>
<td>43.377</td>
<td>44.886</td>
</tr>
<tr>
<td>11.</td>
<td>Equipment</td>
<td>33.467</td>
<td>34.877</td>
<td>35.661</td>
<td>35.978</td>
<td>35.791</td>
<td>36.843</td>
<td>38.375</td>
</tr>
<tr>
<td>13.</td>
<td>Change in private inventories</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>Net exports of goods and services</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td>Services</td>
<td>16.207</td>
<td>15.503</td>
<td>15.142</td>
<td>15.02</td>
<td>15.141</td>
<td>15.234</td>
<td>15.357</td>
</tr>
</tbody>
</table>
The Fundamental Flaw
The Fundamental Flaw

What is DataSF?

DataSF is a clearinghouse of datasets available from the City & County of San Francisco. Our goal in releasing this site is:
1. improve access to data
2. help our community create innovative apps
3. understand what datasets you'd like to see
4. get feedback on the quality of our datasets.

looking for a dataset or app that you'd like to see?
<tell us>

Numbers quit unexpectedly while using the SFCompatibility plug-in.
Click Reopen to open the application again. Click Report to see more detailed information and send a report to Apple.

Agency Name: San Francisco Film Commission
Date Released: 12/21/2010
Time Period: 1924 - 2010
Frequency: Annual
Location of dataset: Film Locations in SF Dataset
Format: excel
Location of Data Dictionary: www.filmsf.org

1 Comment
Tags: film locations movies actors directors

Comments
The Fundamental Flaw

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
The Fundamental Flaw

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
The Fundamental Flaw

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
The Fundamental Flaw

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
The Fundamental Flaw

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
EPA CTO told us:

Open Data Initiative NEVER intends to have centralized storage

Each dep’t is encouraged to choose its own solution. Each dataset its own API. (when there is one)

Initiative is not funded.

And ...

Sunlight Labs projects do not envision doing it either ...

Although ...

DataWiki Project (at RPI)
The Fundamental Flaw

One possible solution is to give developers access to lots of “third party tools” for data cleaning.
The Fundamental Flaw

All Data Collections

1995 Survey of Adults on Probation (SAP)
The 1995 Survey of Adults on Probation was the first nationally representative survey of probationers through a review of probationers’ administrative records, and staff interviews.

2006 Census of State Parole Supervising Agencies
The 2006 Census of State Parole Supervising Agencies collected data from parole agencies in the U.S. on an annual basis, beginning and end of each year, the number of adults entering and exiting parole supervision from parole and parole agencies.

Annual Probation Survey and Annual Parole Survey
Collects data from probation and parole agencies in the U.S. on an annual basis, beginning and end of each year, the number of adults entering and exiting parole supervision and parole agencies.

Annual Survey of Jails
Collects data from a nationally representative sample of local jails on jail inmates.

Capital Punishment (NPS-8)
Provides an annual summary of inmates admitted to and removed from under death penalty and annual changes to those statutes.

Census of Adult Parole Supervising Agencies
The 2006 Census of Adult Parole Supervising Agencies collected information about parole agencies in the U.S. and also inventory of the names and locations of regional/district parole offices.

Census of Federal Law Enforcement Officers
Collects data from all federal law enforcement agencies with arrest and firearm criminal investigation and enforcement, police patrol and response, security and commercial investigations.

Census of Jail Inmates (2005)
Conducted approximately every five to seven years. Based on a complete enumeration, inmate counts, and movements, and persons supervised in the community.

Census of Jails
Conducted approximately every five to seven years. Provides information on individual jails.

Census of Law Enforcement Aviation Units (CLEAU)
Collects data from all law enforcement agencies with 100 or more sworn officers, locations, available assets and range, personnel, expenditures, and functions and each could provide in the case of a nation emergency.

Census of Law Enforcement Training Academies
Collects data on the number and types of staff employed at training facilities, in addition to basic organizational data, the survey collects information on training.
The Fundamental Flaw

... it’s not just a matter of “cleaning” individual files.

If everyone cleans it up in a slightly different way, it’s almost as bad as no cleaning at all.

There is a fundamental lack of “local integration.”

There needs to be a place (at least one) where such things are done in a centralized fashion.
Our realization

There needs to be a place (at least one) where all the data is hosted, and local integration is done in a centralized fashion.

I) it’s totally feasible, with the right tools

II) it’s not *that* expensive

III) for vast majority of data, no permission needed

IV) but it requires really good tools, and a focused community.
Module 2. The GovData Platform

(needed background: working knowledge of basic programming concepts)
The GovData Platform

- to take as much tabular-format government data as possible,
- lightly structure it so it fits into a uniform DB format,
- hosted in a single location, that is updated regularly,
- with standardized temporal and spatial columns,
- and rich metadata;

and then:

- make the backend data accessible through powerful general use APIs with detailed query control
- apply cutting-edge indexing techniques to make individual data slices searchable
- integrate new javascript visualization techniques to design an effective and powerful frontend
The GovData Platform

Intended audience:

- academics & policymakers: exploration and download for off-line analysis (from politics to science ...)

- journalists: for research embedding in their own online frontends

- developers: crafting their own frontends for specialized data projects

- data repos: for harvesting and subsetting data and metadata.
The GovData Project: Sponsorship

Primary sponsor:

The Institute for Quantitative Social Sciences at Harvard University

Co-sponsor:

The Center for Future Civic Media at MIT Media Lab
The GovData Project: Principles

1. Host all the data in one place.

2. Lightly standardize format to a single database.

3. Do “Local synthesis” to make implicit structure explicit.

4. Support “dynamically-grained natural language search”.

5. Unified rich, high-throughput APIs for acquisition and search.

6. Use lightweight, well-structured, pure-javascript frontends.

7. Make data operations transparent, reproducible, and versioned.

8. Be completely open source.  

The GovData Project: Principles

1. Host all the data in one place.  (At least one such place.)

Two reasons:

1) it forces standardization

2) practical issues of back-end speed can be solved ONCE.
2. Lightly standardize format to a single database.

Standard enough so that a single query language and API logic can apply to everything and so that improvements in integration improve everything.

Flexible enough so that data of many different schemata can be hosted all together; without having to solve a VERY hard “strong automated schema integration problem,” or building a semantic web.

... compare to e.g. Wolfram Alpha.

Be STRONGLY standard with regard to space & time.
The GovData Project: Principles

3. Do “local synthesis” to make implicit structure explicit.

... e.g, make formatting into data.
3. Do “local synthesis” to make implicit structure explicit.

### All NIPA Tables

1. Domestic Product and Income
2. Personal Income and Outlays
3. Government Current Receipts and Expenditures
4. Foreign Transactions

Data Availability Legend: (A) Annually  (Q) Quarterly  (M) Monthly

**Section 1 - Domestic Product and Income**

Table 1.1. Percent Change From Preceding Period in Real Gross Domestic Product (A) (Q)
Table 1.1. Contributions to Percent Change in Real Gross Domestic Product (A) (Q)
Table 1.1.3. Real Gross Domestic Product, Quantity Indexes (A) (Q)
Table 1.1.4. Price Indexes for Gross Domestic Product (A) (Q)
Table 1.1.5. Gross Domestic Product (A) (Q)
Table 1.1.6. Real Gross Domestic Product, Chained Dollars (A) (Q)
Table 1.1.6A. Real Gross Domestic Product, Chained (1937) Dollars (A)
Table 1.1.6B. Real Gross Domestic Product, Chained (1952) Dollars (A) (Q)
Table 1.1.6C. Real Gross Domestic Product, Chained (1972) Dollars (A) (Q)
Table 1.1.6D. Real Gross Domestic Product, Chained (1967) Dollars (A) (Q)
Table 1.1.7. Percent Change From Preceding Period in Prices for Gross Domestic Product (A) (Q)
Table 1.1.8. Contributions to Percent Change in the Gross Domestic Product Price Index (A) (Q)
Table 1.1.9. Implicit Price Deflators for Gross Domestic Product (A) (Q)
Table 1.1.10. Percentage Shares of Gross Domestic Product (A) (Q)
Table 1.1.11. Percent Change in Gross Domestic Product by Major Type of Product: One Year Ago (Q)
Table 1.1.2. Percent Change in Gross Domestic Product by Major Type of Product: Major Type of Product by Major Type of Product (A) (Q)
Table 1.1.2. Contributions to Percent Change in Real Gross Domestic Product by Major Type of Product (A) (Q)
Table 1.1.4. Price Indexes for Gross Domestic Product by Major Type of Product (A) (Q)
Table 1.1.5. Gross Domestic Product by Major Type of Product (A) (Q)
Table 1.1.6. Real Gross Domestic Product by Major Type of Product, Chained Dollars (A) (Q)
Table 1.1.7. Percent Change From Preceding Period in Real Gross Value Added by Sector (A) (Q)
Table 1.1.8. Percent Change From Preceding Period in Real Gross Value Added by Sector (A) (Q)
Table 1.1.9. Gross Value Added by Sector, Quantity Indexes (A) (Q)
Table 1.1.10. Gross Value Added by Sector, Value Added (A) (Q)
Table 1.1.11. Gross Value Added by Sector, Chained Dollars (A) (Q)
Table 1.4.1. Percent Change From Preceding Period in Real Gross Domestic Product, Real Gross Domestic Product, Real Gross Domestic Purchases, and Real Final Sales (A) (Q)
Table 1.4.2. Price Indexes for Gross Domestic Product, Gross Domestic Purchases, and Final Sales (A) (Q)
Table 1.4.3. Relation of Real Gross Domestic Product, Gross Domestic Purchases, and Final Sales to
3. Do “local synthesis” to make implicit structure explicit.

... make sure these four data sets all use the column names when appropriate, e.g. for “subject”, “title”, “year”, &c.

Local = relative to the “source hierarchy tree”.

So, you don’t have to understand EPA data to synthesize BEA data.
4. Support “dynamically-grained natural language search”.

Typical search facilities are either too coarse-grained or too fine-grained.

Too coarse: when the search results are at the “dataset level.”

e. g. data.gov

(Q: really, there’s only 2 datasets for GDP?)
4. Support “dynamically-grained natural language search”.

Typical search facilities are either too coarse-grained or too fine-grained.

Too fine: when the search results are at the “record level.”

e. g. EPA envirofacts.
The GovData Project: Principles

4. Support “dynamically-grained natural language search”.

Typical search facilities are either too coarse-grained or too fine-grained.

Instead, what “professional” users really want usually is a dynamically-grained intermediate:

Subslices of the data, defined by categories naturally present in the data itself (not by a prepicked list).
The GovData Project: Principles

4. Support “dynamically-grained natural language search”.

Subslices of the data, defined by categories naturally present in the data itself (not by a prepicked list).
The GovData Project: Principles

4. Support “dynamically-grained natural language search”.

Subslices of the data, defined by categories naturally present in the data itself (not by a prepicked list).

Alaska:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cash receipts: livestock and products</td>
<td>584204</td>
<td>593339</td>
<td>665596</td>
<td>663990</td>
<td>993087</td>
<td>735002</td>
<td>920310</td>
<td>1081209</td>
<td>1022410</td>
<td>1288221</td>
</tr>
<tr>
<td>Meat animals and other livestock</td>
<td>249888</td>
<td>273189</td>
<td>271946</td>
<td>356076</td>
<td>456718</td>
<td>240405</td>
<td>334753</td>
<td>500605</td>
<td>430762</td>
<td>618807</td>
</tr>
<tr>
<td>Cattle and calves</td>
<td>165288</td>
<td>194263</td>
<td>193755</td>
<td>249717</td>
<td>308104</td>
<td>126254</td>
<td>183942</td>
<td>377656</td>
<td>318871</td>
<td>483535</td>
</tr>
<tr>
<td>Hogs and pigs</td>
<td>80284</td>
<td>77513</td>
<td>72786</td>
<td>94602</td>
<td>136457</td>
<td>109754</td>
<td>144265</td>
<td>116332</td>
<td>104471</td>
<td>126786</td>
</tr>
<tr>
<td>Sheep and other livestock</td>
<td>1435</td>
<td>1413</td>
<td>6405</td>
<td>5865</td>
<td>6157</td>
<td>4387</td>
<td>6546</td>
<td>8617</td>
<td>7420</td>
<td>8386</td>
</tr>
<tr>
<td>Dairy products</td>
<td>50340</td>
<td>62844</td>
<td>65976</td>
<td>57268</td>
<td>64289</td>
<td>68755</td>
<td>69576</td>
<td>72226</td>
<td>72633</td>
<td>76061</td>
</tr>
<tr>
<td>Poultry and poultry products</td>
<td>286876</td>
<td>267060</td>
<td>257675</td>
<td>276656</td>
<td>478080</td>
<td>425341</td>
<td>515982</td>
<td>508378</td>
<td>519021</td>
<td>594353</td>
</tr>
</tbody>
</table>

Iowa:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Meat animals and other livestock</td>
<td>2539909</td>
<td>2815470</td>
<td>2638250</td>
<td>3244460</td>
<td>4118971</td>
<td>3649324</td>
<td>3199856</td>
<td>3887302</td>
<td>4000244</td>
<td>522000</td>
</tr>
<tr>
<td>Cattle and calves</td>
<td>1501833</td>
<td>1593967</td>
<td>1573583</td>
<td>1846209</td>
<td>2181732</td>
<td>1959594</td>
<td>1700696</td>
<td>1822640</td>
<td>1932224</td>
<td>27612</td>
</tr>
<tr>
<td>Hogs and pigs</td>
<td>1306771</td>
<td>1190999</td>
<td>1039430</td>
<td>1370481</td>
<td>1906248</td>
<td>1763548</td>
<td>2073748</td>
<td>2029673</td>
<td>2115929</td>
<td>24318</td>
</tr>
<tr>
<td>Sheep and other livestock</td>
<td>31365</td>
<td>30504</td>
<td>24812</td>
<td>27775</td>
<td>30801</td>
<td>26586</td>
<td>25224</td>
<td>25517</td>
<td>27071</td>
<td>27031</td>
</tr>
<tr>
<td>Dairy products</td>
<td>218287</td>
<td>220719</td>
<td>224296</td>
<td>224901</td>
<td>262188</td>
<td>290224</td>
<td>304224</td>
<td>344953</td>
<td>353484</td>
<td>39547</td>
</tr>
<tr>
<td>Poultry and poultry products</td>
<td>94950</td>
<td>96803</td>
<td>81299</td>
<td>91461</td>
<td>159093</td>
<td>122317</td>
<td>125152</td>
<td>132532</td>
<td>121513</td>
<td>13629</td>
</tr>
</tbody>
</table>
4. Support “dynamically-grained natural language search”.

Subslices of the data, defined by categories naturally present in the data itself (not by a prepicked list).

Without having to use a “menu-driven” DB-specific interface.
The GovData Project: Principles

4. Support “dynamically-grained natural language search”.

Without having to use a “menu-driven” DB-specific interface.
The GovData Project: Principles

4. Support “dynamically-grained natural language search”.

Without having to use a “menu-driven” DB-specific interface.
The GovData Project: Principles


---

Financial and Operating Data


Step 6. Select Sectors or Industry Groups

- All Sectors or Industry Groups
- All Industries Total
- Finance (except depository institutions) and insurance
- Information
- Manufacturing

- OR -

Select Subsectors or Individual Industries

- Accommodation (Accommodation and food services)
- Accommodation and food services (Other industries)
- Administration, support, and waste management (Other industries)
- Agriculture (--- All ---)
- Agriculture, construction, and mining machinery (Machinery)
- Agriculture, forestry, fishing, and hunting (Other industries)

To select multiple items, hold down the CTRL (Control) Key and click on each item.

Display Selected Data

Clear Selections

- Value Added (Gross Product)
- Property, Plant, and Equipment Expenditures
- Total Sales
The GovData Project: Principles

5. **Unified** rich, high-throughput APIs for acquisition and search.

http://bea.gov/international/ii_web/timeseries2.cfm?econtypeid=2&dirlevel1id=1&Entitytypeid=1&stepnum=1

http://bea.gov/national/nipaweb/TableView.asp?SelectedTable=10&Freq=Qtr&FirstYear=1995&LastYear=1997
The GovData Project: Principles

5. **Unified** rich, high-throughput **APIs** for acquisition and search.

Should serve up something uniform:

- XML
- JSON
- text CSVs

... doesn’t matter which (tho JSON is best). But must be CONSISTENT and INDEPENDENT of application.
The GovData Project: Principles

5. Unified rich, high-throughput APIs for acquisition and search.

The GovData Project: Principles

5. **Unified** rich, high-throughput APIs for acquisition and search.

Search API → Acquisition API

- http://bea.gov/international/ii_web/timeseries2.cfm?econtypeid=2&dirlevel1id=1&Entitytypeid=1&stepnum=1
- http://bea.gov/national/nipaweb/TableView.asp?SelectedTable=10&Freq=Qtr&FirstYear=1995&LastYear=1997


Search -> Acquisition should be “canonical mapping.”
The GovData Project: Principles

5. Unified rich, high-throughput APIs for acquisition and search.

Seamless service for any size dataset:
The GovData Project: Principles

5. Unified rich, high-throughput APIs for acquisition and search.

API should support more than just acquisition:

- count records (e.g. there’s no records for year=1970&industry=webservers) but 12000 for GDP

- get distinct values for a column

- sort

- skip ahead / limit number of sent results

- get all records for which a given column exists (e.g. “Location”)
6. Use lightweight, well-structured, pure-javascript frontends.

So that front-end development and back-end development are totally modularized

(... I’ll come back to the meaning of this. )
The GovData Project: Principles

6. Use lightweight, well-structured, pure-javascript frontends.

So that front-end development and back-end development are totally modularized

(... we’ll circle back to the meaning of this.)
7. Make data operations transparent, reproducible, and versioned.

So that you never “overwrite” your data

So that everyone can see where the cleaned data came from, how the cleaning happened, and how to modify it if they want to.

So that you can process data in an incremental fashion, not having re-do everything each time the new raw data versions are released.
The GovData Project: Principles

8. Be **completely** open source.

So that third parties can reproduce whatever elements that they want, anywhere in the stack.

This includes the database technologies in the backend. (“We’re totally open-source. ... and the database is Oracle.”)

... recall, Open Data Initiative NEVER intends to have centralized storage /API &c.

Q: But doesn’t this mean that, instead of each bureau having to do its thing, we have to do? Won’t that be a lot of work? I mean, you’ll have to write a separate parser for EACH agency to get the kind of quality you’re talking about.

A: Yes. But see this as the advantage and mission, as opposed to a huge burden.
The GovData Project

10. Open, but not TOO open ...
The GovData Project

10. Open, but not TOO open ...

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. 2010 Report to Congress on White House Staff</td>
<td>Government. Whitehouse, salary, government, congress, white</td>
</tr>
<tr>
<td>3. The White House - Nominations &amp; Appointments</td>
<td>Government</td>
</tr>
<tr>
<td>4. Le Sarkomètre</td>
<td>Government. sarkozy, promesses, campagne, elysée. 113 promesses du candidat Nicolas Sarkozy passées au crible par le Nouvel Obs</td>
</tr>
<tr>
<td>5. 2009 Report to Congress on White House Staff</td>
<td>Government</td>
</tr>
<tr>
<td>6. Unclaimed bank accounts</td>
<td>Government. edmonton, edmonton journal, alberta, money, bank, bank accounts, abandoned. A list of all the abandoned bank accounts at branches in the Edmonton area and or registered to addresses in</td>
</tr>
<tr>
<td>7. Mayor</td>
<td></td>
</tr>
<tr>
<td>8. Seattle Tech Companies Map</td>
<td>Business</td>
</tr>
<tr>
<td>9. Top 1,000 Songs To Hear Before You Die</td>
<td>Fun. songs, music, top. A list of the Guardian's &quot;Top 1,000 Songs to Hear Before You Die&quot;</td>
</tr>
<tr>
<td>10. Interagency Ocean Policy Taskforce Comments</td>
<td>Government. interagency ocean policy taskforce comments</td>
</tr>
</tbody>
</table>
The GovData Project

10. Open, but not TOO open ...
The GovData Project

10. Open, but not TOO open ...
10. Open, but not TOO open ...
Module 3. Data Wrangling Techniques

(needed background: A year of general programming, Python familiarity STRONGLY recommended)
Data Wrangling Outline

Three easy pieces:

1. Scraping & Parsing Tools & Techniques
2. Data “Cleaning” Tools & Techniques
3. A bit of database technology (MongoDB primer)
Data Wrangling Outline

Motivations:

1. Scraping & Parsing Tools & Techniques
   because the web, especially complex data portals, contains lots of data

2. Data “Cleaning” Tools & Techniques
   because the data, even coming from DB-backed sites, is often “dirty”

3. A bit of database technology (MongoDB primer)
   because you want to be able to serve up the data too
Data Wrangling Outline

Three easy pieces:

**prelude:** *How the web works: request / response*

1. Scraping

**interlude:** *How the web works: HTML*

... and then Parsing Tools & Techniques

2. Data “Cleaning” Tools & Techniques

3. A bit of database technology (MongoDB primer)
How the Web Works (sort of)

**SERVER**

A (powerful) computer that hosts a webpage

**CLIENT**

Your computer, where you view the webpage
How the Web Works (sort of)
How the Web Works (sort of)

(the web)
How the Web Works (sort of)
How the Web Works (sort of)

(the web)
How the Web Works (sort of)
How the Web Works (sort of)

http://thissite.com/thispage
How the Web Works (sort of)
How the Web Works (sort of)

Key fact: all the servers know all the other servers’ address, and know how to forward on the message properly
How the Web Works (sort of)

http://thissite.com/thispage

GET me: thissite.com/thispage
How the Web Works (sort of)
How the Web Works (sort of)
Simple web page = static = little computation
Complex page = dynamic (e.g. DB-backed) = more computation

Server computes the response.

thissite.com/thispage.html

like function input
How the Web Works (sort of)
How the Web Works (sort of)

A typical request/response pair looks like:

Request

Response header

Response

Request Headers

Response Headers

Query String Parameters

Headers | Content
---|---
Request URL: https://mail.google.com/mail/?shva=1#inbox
Request Method: GET
Status Code: 200 OK

Accept: application/xml, application/xhtml+xml, text/html; q=0.9, text/plain; q=0.8,
Accept-Charset: ISO-8859-1, utf-8; q=0.7, *; q=0.3
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US, en;q=0.8
Cookie: TZ=380; GMAIL_AT=AF6bupOGIPZTdz3d8vNq3J8v6Y6RBx7Zrxw; gmusicat=dyamins
-TI1n85drl_LzPcRe8GVCUTQ49FxbmUHiVAMW0tH6a8hQCM_79RLHqF0YmHkvrpm0LAgE2xavb3
haZMr-Y; PREF-ID=9f6346c819c4cfab:u-a9958d69946b47ba:FF-4:LD=en;NR=10;TM=1294
cmd=(none); __utmc=29003808.487958594.1294291932.1294281932.1294281932.1;
__ut
VDX-WC38w070438h9x1VnLwWw8F0d1ZcBq6ESDp6VwCNP3DUnEqwVsSsz-1npTIWesYyy2u5w180gM
vsEYOX6V6yMe3nkJXC201ZtEouBreHos0ReVgnaBgH7ms7TuthPZ0565Kr14kfkiAUXVZiCqXm
-A(9:mALydg[1;jj9raPA[5:dZ5kBg[8:nhGRS0[7:FH4tB0[3:585GBA[0:Abmx1Q[2; ;SID=DQA
bob_jgaQ76y0INn6yK94wVemnxmR8IN9jS8CT4oJgAc-2Ro9rgp-bzZndqkV77bAMG1ngT6Ub]L7
User-Agent Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US) AppleWebKitK
How the Web Works (sort of)

The real contents of the response.

It’s HTML.
Your web browser renders HTML into something meaningful.
Data Wrangling Outline Revisited

Three easy pieces:

1. Scraping & Parsing Tools & Techniques
   
   issue the right requests / transform resulting HTML into a data structure
   more suited to analytical manipulation

2. Data “Cleaning” Tools & Techniques
   
   correct and enrich the data structure

3. A bit of database technology (MongoDB primer)
   
   repackage the data structure
   make it available to others just it was made available to you (but better)
Scraping

The Idea Of Scraping:

Issue a GET request *not* through the browser, but instead through some other route, so as to be able to direct the response to your analyze its contents & extract its structured information (as opposed to having it rendered in the browser window).

Sub-issues of Scraping:

- How to issue the request
- How to figure out *which* requests to issue in the first place
- How to extract (that is, parse) data from the response into a useable data structure.
Scraping

How to issue the request:

Command Line Tools
- `wget`
- `curl`
- `python urllib, urllib2`
- `mechanize`
- `selenium`

GUI Web scrapers

More like programming

More like browsing.
Scraping: `wget` / `curl`

You can integrate it into your python scripts trivially:

```python
In [1]: import os

In [2]: os.system("wget http://www.nytimes.com")
```
Scraping: `wget` / `curl`
Scraping: wget / curl
Scraping: `wget / curl`

`wget` has all kinds of options, for recursively getting many subpages of a page, following links in different ways, using passwords with secure pages, controlling how input is named, configuring response headers, &c

`curl` is basically the same
Scraping: `wget / curl`

NYTimes was a pretty simple example. Some are harder.
Scraping: `wget / curl`

NYTimes was a pretty simple example. Some are harder.

The resulting page just doesn’t have the stuff in it.

But it had to have gotten to your computer *somehow*.

Time for Firebug.
Scraping: `wget` / `curl`

NYTimes was a pretty simple example. Some are harder.
Scraping: **wget / curl**

NYTimes was a pretty simple example. Some are harder.
Scraping: `wget` / `curl`

NYTimes was a pretty simple example. Some are harder.
Scraping: **wget / curl**

NYTimes was a pretty simple example. Some are harder.

```bash
wget
--post-data="{"searchQueryString":"p+1-n+12-c+287458-s+5-r+101323338-t++ri++ni+1-x+","isSearchMode":false}"
--no-cache
--max-redirect=0
--header="Content-Type: application/json"
http://www.tiffany.com/Shopping/CategoryBrowse.aspx/GetCategoriesXmlBySearchQS
-O "test.xml"
```

http://www.tiffany.com/shared/media/products/26598044_L_over_M_3.jpg
## Scraping: `wget / curl`

<table>
<thead>
<tr>
<th>Categories</th>
<th>Level2</th>
<th>Page</th>
<th>Image</th>
<th>Item</th>
<th>Price</th>
<th>sku</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Metro five-row ring of diamonds in 18k white gold.</td>
<td>$3,750</td>
<td>GRP03613</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Locks heart pendant in sterling silver, mini.</td>
<td>$100</td>
<td>26914973</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Return to Tiffany™ mini heart tag in sterling silver on a bead bracelet.</td>
<td>$125</td>
<td>GRP03577</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Return to Tiffany™ double heart tag pendant in silver with Tiffany Blue®.</td>
<td>$100</td>
<td>27125107</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Locks vintage charm in sterling silver and 18k rose gold on a chain.</td>
<td>$225</td>
<td>GRP03791</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Locks heart pendant in 18k rose gold, mini.</td>
<td>$300</td>
<td>26914663</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Petals cuff in sterling silver and 18k rose gold with diamonds, medium.</td>
<td>$1,700</td>
<td>GRP03892</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Paloma Picasso® Love &amp; Kisses ring with diamonds in 18k white gold, narrow.</td>
<td>$1,250</td>
<td>GRP03859</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Keys twist heart key pendant in 18k gold on a chain.</td>
<td>$750</td>
<td>GRP03737</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Keys heart key charm with enamel finish in sterling silver on a chain.</td>
<td>$175</td>
<td>GRP03674</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Return to Tiffany™ heart lock, peace tag and Tiffany 1837™ lock bead bracelets.</td>
<td>$375</td>
<td>GRP03766</td>
</tr>
<tr>
<td>Jewelry</td>
<td>New Jewelry</td>
<td>1</td>
<td><img src="image" alt="Image" /></td>
<td>Tiffany Locks vintage charm in sterling silver on a chain.</td>
<td>$130</td>
<td>GRP03790</td>
</tr>
</tbody>
</table>
### Scraping: `wget` / `curl`

<table>
<thead>
<tr>
<th>Jewelry</th>
<th>$500 &amp; Under</th>
<th>1</th>
<th>Paloma Picasso® Love pendant in 18k rose gold, mini.</th>
<th>$500</th>
<th>2629097</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jewelry</td>
<td>$500 &amp; Under</td>
<td>1</td>
<td>Tiffany Sparklers prehnite earrings in sterling silver.</td>
<td>$500</td>
<td>23259478</td>
</tr>
</tbody>
</table>

---

**GovData MongoDB Metadata**

- `wget` and `curl` commands for scraping.

---

**Jewelry Items**

- **Paloma Picasso® Love Pendant**: $500, 2629097
- **Tiffany Sparklers Prehnite Earrings**: $500, 23259478
Scraping: **wget / curl**

Python’s `urllib` and `urllib2` are pure-python libraries for doing similar things.

```python
In [1]: import urllib
In [2]: fh=urllib.urlopen("http://www.nytimes.com")
In [3]: html = fh.read()
In [4]: html[:100]
Out[4]: "<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "$""
In [5]:
```
Scraping: `wget` / `curl`

So let’s look e.g. at BEA. Navigate our way thru to:

What happens when we click on “Download All Years”. Something downloads. From where?
Scraping: wget / curl

Let’s look at the source:
Scraping: `wget / curl`

All right, let’s try it:

```bash
```

Stuck on this ... Go to Firebug and look for something fancy? NO!
Scraping: `wget / curl`

We forgot you use “”s.
### Table 1.1.1. Percent Change From Preceding Period in Real Gross Domestic Product

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.6</td>
<td>-0.3</td>
<td>6.2</td>
<td>6.5</td>
<td>7.6</td>
<td>2.2</td>
<td>0.6</td>
<td>-5.4</td>
<td>-1.4</td>
<td>4.6</td>
<td>-3.7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6.8</td>
<td>1.3</td>
<td>0.1</td>
<td>2.0</td>
<td>4.7</td>
<td>0.6</td>
<td>0.6</td>
<td>6.3</td>
<td>0.9</td>
<td>6.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7.4</td>
<td>2.7</td>
<td>1.5</td>
<td>0.6</td>
<td>3.7</td>
<td>-0.4</td>
<td>2.7</td>
<td>-0.6</td>
<td>9.1</td>
<td>2.0</td>
<td>9.2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8.6</td>
<td>5.7</td>
<td>28.7</td>
<td>-2.1</td>
<td>2.8</td>
<td>6.9</td>
<td>-5.5</td>
<td>-7.1</td>
<td>39.5</td>
<td>23.5</td>
<td>16.9</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7.0</td>
<td>1.9</td>
<td>-5.1</td>
<td>1.3</td>
<td>4.0</td>
<td>-2.4</td>
<td>5.3</td>
<td>1.5</td>
<td>1.1</td>
<td>-3.5</td>
<td>6.7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5.9</td>
<td>-0.9</td>
<td>-2.0</td>
<td>4.3</td>
<td>6.4</td>
<td>2.2</td>
<td>4.1</td>
<td>2.6</td>
<td>2.1</td>
<td>-3.8</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-28.5</td>
<td>-12.2</td>
<td>112.3</td>
<td>50.9</td>
<td>29.1</td>
<td>4.4</td>
<td>-18.5</td>
<td>-45.1</td>
<td>-45.3</td>
<td>39.9</td>
<td>-20.1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-10.2</td>
<td>14.3</td>
<td>40.1</td>
<td>14.2</td>
<td>-0.4</td>
<td>-4.2</td>
<td>-4.6</td>
<td>-21.0</td>
<td>-12.1</td>
<td>-1.0</td>
<td>15.1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>-6.2</td>
<td>-7.7</td>
<td>13.5</td>
<td>27.2</td>
<td>-13.2</td>
<td>1.0</td>
<td>10.7</td>
<td>-17.6</td>
<td>-14.9</td>
<td>-13.4</td>
<td>-6.7</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-2.1</td>
<td>2.7</td>
<td>-5.6</td>
<td>8.6</td>
<td>11.6</td>
<td>12.5</td>
<td>-7.3</td>
<td>-9.0</td>
<td>-19.6</td>
<td>-10.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>-8.2</td>
<td>-12.8</td>
<td>25.4</td>
<td>38.2</td>
<td>-24.0</td>
<td>-5.0</td>
<td>13.5</td>
<td>-22.9</td>
<td>-18.1</td>
<td>-19.4</td>
<td>-4.3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>18.5</td>
<td>7.9</td>
<td>102.7</td>
<td>-4.7</td>
<td>25.5</td>
<td>-12.2</td>
<td>-26.5</td>
<td>-26.8</td>
<td>-6.8</td>
<td>39.9</td>
<td>59.0</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>-1.9</td>
<td>-18.8</td>
<td>-38.2</td>
<td>-17.0</td>
<td>-32.5</td>
<td>9.3</td>
<td>-13.4</td>
<td>57.2</td>
<td>-5.4</td>
<td>-29.4</td>
<td>-43.9</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>-10.7</td>
<td>-20.3</td>
<td>-41.0</td>
<td>-15.6</td>
<td>-34.5</td>
<td>10.7</td>
<td>-15.0</td>
<td>64.0</td>
<td>-6.6</td>
<td>-29.3</td>
<td>-44.6</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>38.4</td>
<td>-8.5</td>
<td>-17.1</td>
<td>-25.5</td>
<td>-18.7</td>
<td>4.0</td>
<td>-3.5</td>
<td>25.4</td>
<td>10.2</td>
<td>-35.3</td>
<td>-40.4</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8.0</td>
<td>-38.2</td>
<td>23.5</td>
<td>61.7</td>
<td>16.8</td>
<td>22.6</td>
<td>-6.3</td>
<td>-5.7</td>
<td>-5.7</td>
<td>-15.4</td>
<td>6.1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>9.5</td>
<td>-43.9</td>
<td>5.5</td>
<td>86.4</td>
<td>16.7</td>
<td>8.8</td>
<td>-1.3</td>
<td>-10.3</td>
<td>-12.3</td>
<td>-13.6</td>
<td>13.8</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>3.4</td>
<td>-14.8</td>
<td>97.8</td>
<td>5.7</td>
<td>17.4</td>
<td>73.9</td>
<td>-19.5</td>
<td>-15.1</td>
<td>15.7</td>
<td>-25.7</td>
<td>-13.4</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2.8</td>
<td>3.7</td>
<td>-5.8</td>
<td>8.2</td>
<td>20.7</td>
<td>8.0</td>
<td>17.9</td>
<td>11.9</td>
<td>16.4</td>
<td>4.1</td>
<td>-4.2</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.8</td>
<td>-13.2</td>
<td>14.3</td>
<td>84.7</td>
<td>8.4</td>
<td>23.5</td>
<td>8.3</td>
<td>15.9</td>
<td>-4.3</td>
<td>-18.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>-23.7</td>
<td>22.3</td>
<td>5.1</td>
<td>-1.1</td>
<td>-3.3</td>
<td>-5.9</td>
<td>21.5</td>
<td>11.0</td>
<td>7.5</td>
<td>-1.1</td>
<td>-22.9</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>231.8</td>
<td>-50.0</td>
<td>-56.7</td>
<td>106.4</td>
<td>222.7</td>
<td>62.9</td>
<td>29.1</td>
<td>1.6</td>
<td>40.9</td>
<td>-11.7</td>
<td>-5.7</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>5.5</td>
<td>9.7</td>
<td>6.6</td>
<td>-0.6</td>
<td>12.0</td>
<td>7.3</td>
<td>10.0</td>
<td>17.6</td>
<td>24.8</td>
<td>17.7</td>
<td>7.9</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>5.6</td>
<td>7.0</td>
<td>17.1</td>
<td>9.7</td>
<td>11.1</td>
<td>10.1</td>
<td>1.9</td>
<td>-7.3</td>
<td>-5.4</td>
<td>2.2</td>
<td>-3.6</td>
<td></td>
</tr>
</tbody>
</table>
Scraping: `wget` / `curl`

How to figure out which requests to get in the first place?

<-- Scrape this page first.
Scraping: `wget / curl`

How to figure out *which* requests to get in the first place?

`<-- Scrape this page first.`

The key point is that there’s a simple discoverable correspondence between the data on this page, and the urls of the actual data files.
How to figure out *which* requests to get in the first place?

<-- Scrape *this* page first.

In fact, there’s more to be gleaned as well, like metadata, stuff that may not be obvious in the actual data file.
Scraping: `wget / curl`

So your parser program goes something like this:

1) First `wget` and parse the “top page” (I’ll get to how you parse it later.)

2) Use the results of 1) to `wget` a whole bunch more pages and then clean those.

We’ll get to you how you deal with the parsing step in a bit ...
Scraping: mechanize

First, let’s try another one!

U.S. direct investment abroad
- Balance of payments and Direct Investment position data (Selected Tables, Interactive Tables)
- Financial and operating data (Selected Tables, Interactive Tables)
- Product Guide for U.S. Direct Investment Abroad

Foreign direct investment in the U.S.
- Balance of payments and Direct Investment position data (Selected Tables, Interactive Tables)
- Financial and operating data (Selected Tables, Interactive Tables)
- Product Guide for Foreign Direct Investment in the U.S.

Articles and presentations
- U.S. Direct Investment Abroad
- Foreign Direct Investment in the U.S.
- Globalization

From:
http://bea.gov/international/index.htm

E.g. the same portal, different data set.
Scraping: mechanize

Here, the first screen you encounter is this:

And you haven’t got a “list.” but notice the URL:

http://bea.gov/international/ii_web/timeseries2.cfm?econtypeid=1&dirlevel1id=1&Entitytypeid=1&stepnum=1
Our choices are being recorded by the data set, along the way:
Scraping: mechanize

Eventually we get to:

Looking at the source and watching firebug, we see the url of interest is

http://bea.gov/international/ii_web/timeseries7-2.cfm
Scraping: mechanize

And sure enough, if we wget this url and send the right data along with this URL, we get the final page.

But there’s a problem. *How to figure out the right values of these things to send, to get a comprehensive dataset?*

Well, one could try parsing down the hierarchy. But this is quite annoying, because the hierarchy has different numbers of levels in different branches.

Enter mechanize.
Scraping: mechanize

Mechanize is a python library that allows you to (sort of) load a page as if you browsing it, find and “click” on buttons in forms, etc... (It’s like a browser w/o JS interpreter.)

So we can “browse” this page programmatically.
Scraping: mechanize

Load up a browser:

```python
In [1]: from mechanize import Browser

In [2]: br = Browser()

In [3]: br.open("http://bea.gov/international/ii\nd=1&dirlevelid=1&Entitytypeid=1&stepnum=1")
Out[3]: <response_seek_wrapper at 0x10263a368 whose_response at 0x102639dd0 whose_fp = <socket._file

```
Scraping: mechanize

See what forms it finds:

```
In [8]: print(list(br.forms()))
[<mechanize._form.HTMLForm instance at 0x10263ed40>, <mechanize._form.HTMLForm instance at 0x10263ee18>]
```

Select that form; see it’s the right one:

```
In [10]: br.select_form(nr=1)

In [11]: print(br)
<Browser visiting http://bea.gov/international/ii_web/timeseries2.cfm?ecoid=1&dirlevelid=1&Entitytypeid=1&stepnum=1
   selected form:
   <step3 POST http://bea.gov/international/ii_web/timeseries3.cfm multipart-data
   <RadioControl(seriesid=[30, 31, 32, 25, 33, 27, 34, 39, 35, 40])>
   <HiddenControl(ecotypeid=1) (readonly)>  
   <HiddenControl(dirlevelid=1) (readonly)>  
   <HiddenControl(entitytypeid=1) (readonly)>  
   <HiddenControl(stepnum=2) (readonly)>
```
Scraping: mechanize

Select the control for the form:

```
In [12]: control = br.find_control(predicate=lambda x: x.name == 'seriesid' and not x.readonly)

In [13]: control
Out[13]: <mechanize._form.RadioControl instance at 0x10263ee60>

In [14]: print(control)
<RadioControl(seriesid=[30, 31, 32, 25, 33, 27, 34, 39, 35, 40])>
```
Scraping: mechanize

Here’s where all the control data is buried:

```python
In [15]: control.items
Out[15]:
[<Item name='30' id='seriesid30' value='30' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid30'>,
 <Item name='31' id='seriesid31' value='31' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid31'>,
 <Item name='32' id='seriesid32' value='32' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid32'>,
 <Item name='25' id='seriesid25' value='25' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid25'>,
 <Item name='33' id='seriesid33' value='33' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid33'>,
 <Item name='27' id='seriesid27' value='27' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid27'>,
 <Item name='34' id='seriesid34' value='34' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid34'>,
 <Item name='39' id='seriesid39' value='39' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid39'>,
 <Item name='35' id='seriesid35' value='35' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid35'>,
 <Item name='40' id='seriesid40' value='40' type='radio' name='seriesid' ck='document.step3.submit()' id='seriesid40'>]
```
Scraping: mechanize

Select the value:

```python
In [16]: br['seriesid'] = [control.items[0].attrs['value']]
In [17]: br['seriesid']
Out[17]: ['30']
```

“Click” the button:

```python
In [18]: br.submit()
Out[18]: <response_seek_wrapper at 0x10263a320 whose wrapped object = <closeable_response at 0x1026c9128 whose fp = <socket._fileobject object at 0x1026b300>>
```

Capture the result:

```python
In [36]: resp = br.response().read()
In [37]: open
KeyboardInterrupt
In [37]: F = open('test.html','w')
F.
In [38]: F.write(resp)
In [39]: F.close()
```
Scraping: mechanize

OK!

- Home
- A-Z Index

Home > International Economic Accounts > Operations of Multinational Companies > Balance of Payments and Direct Investment Position

Balance of Payments and Direct Investment Position Data

U.S. Direct Investment Abroad, U.S. Direct Investment Position Abroad on a Historical

Step 2. Choose Classification:
- Aggregate Totals Only
- By Industry of Affiliate Only (All Industries)
- By Country Only (Major Countries)
- By Country Only (All Countries)
- By Country and Industry

Continue

Back

Repeat.
Scraping: selenium

Sometimes you have to execute the site’s Javascript code to figure out what all the URLs you need are.

0. Watch: see the magic.

1. Begin: write and run tests in Firefox.
   Selenium IDE is a Firefox add-on that records clicks, typing, and other actions to make a test, which you can play back in the browser.
   Download Selenium IDE  Learn more  Screenshots

2. Customize: your language, your browser.
   Selenium Remote Control (RC) runs your tests in multiple browsers and platforms. Tweak your tests in your preferred language.
   Download Selenium RC  Learn more

3. Deploy: scale out, speed up.
   Selenium Grid extends Selenium RC to distribute your tests across multiple servers, saving you time by enabling tests to run in parallel.
Scraping

Sub-issues of Scraping:

- How to issue the request
  -- watch in the browser debug mode
  -- then wget (curl /urllib)

- How to figure out which requests to issue in the first place
  -- parse the tree page hierarchy to get metadata and links
  then wget in a loop

  -- or, write a mechanize spider

- Now: How to extract (that is, parse) data from the response into a useable data structure.
Scraping: Parsing

How to extract (that is, parse) data from the response into a useable data structure.

It depends on the kind of response. If it’s HTML:

- BeautifulSoup
- lxml
- python’s native xml lib
- pyquery

If it’s excel: xlrd

If it’s csv: well, then you’re sort of done with the “parsing” phase and read for cleaning.
```
<html>
  <head>
  </head>
  <body>
    <div class="box" id="main">
      <p>Some text</p>
      <a href="http://www.nytimes.com">A link to the NYTimes.</a>
    </div>
    <div class="box" id="footer">
      <p>More text</p>
    </div>
  </body>
</html>
```
How the Web Works: HTML

tags: basic part-types of a page  ex: <a>, <div>, <p> ;
attributes: information about part  ex: class, id
How the Web Works: HTML

Basic syntax of HTML:

```html
<tag attr1="val1" attr2="val2">
  <tag2 attr = "val3">
    Contents
  </tag2>
</tag>
```

- nested structure; one tag object contains many others
- `<tag>` always (should) “close” with `</tag>`
- attributes are put in at the beginning of the tag using `name = “value”` syntax
<html>
  <head>
  </head>
  <body>
    <div class="box" id="main">
      <p>Some text</p>
      <a href="http://www.nytimes.com">A link to the NYTimes.</a>
    </div>
    <div class="box" id="footer">
      <p>More text</p>
    </div>
  </body>
</html>
How the Web Works: HTML

Some basic tag types:

- `<html>` : begin / end the whole page
- `<head>` : the page header
- `<body>` : the page body
- `<div>`, `<span>` : basic “container” types
- `<a>` : a hyper link
- `<p>` : a paragraph, for text
- `<ul>`, `<li>` : a bulleted list / element of a list
- `<img>` : an image
- `<script>` : JS code
How the Web Works: HTML

```html
<html>
  <head>
  </head>
  <body>
    <div class="box" id="main">
      <p>Some text</p>
      <a href="http://www.nytimes.com">A link to the NYTimes.</a>
    </div>
    <div class="box" id="footer">
      <p>More text</p>
    </div>
  </body>
</html>
```
How the Web Works: HTML

Basic attributes:

class : a user-defined type, many instances per page

id : a user-defined name, only one per page

src: for images

href: for hyper links

style: for CSS style
How the Web Works: HTML

```html
<html>
  <head>
  </head>
  <body>
    <div class="box" id="main">
      <p style="color:red; font-size:40px">Some text</p>
      <a href="http://www.nytimes.com">A link to the NYTimes.</a>
    </div>
    <div class="box" id="footer">
      <p>More text</p>
      <img src="http://www.harvard.edu/images/harvard.gif" />
    </div>
  </body>
</html>
```
How the Web Works: HTML

```html
<html>
<head>
<link type="text/css" href="style.css" rel="Stylesheet" />
</head>
<body>

<div class="box" id="main">
  <p>Some text</p>
  <a href="http://www.nytimes.com">A link to the NYTimes.</a>
</div>

<div class="box" id="footer">
  <p>More text</p>
  <img src="http://www.harvard.edu/images/harvard.gif" />
</div>

</body>
</html>

/* style.css */

#main p{
  color : red;
  font-size : 40px;
}
Scraping: BeautifulSoup

http://www.crummy.com/software/BeautifulSoup/

You didn't write that awful page. You're just trying to get some data out of it. Right now, you don't really care what HTML is supposed to look like.

Neither does this parser.

**Beautiful Soup**

"A tremendous boon." -- [Python411 Podcast](http://www.crummy.com/software/BeautifulSoup/)  

[Download | Documentation | Source | What's New | Contributors | Discussion group ]

If Beautiful Soup has saved you a lot of time and money, please share the wealth.

[Donate]
Scraping: BeautifulSoup

Basic operation: selecting objects

```python
In [1]: import BeautifulSoup

In [2]: S = BeautifulSoup.BeautifulSoup(open('index.html'))
```

```python
Soup.findAll("tagname")     -- get 'em all

Soup.findAll("tagname", attr1=val1, attr2=val2)

Soup.findAll("tagname")     -- for class spec
```
Basic operation: selecting objects

Soup.findAll("tagname",classname) --for class spec

```python
In [6]: stories = S.findAll('div','story')
In [7]: len(stories)
Out[7]: 40
```
Scraping: BeautifulSoup

```
In [1]: import BeautifulSoup

In [2]: S = BeautifulSoup.BeautifulSoup(open('index.html'))

In [6]: stories = S.findAll('div','story')

Out[7]: 40

```

```
<nl>
  <span class="timestamp">57 minutes ago</span>
  In [6]: stories = S.findAll('div','story')

  at conflated the religious and political, the populist Iraqi O
  state that he had once derided as a

  Out[7]: 40

  In [14]: ByLines = [story.findAll(True,'byline') for story in stories]

  In [15]: ByLines[0]

  Out[15]:
    [<h6 class="byline">
      By J. DAVID GOODMAN  <span class="timestamp">18 minutes ago</span>
    </h6>]
```
Scraping: BeautifulSoup

In [12]: Summaries = [story.findAll('p','summary') for story in stories]

In [13]: Summaries[0]
Out[13]:

<p class="summary">
The Democratic congresswoman from Arizona was shot in the head at a public event at a grocery store, according to her spokesman and news reports. Several other people were injured. Their conditions remain unclear.</p>

from starflow import utils

In [10]: Contents = [utils.Contents(s[0]).strip() for s in Summaries if s]

In [11]: Contents[0]
Out[11]: 'The Democratic congresswoman from Arizona was shot in the head at a public event at a grocery store, according to her spokesman and news reports. Several other people were injured. Their conditions remain unclear.'
Scraping: lxml

http://codespeak.net/lxml/

lxml is the most feature-rich and easy-to-use library for working with XML and HTML in the Python language.

Introduction

lxml is a Pythonic binding for the libxml2 and libxslt libraries. It is unique in that it combines the speed and feature completeness of these libraries with the simplicity of a native Python API, mostly compatible but superior to the well-known ElementTree API. See the introduction for more information about background and goals. Some common questions are answered in the FAQ.

For commercial consulting and customisations, please contact Stefan Behnel.

This page describes the current stable version of lxml. See also the web page of the development version.

Documentation

The complete lxml documentation is available for download as PDF documentation. The HTML documentation from this web site is part of the normal source download.

- ElementTree:
  - ElementTree API
  - compatibility and differences of lxml.etree
  - benchmark results
- lxml.etree:
Scraping: etree

http://docs.python.org/library/xml.etree.elementtree.html

19.13. xml.etree.ElementTree — The ElementTree XML API

New in version 2.5.

The Element type is a flexible container object, designed to store hierarchical data structures in memory. The type can be described as a cross between a list and a dictionary.

Each element has a number of properties associated with it:

- a tag which is a string identifying what kind of data this element represents (the element type, in other words).
- a number of attributes, stored in a Python dictionary.
- a text string.
- an optional tail string.
- a number of child elements, stored in a Python sequence

BeautifulSoup does XML really well, too.
(BeautifulSoup.BeautifulStoneSoup)
Scraping: pyquery

http://pypi.python.org/pypi/pyquery

You can use the PyQuery class to load an XML document from a string, an XML document, from a file or from an url:

```python
>>> from pyquery import PyQuery as pq
>>> from lxml import etree
>>> import urllib
>>> d = pq("<html></html>")
>>> d = pq(etree.fromstring("<html></html>"))
>>> d = pq(url='http://google.com/')
>>> d = pq(url='http://google.com/', opener=lambda url: urllib.urlopen(url).read())
>>> d = pq(filename=path_to_html_file)
```

Now `d` is like the `$` in jQuery:

```python
>>> d("#hello")
[<p id="hello">hello</p>]
>>> p = d("#hello")
>>> p.html()
'Hello world !'
>>> p.html("you know <a href='http://python.org/'>Python</a> rocks")
[<p id="hello">you know Python rocks</p>]
>>> p.html()
'u'you know <a href="http://python.org/">Python</a> rocks'
Scraping: excel with xlrd

http://pypi.python.org/pypi/xlrd

xlrd 0.7.1

*Library for developers to extract data from Microsoft Excel (tm) spreadsheet files*

Extract data from new and old Excel spreadsheets on any platform. Pure Python (2.1 to 2.6). Strong support for Excel dates. Unicode-aware.

Forum for comments, questions, and bug reports: http://groups.google.com/group/python-excel
Scraping: ScraperWiki

Welcome to ScraperWiki

*Scraper*: a computer program that copies structured information from webpages into a database

*ScraperWiki*: a website where people can write and repair public web scrapers and invent uses for the data

http://scraperwiki.com/
Scraping: ScraperWiki

Basically organizes the various underlying tools:

Scraping

How to retrieve HTML pages from websites.

`scrapewiki.scrape(url, [params=], )`

Downloads a web page, and returns you the HTML. The page will appear in the "sources" pane in the scraper code editor, so you can easily view what was downloaded.

`url`
   The address of the web page, e.g. http://www.parliament.uk/mps/leadersandoffices/government_and_opposition/hmg.cfm

`params (optional)`
   If present, makes the HTTP request a POST request, such as if a form had been submitted. `params` is a dictionary, whose keys are the names of the fields being posted, and values are their values.

Standard libraries

You can download pages using all the standard Python functions. The page will not appear in the "sources" pane.

`urllib2`, `urlparse` Standard python libraries for opening urls. [docs](#)

`BeautifulSoup` For parsing html. [docs](#)

`mechanize` For navigating form fillings. [docs](#)

`x1rd` For reading Excel files. [docs](#)

`Python Google Chart` For generating charts using the Google Chart API. [docs](#)

We’d love to find a way to integrate ....
Scraping: Our `htools`

```python
import utils.htools as htools

def TiffanyCatalogInstantiator(creates = protocolroot + 'Tiffany.py'):
    L = [ParseTiffany1, {'Parser': ParseTiffany2, 'Getter': JSONWgetter}, ParseTiffanyFinal]

def ParseTiffany1(page,x):
    Soup = BeautifulSoup(open(page), convertEntities=BeautifulStoneSoup.HTML_ENTITIES)
    G = [a for a in Soup.findAll('a', 'globalnav') if a.findNext().name == 'ul']
    Recs = []
    for g in G:
        lev1 = Contents(g)
        LI = g.findNext().findAll('li')
        newrecs = [(lev1, Contents(li), str(dict(li.findAll('a')[0].attrs)['href'])) for li in LI if Contents(li).strip('\xc2\xa0')]
        for i in range(len(newrecs)):
            l = newrecs[i]
            if not l[2].startswith('http://'):
                newrecs[i] = (l[0], l[1], 'http://www.tiffany.com' + l[2])
            l = newrecs[i]
            newrecs[i] = (l[0], l[1], l[2].replace('Category.aspx', 'CategoryBrowse.aspx'))
    Recs += newrecs
    return tb.tabarray(records = Recs, names = ['Level1', 'Level2', 'URL'], coloring={Categories:['Level1', 'Level2']})
```
Data Wrangling Outline

Three easy pieces:

1. Scraping & Parsing Tools & Techniques
2. Data “Cleaning” Tools & Techniques
3. A bit of database technology (MongoDB primer)
### Data Wrangling Outline

Even if you have this in CSV format:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.6</td>
<td>-0.3</td>
<td>6.2</td>
<td>6.5</td>
<td>7.6</td>
<td>2.2</td>
<td>0.6</td>
<td>-5.4</td>
<td>-1.4</td>
<td>4.6</td>
<td>-3.7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6.8</td>
<td>1.3</td>
<td>0.1</td>
<td>2.0</td>
<td>4.7</td>
<td>0.6</td>
<td>3.2</td>
<td>0.6</td>
<td>6.3</td>
<td>0.9</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>7.4</td>
<td>2.7</td>
<td>1.5</td>
<td>0.6</td>
<td>3.7</td>
<td>-0.4</td>
<td>2.7</td>
<td>-0.6</td>
<td>9.1</td>
<td>2.0</td>
<td>9.2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8.6</td>
<td>5.7</td>
<td>28.7</td>
<td>-2.1</td>
<td>2.6</td>
<td>6.9</td>
<td>6.5</td>
<td>7.1</td>
<td>39.5</td>
<td>23.5</td>
<td>16.9</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.7</td>
<td>-1.9</td>
<td>-2.1</td>
<td>1.3</td>
<td>4.0</td>
<td>-2.4</td>
<td>5.3</td>
<td>1.5</td>
<td>1.1</td>
<td>3.5</td>
<td>6.7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5.9</td>
<td>0.9</td>
<td>1.5</td>
<td>2.0</td>
<td>3.4</td>
<td>2.2</td>
<td>4.1</td>
<td>2.1</td>
<td>0.8</td>
<td>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-28.5</td>
<td>-12.2</td>
<td>112.3</td>
<td>50.9</td>
<td>29.1</td>
<td>4.4</td>
<td>-18.5</td>
<td>-49.1</td>
<td>-45.3</td>
<td>39.9</td>
<td>-20.1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-10.2</td>
<td>14.3</td>
<td>40.1</td>
<td>14.2</td>
<td>-0.4</td>
<td>-4.2</td>
<td>-4.6</td>
<td>-21.0</td>
<td>-12.2</td>
<td>-1.0</td>
<td>15.1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>-6.2</td>
<td>-7.7</td>
<td>13.5</td>
<td>27.2</td>
<td>-13.2</td>
<td>1.0</td>
<td>10.7</td>
<td>-17.6</td>
<td>-14.9</td>
<td>-19.4</td>
<td>-6.7</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-2.1</td>
<td>2.7</td>
<td>-5.6</td>
<td>8.6</td>
<td>11.6</td>
<td>12.5</td>
<td>5.9</td>
<td>-7.3</td>
<td>-9.0</td>
<td>-15.6</td>
<td>-10.5</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>-8.2</td>
<td>-12.8</td>
<td>25.4</td>
<td>38.2</td>
<td>-24.0</td>
<td>-5.0</td>
<td>13.5</td>
<td>-22.9</td>
<td>-19.4</td>
<td>-4.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-18.5</td>
<td>76.9</td>
<td>102.7</td>
<td>-4.7</td>
<td>25.5</td>
<td>-12.2</td>
<td>-26.8</td>
<td>-6.8</td>
<td>39.9</td>
<td>59.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>-18.5</td>
<td>76.9</td>
<td>102.7</td>
<td>-4.7</td>
<td>25.5</td>
<td>-12.2</td>
<td>-26.8</td>
<td>-6.8</td>
<td>39.9</td>
<td>59.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>-18.5</td>
<td>76.9</td>
<td>102.7</td>
<td>-4.7</td>
<td>25.5</td>
<td>-12.2</td>
<td>-26.8</td>
<td>-6.8</td>
<td>39.9</td>
<td>59.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Even if you have this in CSV format, supposed clean:

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GDP</td>
<td>-0.6</td>
<td>-0.3</td>
<td>6.2</td>
<td>6.5</td>
<td>7.6</td>
<td>2.2</td>
<td>0.5</td>
<td>-5.4</td>
<td></td>
</tr>
<tr>
<td>Goods</td>
<td>7.4</td>
<td>2.7</td>
<td>1.5</td>
<td>0.6</td>
<td>3.7</td>
<td>-0.4</td>
<td>2.7</td>
<td>-0.6</td>
<td>9.1</td>
</tr>
<tr>
<td>Durables</td>
<td>8.6</td>
<td>5.7</td>
<td>28.7</td>
<td>-2.1</td>
<td>2.8</td>
<td>6.9</td>
<td>-5.5</td>
<td>-7.1</td>
<td>39.5</td>
</tr>
<tr>
<td>Nondurables</td>
<td>7.0</td>
<td>1.9</td>
<td>-5.1</td>
<td>1.3</td>
<td>4.0</td>
<td>-2.4</td>
<td>5.3</td>
<td>1.5</td>
<td>1.1</td>
</tr>
<tr>
<td>Services</td>
<td>5.9</td>
<td>-0.9</td>
<td>-2.0</td>
<td>4.3</td>
<td>6.4</td>
<td>2.2</td>
<td>4.1</td>
<td>2.6</td>
<td>-0.8</td>
</tr>
<tr>
<td>Private</td>
<td>-28.5</td>
<td>-12.2</td>
<td>112.3</td>
<td>50.9</td>
<td>29.1</td>
<td>4.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fixed</td>
<td>-10.2</td>
<td>14.3</td>
<td>40.1</td>
<td>14.2</td>
<td>-0.4</td>
<td>-4.2</td>
<td>-4.6</td>
<td>-21.0</td>
<td></td>
</tr>
<tr>
<td>Nonresidential</td>
<td>-6.2</td>
<td>-7.7</td>
<td>13.5</td>
<td>27.2</td>
<td>-13.2</td>
<td>1.0</td>
<td>10.7</td>
<td>-17.6</td>
<td>-20.0</td>
</tr>
<tr>
<td>Structures</td>
<td>-2.1</td>
<td>2.7</td>
<td>-5.6</td>
<td>8.6</td>
<td>11.6</td>
<td>12.5</td>
<td>5.9</td>
<td>-7.3</td>
<td>-9.0</td>
</tr>
<tr>
<td>Equipment and software</td>
<td>-8.2</td>
<td>-12.8</td>
<td>25.4</td>
<td>38.2</td>
<td>-24.0</td>
<td>-5.0</td>
<td>13.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Residential</td>
<td>-18.5</td>
<td>76.9</td>
<td>102.7</td>
<td>-4.7</td>
<td>25.5</td>
<td>-12.2</td>
<td>-26.5</td>
<td>-26.8</td>
<td></td>
</tr>
<tr>
<td>Change in private inventories</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net exports of goods and services</td>
<td>-4.9</td>
<td>-18.8</td>
<td>-38.2</td>
<td>-17.0</td>
<td>-32.5</td>
<td>9.8</td>
<td>-13.4</td>
<td>57.2</td>
<td>-3.4</td>
</tr>
<tr>
<td>Exports</td>
<td>-10.7</td>
<td>-20.3</td>
<td>-41.0</td>
<td>-15.6</td>
<td>-34.5</td>
<td>10.7</td>
<td>-15.0</td>
<td>64.0</td>
<td>-5.6</td>
</tr>
<tr>
<td>Goods</td>
<td>38.4</td>
<td>-8.5</td>
<td>-17.1</td>
<td>-25.5</td>
<td>-18.7</td>
<td>4.0</td>
<td>-3.5</td>
<td>23.4</td>
<td>10.2</td>
</tr>
<tr>
<td>Services</td>
<td>8.0</td>
<td>-38.2</td>
<td>23.5</td>
<td>61.7</td>
<td>16.8</td>
<td>22.6</td>
<td>-6.3</td>
<td>-11.4</td>
<td>-5.7</td>
</tr>
<tr>
<td>Imports</td>
<td>9.5</td>
<td>-43.9</td>
<td>5.5</td>
<td>86.4</td>
<td>16.7</td>
<td>8.8</td>
<td>-1.3</td>
<td>-10.3</td>
<td>-12.3</td>
</tr>
<tr>
<td>Services</td>
<td>3.4</td>
<td>-14.8</td>
<td>97.8</td>
<td>5.7</td>
<td>17.4</td>
<td>73.9</td>
<td>-19.5</td>
<td>-15.1</td>
<td>15.7</td>
</tr>
<tr>
<td>Government consumption expenditures and gross investment</td>
<td>2.6</td>
<td>3.7</td>
<td>-5.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Federal</td>
<td>0.8</td>
<td>0.2</td>
<td>-13.2</td>
<td>14.3</td>
<td>26.7</td>
<td>8.4</td>
<td>23.5</td>
<td>8.3</td>
<td>15.9</td>
</tr>
<tr>
<td>National defense</td>
<td>-23.7</td>
<td>22.3</td>
<td>2.8</td>
<td>-1.1</td>
<td>-3.3</td>
<td>-5.9</td>
<td>21.5</td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td>Nondefense</td>
<td>231.8</td>
<td>-50.0</td>
<td>-56.7</td>
<td>106.4</td>
<td>227.2</td>
<td>62.9</td>
<td>29.1</td>
<td>-16.4</td>
<td></td>
</tr>
<tr>
<td>State and local</td>
<td>5.5</td>
<td>9.7</td>
<td>6.6</td>
<td>-0.6</td>
<td>12.0</td>
<td>7.3</td>
<td>10.0</td>
<td>17.6</td>
<td>24.8</td>
</tr>
<tr>
<td>Addendum:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gross domestic product, current dollars</td>
<td>5.6</td>
<td>7.0</td>
<td>17.1</td>
<td>9.7</td>
<td>11.1</td>
<td>10.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

It still needs cleaning. Let me count the ways.
Data Wrangling Outline

1. Header needs to be removed. (and its data used)

2. Empty column header needs filling in. It’s the “category” column.

3. Column headers refer to dates in a weird way.

4. Line 14 is empty

5. The “category” column is actually several columns collapsed via formatting.
Data Wrangling Outline

You have several options for something like this job

Python itself (esp csv module)

Numpy/Tabular

Google Refine
Data Wrangling: Cleaning

Python itself

Use “/home/NIPA_Table_1.1.1.csv”
(make sure you open it ‘rU’)

```python
In [20]: X = open('/Users/dyamins/Desktop/test.csv','rU').read()
In [21]: Lines = X.split('
')
In [22]: Lines[0]
Out[22]: '"Table 1.1.1. Percent Change From Preceding Period in Real Gross Domestic Product"

In [23]: Lines[5]
```
Data Wrangling: Cleaning

Python itself

```python
In [24]: ColumnHeaders = Lines[5].split(', ')
In [25]: ColumnHeaders[:10]
Out[25]:
```

Notice the error
Data Wrangling: Cleaning

The native CSV module can often help:

```
In [31]: ColumnHeaders[0]
Out[31]: ' Line ','

In [32]: import csv

In [33]: H = csv.reader([[Lines[5]]])

In [34]: ColumnHeaders = H.next()

In [35]: ColumnHeaders[:10]
Out[35]: [' Line ',
' ',
' 1947-II ',
' 1947-III ',
' 1947-IV ',
' 1948-I ',
' 1948-II ',
' 1948-III ',
' 1948-IV ',
' 1949-I ']
```
Data Wrangling: Cleaning

http://entabular.com/tabular/

Tabular

Tabular data can be easily represented in Python using the language’s native objects — e.g. by lists of tuples. Create, these kind of representations typically do not enable important tabular data manipulations, like spreadsheet-style operations.

`Tabular` is a package of Python modules for working with tabular data. Its main object is the `tabarray` class. Data. By putting data into a `tabarray` object, you’ll get a representation of the data that is more flexible and specifically, `tabarray` provides:

- ultra-fast filtering, selection, and numerical analysis methods, using convenient Matlab-style matrix operations,
- spreadsheet-style operations, including row & column operations, ‘sort’, ‘replace’, ‘aggregate’, ‘pivot’
- flexible load and save methods for a variety of file formats, including delimited text (CSV), binary, and
- sophisticated inference algorithms for determining formatting parameters and data types of input files
- support for hierarchical groupings of columns, both as data structures and file formats
Data Wrangling: Cleaning

```python
import tabular as tb

use "/home/TiffanyCatalog.csv"

In [33]: X = tb.tabarray(SVfile = '/Users/dyamins/govdata/Data/catalogs/Tiffany/catalog.tsv')
Setting metadata attribute from dialect delimiter to equal specified value: '
Setting dialect attribute delimiter to equal specified value: '

In [34]: X.dtype
Out[34]: dtype([('Level1', '|S23'), ('Level2', '|S28'), ('Page', '|S2'), ('Image', '|S75'), ('Item', '|S83'), ('Price', '|S10'), ('sku', '|S8'), ('Image2', '|S80'), ('Image3', '|S81'), ('Image4', '|S75')])

Y = X[['Level1', 'Level2', 'Price']]
```
Data Wrangling: Cleaning

```python
In [4]: from starflow import utils
   ...: util.uniqify(X[['Level1']])
Out[4]: ['Jewelry', 'Diamonds', 'Watches', 'Men's', 'Designers & Collections', 'Accessories', 'Gifts', 'Table']
   ...: In [6]: util.uniqify(X[['Level1', 'Level2']].tolist())
```
Data Wrangling: Cleaning

In [16]: Prices = [int(x.strip(')').replace(',', '')) for x in X['Price']]

In [17]: Y1 = Y.addcols(Prices, names=['Price'])
('Replacing columns', ['Price'])

In [18]: Y1
Out[18]:
tabarray([('Jewelry', 'New Jewelry', 3750), ('Jewelry', 'New Jewelry', 100
('Jewelry', 'New Jewelry', 125), ...
('Table', 'Wedding Gifts', 155), ('Table', 'Wedding Gifts', 115),
('Table', 'Wedding Gifts', 330)],
dtype=[('Level1', '<|S23'), ('Level2', '<|S28'), ('Price', '<i8')])
Data Wrangling: Cleaning

```python
import numpy as np

In [19]: AggFuncDict = {}

In [20]: AggFuncDict['Price'] = lambda x: np.array(x).mean()

In [21]: AggFuncDict['Level2'] = lambda x: 'Average'

In [22]: Z = Y1.aggregate(On=['Level1'], AggFuncDict = AggFuncDict)

In [26]: Z
Out[26]:

tabarray([(‘Accessories’, 496.4193548387097, ‘Average’),
          (‘Designers & Collections’, 3726.367137355584, ‘Average’),
          (‘Diamonds’, 3554.987037037037, ‘Average’),
          (‘Gifts’, 1452.8847406664995, ‘Average’),
          (‘Jewelry’, 7793.111458333334, ‘Average’),
          (‘Men’s’, 1573.1723484848485, ‘Average’),
          (‘Table’, 541.8221343873518, ‘Average’),
          (‘Watches’, 11600.310559006211, ‘Average’)],
          dtype=[(‘Level1’, ‘|S23’), (‘Price’, ‘<f8’), (‘Level2’, ‘|S7’)])

Z.sort(order=['Price'])
```
Data Wrangling: Cleaning

http://code.google.com/p/google-refine/
Data Wrangling Outline

Three easy pieces:

1. Scraping & Parsing Tools & Techniques
2. Data “Cleaning” Tools & Techniques
   
   A simple example to pull things together ...

3. A bit of database technology (MongoDB primer)
Data Wrangling Outline

An example to pull things together:

```
cp /home/parse_FEC.py .
cp /home/runFEC.sh .
emacs parse_FEC.py
   execute and understand all the lines with your partner
uncomment "ALL DATA" line and comment "SOME DATA" line, using ‘#’ char to start comment
qsub runFEC.sh
```
Data Wrangling Outline

Three easy pieces:

1. Scraping & Parsing Tools & Techniques
2. Data “Cleaning” Tools & Techniques
3. A bit of database technology (MongoDB primer)
Data Wrangling: Databases

A Database is a program that allows data to be retrieved by topically-meaningful query (as opposed to loading from a file).

The basic “process” you do is:

1. Start your DB program.

2. Insert data (like putting files into a directory, but the file system structure is hidden)

3. Then you get data by:
   INPUT: a query
   OUTPUT: one or more records
Data Wrangling: Databases

One of the nice things about modern databases is that they allow many users to insert and retrieve data simultaneously.
Data Wrangling: Databases

MongoDB is a good modern database program.

MongoDB has three levels in its “schema”:

1) The “database” level; you’re allowed to have multiple databases on your machine.

2) The “collection” level; each collection is a little like a tabular data file. One database can contain many collections.

3) The “document” level; each document is like a record in a CSV. Each collection is composed of many documents.
Data Wrangling: Databases

1) The “database” level; you’re allowed to have multiple databases on your machine.
Data Wrangling: Databases

2) The “collection” level; each collection is a little like a tabular data file. One database can contain many collections.
3) The “document” level; each document is like a record in a CSV. Each collection is composed of many documents.
Data Wrangling: Databases

Let’s do an example.

use ‘testcollection_myname’

```
In [1]: import pymongo
In [2]: connection = pymongo.Connection()
In [3]: connection
Out[3]: Connection('localhost', 27017)
In [4]: database = connection['testdb']
In [5]: collection = database['testcollection']
In [6]: collection
Out[6]: Collection(Database(Connection('localhost', 27017), 'testdb'), 'testcollection')
```

How many records are in our collection?

```
In [7]: collection.count()
Out[7]: 0
```

0, since we just started it.
Data Wrangling: Databases

In [8]: collection.insert({'State': 'Alabama', 'Employment': 22})
Out[8]: ObjectId('4d28edfab90022493b000000')

In [9]: collection.insert({'State': 'Alaska', 'Employment': 3})
Out[9]: ObjectId('4d28ee05b90022493b000001')

In [10]: collection.insert({'State': 'Massachusetts', 'Employment': 54})
Out[10]: ObjectId('4d28ee13b90022493b000002')

In [11]: collection.count()
Out[11]: 3

In [32]: collection.find().distinct('State')
Out[32]: ['u'Alabama', 'u'Alaska', 'u'Massachusetts']

In [12]: collection.find_one()
Out[12]: {u'Employment': 22, u'State': u'Alabama', u'_id': ObjectId('4d28edfab90022493b000000')}
Data Wrangling: Databases

In [15]: collection.insert({'State': 'Massachusetts', 'Employment': 4, 'Year': 2001})
Out[15]: ObjectId('4d28ee96b90022493b000003')

In [16]: collection.insert({'State': 'Massachusetts', 'Employment': 5, 'Year': 2002})
Out[16]: ObjectId('4d28ee9ab90022493b000004')

In [17]: collection.insert({'State': 'Massachusetts', 'Employment': 10, 'Year': 2003})
Out[17]: ObjectId('4d28ee9fb90022493b000005')

In [18]: collection.insert({'State': 'Massachusetts', 'Employment': 3, 'Year': 2004})
Out[18]: ObjectId('4d28eeab90022493b000006')

In [19]: collection.insert({'State': 'Massachusetts', 'Employment': 6, 'Year': 2005})
Out[19]: ObjectId('4d28eeab90022493b000007')

In [20]: collection.insert({'State': 'Massachusetts', 'Employment': 4, 'Year': 2006})
Out[20]: ObjectId('4d28eeb3b90022493b000008')

In [80]: collection.find({'State': 'Massachusetts'}).count()
Out[80]: 7

In [81]: collection.find({'State': 'Massachusetts'}).distinct('Year')
Data Wrangling: Databases

In[21]: collection.find_one({'State': 'Massachusetts'})
Out[21]:
{u'Employment': 54, 
 u'State': u'Massachusetts', 
 u'_id': ObjectId('4d28ee13b90022493b000002')}
Data Wrangling: Databases

In [27]: data = collection.find({'State': 'Massachusetts'})
In [28]: data.next()
Out[28]:
{'u'Employment': 54, 'u'State': 'u'Massachusetts', 'u'_id': ObjectId('4d28ee13b9002493b000002')}
In [29]: data.next()
Out[29]:
{'u'Employment': 4, 'u'State': 'u'Massachusetts', 'u'Year': 2001, 'u'_id': ObjectId('4d28ee96b9002493b000003')}
In [30]: data.next()
Out[30]:
{'u'Employment': 5, 'u'State': 'u'Massachusetts', 'u'Year': 2002, 'u'_id': ObjectId('4d28ee9ab9002493b000004')}

In [26]: collection.find({'State': 'Massachusetts'})
In [26]:
Out[26]:
{'u'Employment': 4, 'u'State': 'u'Massachusetts', 'u'Year': 2001, 'u'_id': ObjectId('4d28ee9ab9002493b000004')}
Data Wrangling: Databases

In [33]: collection.insert({'Location': {'State': 'Massachusetts', 'City': 'Cambridge'}, 'Employment': 20, 'Year': 2010})
Out[33]: ObjectId('4d28f3b3b90022493b000009')

In [34]: collection.insert({'Location': {'State': 'New York', 'City': 'Albany'}, 'Employment': 4, 'Year': 2010})
Out[34]: ObjectId('4d28f3fbb90022493b00000a')

In [35]: collection.find_one({'Location.State': 'Massachusetts'})
Out[35]:
{u'Employment': 20,
 u'Location': {u'City': u'Cambridge', u'State': u'Massachusetts'},
 u'Year': 2010,
 u'_id': ObjectId('4d28f3b3b90022493b000009')}
Data Wrangling: Databases

In [36]: collection.find_one({'Location.City': 'Albany', 'Employment': 22})

Out[37]:
{u'Employment': 4,
 u'Location': {u'City': u'Albany', u'State': u'New York'},
 u'Year': 2010,
 u'_id': ObjectId('4d28f3fbb90022493b00000a')}

In [56]: data = collection.find({'Employment': {$in: [3,4,6]}})

In [57]: data.next()

Out[57]:
{u'Employment': 3,
 u'State': u'Alaska',
 u'_id': ObjectId('4d28ee05b90022493b00001')}

In [58]: data.next()

Out[58]:
{u'Employment': 4,
 u'State': u'Massachusetts',
 u'Year': 2001,
 u'_id': ObjectId('4d28ee96b90022493b00003')}
In [40]: data = collection.find({'$or':[{'Location.City':'Albany'},{'Employment':4}]})

In [41]: data.next()
Out[41]:
{u'Employment': 4,
 u'State': u'Massachusetts',
 u'Year': 2001,
 u'_id': ObjectId('4d28ee96b90022493b000003')}

In [42]: data.next()
Out[42]:
{u'Employment': 4,
 u'State': u'Massachusetts',
 u'Year': 2006,
 u'_id': ObjectId('4d28e3b3b90022493b000008')}

In [43]: data.next()
Out[43]:
{u'Employment': 4,
 u'Location': {u'City': u'Albany', u'State': u'New York'},
 u'Year': 2010,
 u'_id': ObjectId('4d28f3fbb90022493b00000a')}
Data Wrangling: Databases

In [52]: collection.find_one({'Location.City': {'$exists': True}})
Out[52]:
{'Employment': 20,
 'Location': {'City': 'Cambridge', 'State': 'Massachusetts'},
 'Year': 2010,
 'id': ObjectId('4d28f3b3b90022493b000009')}

In [53]: collection.find_one({'Year': {'$exists': False}})
Out[53]:
{'Employment': 22,
 'State': 'Alabama',
 'id': ObjectId('4d28edfab90022493b000000')}
In [61]: data = collection.find().sort('Employment')

In [62]: data.next()
Out[62]:
{u'Employment': 3,
 u'State': u'Alaska',
 u'_id': ObjectId('4d28ee05b90022493b000001')}

In [63]: data.next()
Out[63]:
{u'Employment': 3,
 u'State': u'Massachusetts',
 u'Year': 2004,
 u'_id': ObjectId('4d28ee4b90022493b000006')}

In [64]:

In [65]: data.next()
Out[65]:
{u'Employment': 4,
 u'State': u'Massachusetts',
 u'Year': 2001,
 u'_id': ObjectId('4d28ee96b90022493b000003')}
Data Wrangling: Databases

```python
In [68]: data = collection.find().sort('Employment',direction=-1)
```

```python
In [69]: data.next()
Out[69]:
{u'Employment': 54,
 'State': u'Massachusetts',
 '_id': ObjectId('4d28ee13b90022493b000002')}  
```

```python
In [70]: data.next()
Out[70]:
{u'Employment': 22,
 'State': u'Alabama',
 '_id': ObjectId('4d28edfab90022493b000000')}  
```

```python
In [77]: data = collection.find().sort('Employment').skip(5)
```

```python
In [78]: data.next()
Out[78]:
{u'Employment': 5,
 'State': u'Massachusetts',
 'Year': 2002,
 '_id': ObjectId('4d28ee9ab90022493b000004')}  
```
**Agile and Scalable**

MongoDB (from "humongous") is a scalable, high-performance, open source, document-oriented database. Written in C++, MongoDB features:

- **Document-oriented storage**
  JSON-style documents with dynamic schemas offer simplicity and power.

- **Full Index Support**
  Index on any attribute, just like you're used to.

- **Replication & High Availability**
  Mirror across LANs and WANs for scale and peace of mind.

- **Auto-Sharding**
  Scale horizontally without compromising functionality.

- **Querying**

- **Fast In-Place Updates**
  Atomic modifiers for contention-free performance.

- **Map/Reduce**
  Flexible aggregation and data processing.

- **GridFS**
  Store files of any size without complicating your stack.

**Advanced Queries**

- Introduction
- Retrieving a Subset of Fields
- Conditional Operators
  - `<`, `<=`, `>`, `>=`
  - `$all`
  - `$exists`
  - `$mod`
  - `$ne`
  - `$in`
  - `$nin`
  - `$nor`
  - `$or`
  - `$size`
  - `$type`
- Regular Expressions
- Value in an Array
  - $elemMatch
- Value in an Embedded Object
- Meta operator: $not
- Javascript Expressions and $where
- Cursor Methods
  - `count()`
  - `limit()`
  - `skip()`
  - `snapshot()`
  - `sort()`
- Special operators
  - `group()`
  - See Also
Module 4. GovData Platform

(needed background: A year of general programming, Python familiarity STRONGLY recommended + Module 3)
The GovData Project: Overview

Back-back end → Backend → Frontend
The GovData Project: Overview

Back-back end
- Acquisition
- Cleaning & Parsing
- Insertion
- Versioning
- Search Indexing

Amazon EC2, Python, Starflow

Backend
- Database
- Search Engine
- GIS server
- APIs

MongoDb, Solr, Tornado, Geodjango

Frontend(s)
- Search Page
- Widgets
- Timelines
- Table Vis.
- Javascript
  - jquery, underscore, requireJS, jquery.address, jquery.ui, headJS, processingJS, raphaelJS, canvas, SVG

Dataset browser

StarFlow is an industrial-strength in-house data workflow engine.

We’ve customized an overall framework for getting government data from the web into a MongoDB.

To be populated by hundreds of parser scripts.

Individual parsers job = Cleaning / formatting / time & space
= “Lightly structuring” ....

Uses: **Python,**
wget, curl,
urllib,
BeautifulSoup,
lxml,
elemenTree,
pyquery,
mechanize,
clientform
selenium

spidermonkey,
pyV8,
... and some home-grown utilities

Unfortunately, doesn’t use:
Google Refine

But maybe soon ...

Starflow’s job = managing features common to all parsers ....
   = incremental recomputation multi-step parsers, ensuring metadata format correctness, managing insertion into MongoDB versioning,
   managing search indexing process
   = handling all the workflow things that make recomputable, heavy-duty data parsing really annoying, so that the community can focus on writing good parsers

https://github.com/yamins81/StarFlow
The GovData Project: Data Backend

1. The GovData Mongo Schema
2. Govdata MetaData Schema
3. Govdata Search Engine
4. APIs

- gov.thedata.org/data
- gov.thedata.org/search
- gov.thedata.org/metadata
- gov.thedata.org/geo
GovData Wrangling: The “Schema”

There is one “govdata” database.

Govdata datasets = MongoDB Collections
~ SQL Tables ~ csv file (or a bunch of them)

Dataset rows = MongoDB Documents = records in our case

Column Names = Document Keys, Values = Values

There is a ‘__metadata__’ collection to which metadata is attached for each collection.
GovData Wrangling: The “Schema”

GovDB = connection[‘govdata’]

nipa_tables = GovDB[‘bea_nipa’]

record = nipa_tables.find_one()

{‘Location’:‘Alabama’,  ‘Unemployment’:23145}
GovData Wrangling: The “Schema”

Govdata datasets = MongoDB Collections
Dataset rows = MongoDB Documents
Column Names = Document Keys
Values = Values

Each query produces a slice of a collection, e.g. a set of records.

{Topic: Employment, Element: Rate, 1984_val > 10}

{State: MA, Industry: Arts & Entertainment}
GovData Wrangling: The “Schema”

Unified Time Schema:

```python
TIME_CODE_MAP = [(
    'Y', 'Year', None),
    ('h', 'Half', range(1, 3)),
    ('q', 'Quarter', range(1, 5)),
    ('m', 'Month', range(1, 13)),
    ('d', 'DayOfMonth', range(1, 32)),
    ('U', 'WeekOfYear', range(1, 153)),
    ('w', 'DayOfWeek', range(1, 8)),
    ('j', 'DayOfYear', range(1, 366)),
    ('H', 'HourOfDay', range(1, 24)),
    ('M', 'MinuteOfHour', range(0, 60)),
    ('S', 'Second', range(0, 60)),
    ('Z', 'TimeZone', None)]
```

```python
(TIME_CODES, TIME_DIVISIONS, TIME_RANGES) = zip(*TIME_CODE_MAP)
TIME_DIVISIONS = dict(zip(TIME_CODES, TIME_DIVISIONS))
TIME_RANGES = dict(zip(TIME_CODES, TIME_RANGES))
```

# For each code it gives you a hierarchical list of direct descendents in order

```python
TIME_HIERARCHY_RELATIONS = dict([(
    ('Y', ['h', 'q', 'm', 'U', 'j']),
    ('m', ['d']),
    ('d', ['H']))])
```

dateFormat: ‘YYYYYhqmmm’

- 1945XX04: {‘Y’:1945, ‘m’: 4}
- 19451XXX: {‘Y’:1945, ‘h’: 1}
- 1945X3XX: {‘Y’:1945, ‘q’: 3}
GovData Wrangling: The “Schema”

Unified Space Schema:

```
{‘s’ : ‘Massachusetts’}

{‘s’ : ‘Massachusetts’,
  ‘c’ : ‘Hampden’}

{‘s’ : ‘Massachusetts’,
  ‘c’ : ‘Hampden’,
  ‘f’ : {‘s’ : “25”,
          ‘c’ : “013”}}
```
The GovData Project: the /data APIs

Backend
Database
Search Engine
GIS server
APIs

MongoDb, Solr,
Tornado, Geodjango

/data: full-fledged mongoDB interface to data collections

gov.thedata.org/data?query=Q&collection=ColName

Uses temporal and spatial standardization to make unified simple time and space filters

/data?query=Q&collection=ColName&timeQuery=Monthly&spaceQuery=State
The GovData Project: the /metadata API

Since our metadata repo is a mongoDB, it can be queried as such. E.g. to find the mapping between collection codes and titles:

/metadata?fields=[“code”,”metadata.title”]
GovData Wrangling: Metadata Schema

There is a ‘__metadata__’ collection to which metadata is attached for each collection. Each regular collection (dataset) has a short name, and that name is ID field of the metadata collection.

gov.thedata.org/metadata

All dataset short names in there now:

GovData Wrangling: Metadata Schema

The title attribute:

```json
{"_id": {"$oid": "4c598b918e0d7e149d000539"}, "name": "BLS_ap", "metadata": {"title": "Average Price Data"}}
```

The Source attribute:

```json
{"_id": {"$oid": "4c598b918e0d7e149d000539"}, "source": {"agency": {"shortName": "DOL", "name": "Department of Labor"}, "subagency": {"shortName": "BLS", "name": "Bureau of Labor Statistics"}, "topic": {"name": "Inflation & Prices"}, "subtopic": {"name": "Prices - Consumer"}, "program": {"shortName": "CPI", "name": "Consumer Price Index"}, "dataset": {"shortName": "ap", "name": "Average Price Data"}}, "name": "BLS_ap", "metadata": {"title": "Average Price Data"}}
```
GovData Wrangling: Metadata Schema

Reaching deeper into the metadata structure:
GovData Wrangling: Metadata Schema

The description:

```
{"_id": {"$oid": "4c598b918e0d7e149d000539"}, "name": "BLS_ap", "metadata": {"description": "Survey Description: Average consumer prices are calculated for household fuel, motor fuel, and food items from prices collected for the Consumer Price Index (CPI). Average prices are best used to measure the price level in a particular month, not to measure price change over time. It is more appropriate to use the CU or CW and its individual item indexes to measure price change."}}
```

Keywords:

```
{"_id": {"$oid": "4c598b918e0d7e149d000539"}, "name": "BLS_ap", "metadata": {"keywords": ["CPI", "inflation", "prices"], "title": "Average Price Data"}}
```
GovData Wrangling: Metadata Schema

Column names:

uniqueIndexes and sliceCols:
GovData Wrangling: Metadata Schema

The dateFormat attribute:

```
{ "_id": { "$oid": "4c598b918e0d7e149d000539" }, "name": "BLS_ap", "metadata": { "dateFormat": "YYYYhomme" } }
```

The beginDate and endDate attributes:

```
{ "_id": { "$oid": "4c598b918e0d7e149d000539" }, "name": "BLS_ap", "metadata": { "endDate": { "Y": { "": 2010, "m": { "": 6} }, "beginDate": { "Y": { "": 1973, "m": { "": 10} } } } }
```
GovData Wrangling: Metadata Schema

metadata?action=find_one&fields=[“name”,”metadata.columnGroups”]


timeColNames = Column Names that refer to dates/times

timeColumns = Column Names whose values are dates/times

spaceColNames = Column Names that refer to locations

spaceColumns = Column Names whose values are locations

labelColumns= Column Names whose values are “descriptive”
GovData Wrangling: The Data API

gov.thedata.org/data

The whole dataset.
GovData Wrangling: The Data API

# of records in the whole dataset

```
{"data":1330}
```

Distinct values in the “item” column

```
{"data":"All Ham (Excluding Canned Ham and Luncheon Slices), per lb. (453.6 gm)","All Other Pork (Excluding Canned Ham and Luncheon Slices), per lb. (453.6 gm)","All Pork Chops, per lb. (453.6 gm)","All Uncooked Beef Roasts, per lb. (453.6 gm)","All Uncooked Beef Steaks, per lb. (453.6 gm)","All Uncooked Veal (Excluding Veal), per lb. (453.6 gm)","All uncooked ground beef, per lb. (453.6 gm)","American processed cheese, per lb. (453.6 gm)","Apples, Red Delicious, per lb. (453.6 gm)","Apples, Red Delicious, grade AA, per lb. (453.6 gm)","Apples, Red Delicious, grade AA, per lb. (453.6 gm)","Apples, Red Delicious, per lb. (453.6 gm)","Automotive diesel fuel, per gallon/3.785 liters","Bacon, sliced, per lb. (453.6 gm)","Bacon, sliced, any type, all sizes, per lb. (453.6 gm)","Beans, green, snap (cost per pound/453.6 grams)","Beef for stew, boneless, per lb. (453.6 gm)","Beef liver (cost per pound/453.6 grams)","Bologna, all beef or mixed, per lb. (453.6 gm)","Bourbon whiskey, 375 ml-1.75 liter (cost per 25.4 probationary pint/750 ml)","Bread, French, per lb. (453.6 gm)","Bread, rye, pan (cost per pound/453.6 grams)","Bread, wheat blend, pan (cost per pound/453.6 grams)","Bread, wheat, per lb. (453.6 gm)","Bread, whole wheat, pan, per lb. (453.6 gm)","Broccoli, per lb. (453.6 gm)","Butter, salted, grade AA, stick, per lb. (453.6 gm)","Cabbage, white, per lb. (453.6 gm)","Carrots, short trimmed and topped, per lb. (453.6 gm)"...
```
GovData Wrangling: The Data API

Just the Location, Item, and Jan. 1981 data:

... with 1981 inflation value > .532:
GovData Wrangling: The Data API

querySequence allows stringing of actions, like “find” and “count”:

```javascript
{"data":651}
```

```javascript
{"data":624}
```

```javascript
{"data":473}
```

```javascript
{"data":222}
```

```javascript
{"data":173}
```
GovData Wrangling: The Data API

... or “find” and “distinct”:

```
{"data":173}
```

```
{"data":"Coffee, freeze dried, plain, 3.1-9 ounce (cost per 16 ounces/453.6 grams), "Coffee, instant, plain, 3.1-6 ounce (cost per 16 ounces/453.6 grams), "Coffee, instant, plain, 9.1-14 ounce (cost per 16 ounces/453.6 grams), "Electricity per 500 KWH", "Rib roast, USDA Choice, bone-in, per lb. (453.6 gm), "Steak, T-Bone, USDA Choice, bone-in, per lb. (453.6 gm), "Steak, porterhouse, U.S. choice, bone-in (cost per pound/453.6 grams), "Steak, round, USDA Choice, boneless, per lb. (453.6 gm), "Steak, sirloin, USDA Choice, bone-in, per lb. (453.6 gm), "Utility (piped) gas - 100 therm", "Utility (piped) gas - 40 therm"
```

```
```

GovData Wrangling: The Data API

Or, **skip** the first 30 records and **limit** the response to only the next 10 records:

```shell
gov.thedata.org/data?collection=BLS_ap&querySequence=[{"action":"find","args":[]},{"action":"skip","args":30},{"action":"limit","args":10}]
```
GovData Wrangling: The Data API

Summary:
Exposed actions = find, find_one, count, distinct, skip, limit, sort

Simple single actions, actions are composable

Full MongoDB query language.

Metadata collection is also full MongoDB interface.

All return simple standard JSON. (Also supports JSONP)
The GovData Project: Search Engine

Support “dynamically-grained natural language search”.

Typical search facilities are either too coarse-grained or too fine-grained.

Search subslices of the data, defined by categories naturally present in the data itself (not by a prepicked list).

Without having to use a “menu-driven” DB-specific interface.
GovData Wrangling: Search

Key Idea: “Search Indexing”

1. You have a bunch of documents you want to search

2. You run an “indexing process”, which essentially computes word-frequency statistics for each document, and stores that information in a fast database.

   (the various statistics are like: how many distinct words are present, how many instances of each, how many after removing various common words & word-endings, &c)

3. Then when someone runs a query, the fast database looks to see which documents score highest on the statistics for the words in the query, and returns those first.
Apache Solr is a framework for doing indexing really well and quickly.

Apache Solr is an open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, and rich document (e.g., Word, PDF) handling. Providing distributed search and index replication, Solr is highly scalable.\cite{1}

Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Apache Tomcat. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Solr's powerful external configuration allows it to be tailored to almost any type of application without Java coding, and it has an extensive plugin architecture when more advanced customization is required.

Apache Lucene and Apache Solr are both produced by the same ASF development team since the project merge in 2010. It is common to refer to the technology or products as Lucene/Solr or Solr/Lucene.
Search: what we DON’T do

Typical usage (e.g. by data.gov): index just collection-wide metadata, producing a (small) index that can power a search box.
Data Wrangling: Search

govdata project innovation:

use solr to index slices of collections, not just collection-wide metadata

basically:

1) at collection parse-time, choose columns that define the “meaningful” slices, e.g. Location Industry Dataelement

2) indexing process combinatorially generates all non-empty slices, and indexes query definition as text (plus other features)

3) solr schema combines search over collection-wide search data like keywords with slice-specific information

RESULT: simple interface to query many different databases thats supports dynamically-grained natural language search
Data Wrangling: Search

Govdata usage: produces a large index that enables (weak) schema integration between underlying diverse DBs.

Generates much larger index set, but works well since Solr is optimized for handling hundreds of millions of documents.
The GovData Project: the APIs

gov.thedata.org/data

GovData MongoDB

gov.thedata.org/metadata

/search: full-fledged SOLR interface to search data

http://gov.thedata.org/search?q="employment alabama"
The GovData Project: the /search API

http://gov.thedata.org/search?q="employment alabama"

...search results as a structured document amenable to parsing.
Again, uses temporal and spatial standardization to make unified simple time and space filters.
### Data Wrangling: Search

**The GovData Project**

<table>
<thead>
<tr>
<th>Total Slices</th>
<th>Filter date by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>262306</td>
<td>Month 15169, Year 18351, Quarter 85385, Half 294</td>
</tr>
</tbody>
</table>

Filter space by:

- State 99576
- County 75600
- Metropolitan/Micropolitan Statistical Area


**All Data for:** Dataset: Employment, Hours, and Earnings - State and Metro Area

- **Data_type:** All Employees, In Thousands (1512)
- **Supersector:** Government (277)
- **Industry:** Total Private (1860)
- **Supersector:** Total Private (1909)
- **Data_type:** All Employees, In Thousands (1517)
- **Supersector:** Education and Health Services (1706)
- **Supersector:** Professional and Business Services
- **Data_type:** All Employees, In Thousands (1613)
- **Supersector:** Financial Activities (1411)
- **Supersector:** Leisure and Hospitality
- **Data_type:** All Employees, In Thousands (1333)
- **Supersector:** Durable Goods (3356)
- **Supersector:** Nondurable Goods
- **Supersector:** Professional and business services (1829)
- **Supersector:** Retail trade (1651)
Data Wrangling: Search

The GovData Project

Filter date by:
Month Quarter Year

Filter space by:
State County Metropolitan/Micropolitan Statistical Area

4734 Total Slices


All Data for: Dataelement: Employment Location: Alabama, National, National

Dataclass: Expansions
Dataclass: Gross Job Gains
Dataclass: Openings Ratelevel: Level (U.S. Data In Thousands)
Dataclass: Closings Ratelevel: Rate (Percent)

Dataclass: Openings
Dataclass: Gross Job Losses Ratelevel: Rate (Percent)

Dataclass: Contractions
Dataclass: Gross Job Gains Ratelevel: Level (U.S. Data In Thousands)
Dataclass: Gross Job Losses Ratelevel: Rate (Percent)

Dataclass: Closings Ratelevel: Level (U.S. Data In Thousands)
sliceCols:

The writer of the parser specifies just the column names that generated “meaning slices” as part of the metadata specification.
The GovData Project: the /geo API

gov.thedata.org/geo

/geo/regions: find regions within given radius of given point or within given bounding box

/geo/fips: given standardized region codes, return region name and enclosing regions
The GovData Project: the /geo API
The GovData Project: the `/geo` API

`/geo/regions/s?bounds=-75,40,-73,42`

```
["state_code": "36", "state_name": "New York"], 
{"state_code": "42", "state_name": "Pennsylvania"}, 
{"state_code": "09", "state_name": "Connecticut"}, 
{"state_code": "34", "state_name": "New Jersey"}
```

`/geo/fips?s=25`

```
["state_name": "Massachusetts", "census_division_code": "1", "census_division_name": "New England", "census_region_code": "1", "census_region_name": "Northeast", "state_code": "25"]
```
The GovData Project: **the APIs**

- **Backend**
  - Database
  - Search Engine
  - GIS server
  - APIs
    - Mongodb, Solr, Tornado, Geodjango

- **APIs**
  - `gov.thedata.org/data`
  - `gov.thedata.org/search`
  - `gov.thedata.org/metadata`
  - `gov.thedata.org/geo`
Module 5. The GovData Parser Format

(needed background: A year of general programming, Python familiarity STRONGLY recommended + Modules 3 & 4)
GovData Wrangling: The Parser Format
GovData Wrangling: The Parser Format

```python
NIPA_PARSER = govdata.core.GovParser(NIPA_NAME,
    govdata.core.CsvParser,
    downloader = [(NIPA_downloader,'raw'),
                  (NIPA_preparserr,'preparse1'),
                  (NIPA_preparserr2,'preparse2'),
                  (get_additional_info,'additional_info')],
    trigger = trigger,
    incremental=True)
```
GovData Wrangling: The Parser Format

```python
NIPA_PARSER = govdata.core.GovParser(NIPA_NAME,
    govdata.core.CsvParser,
    downloader = [(NIPA_downloader, 'raw'),
                  (NIPA_preparser1, 'preparse1'),
                  (NIPA_preparser2, 'preparse2'),
                  (get_additional_info, 'additional_info')],
    trigger = trigger,
    incremental=True)

FAT_PARSER = govdata.core.GovParser(FAT_NAME,
    govdata.core.CsvParser,
    downloader = [(FAT_downloader, 'raw'),
                  (FAT_preparser1, 'preparse1'),
                  (FAT_preparser2, 'preparse2')],
    trigger = fat_trigger,
    incremental=True)
```

#==-=---------------------------==NTPA

ITRADE_NAME = 'BEA_InternationalTrade'
ITRADE_PARSER = govdata.core.GovParser(ITRADE_NAME,
    None,
    downloader = [(MakeDirs, 'initialize'),
                  (get_intl_trade, 'get_trade'),
                  (get_intl_trade_goods_detailed, 'get_trade_goods_detailed')]
    )
```
GovData Wrangling: The Parser Format

```python
NIPA_PARSER = govdata.core.GovParser(NIPA_NAME,
    govdata.core.CsvParser,
    downloader = [(NIPA_downloader, 'raw'),
       (NIPA_preparser1, 'preparse1'),
       (NIPA_preparser2, 'preparse2'),
       (get_additional_info, 'additional_info')],
    trigger = trigger,
    incremental=True)
```
GovData Wrangling: The Parser Format

class govdata.core.GovParser

class GovParser(object):
    def __init__(self,
        collectionName,
        parser,
        downloader = None,
        downloadProtocol = None,
        downloadArgs = None,
        downloadKwargs = None,
        parserArgs = None,
        parserKwargs = None,
        trigger = None,
        slicesCorrespondToIndexes = True,
        ID = None,
        incremental = False,
    ):
GovData Wrangling: The Parser Format

class GovParser(object):
    def __init__(self,
        collectionName,
        parser,
        downloader = None,
        downloadProtocol= None,
        downloadArgs = None,
        downloadKwargs = None,
        parserArgs = None,
        parserKwargs = None,
        trigger = None,
        slicesCorrespondToIndexes=True,
        ID = None,
        incremental = False,
    ):
GovData Wrangling: The Parser Format

class GovParser(object):

def __init__(self, collectionName, parser, downloader= None, downloadProtocol= None, downloadArgs= None, downloadKwargs= None, parserArgs= None, parserKwargs= None, trigger= None, slicesCorrespondToIndexes= True, ID= None, incremental= False,):

`collectionName`: string with dataset short name. Must be verifiable: (“/source” API)

`versus:`

```
{"_id": {"Soid": "4d23f93feb0a196203000000"}, "source": {"agency": {"shortName": "DOC", "name": "Department of Commerce"}, "subagency": {"shortName": "BEA", "name": "Bureau of Economic Analysis"}, "program": {"shortName": "REA", "name": "National Economic Accounts"}, "dataset": {"shortName": "NIPA", "name": "National Income and Product Account Tables"}}, "name": "BEA_NIPA"}
```

```
null
```
GovData Wrangling: The Parser Format

collectionName: (str) dataset short name

```python
class GovParser(object):
    def __init__(self,
                 collectionName,
                 parser,
                 downloader = None,
                 downloadProtocol = None,
                 downloadArgs = None,
                 downloadKwargs = None,
                 parserArgs = None,
                 parserKwargs = None,
                 trigger = None,
                 slicesCorrespondToIndexes = True,
                 ID = None,
                 incremental = False,
                 ):
```

Parser = instance of subclass of `govdata.core.DataIterator`

We’ll come back to it
GovData Wrangling: The Parser Format

collectionName: (str) dataset short name

def __init__(self, collectionName, parser, 
      downloader = None, 
      downloadProtocol= None, 
      downloadArgs = None, 
      downloadKwargs = None, 
      parserArgs = None, 
      parserKwargs = None, 
      trigger = None, 
      slicesCorrespondToIndexes=True, 
      ID = None, 
      incremental = False, 
      ):
GovData Wrangling: The Parser Format

downloader = [(NIPA_downloader,'raw'),
              (NIPA_parsers,'preparse'),
              (NIPA_parsers,'preparse2'),
              (get_additional_info,'additional_info')],

@activate
lambda x : 'http://www.bea.gov/national/nipa/{}'.format(x) }

def NIPA_downloader(maindir):
    MakeDirs(maindir)
    get_manifest(maindir)
    connection = pm.Connection()
    incremental = NIPA_NAME in connection['govdata']
    MakeDir(maindir + 'raw/'
    URLBase = 'http://www.bea.gov/national/nipa/
    Yars = ['TableName','FirstYear','LastYear',
            'X = list(map(lambda y: maindir + 'manifest.txt', Yars))
        for x in X:
            NC = x['NumCode']
            Freq = x['Freq']
            if incremental:
                FY = x['FirstYear']
                LY = x['LastYear']
            else:
                FY = 1800
                LY = 2200
                ystr = ''
            url = URLBase + '&'.join([v + '=' + str for v in x])
            topath = maindir + 'raw/' + x['Number']
            WgetMultiple(url,topath)
GovData Wrangling: The Parser Format

def NIPA_downloader(maindir):
    MakeDirs(maindir)
    get_manifest(maindir)
    connection = pm.Connection()
    incremental = NIPA_NAME in connection['govdb']
    MakeDir(maindir + '/raw/')
    URLBase = 'http://www.bea.gov/national/nipa/
    Vars = ['TableName','FirstYear','LastYear'],
    X = tb.tabarray(SVfile = maindir + '/manifest.tsv')
    if incremental:
        FY = x['FirstYear']
        LY = x['LastYear']
    else:
        FY = 1800
        LY = 2200
    url = URLBase + '&'.join([v + '=' + str(x[v]) for v in Vars])
    topath = maindir + '/raw/' + x['Number']
    WgetMultiple(url,topath)

    def NIPA_preparser1(maindir):
        targetdir = maindir + '/preparsed/'
        sourcedir = maindir + '/raw/'
        MakeDir(targetdir)
        M = tb.tabarray(SVfile = maindir + '/manifest.tsv')
        for x in M:
            f = sourcedir + x['Number'].strip('.').strip(' ') + '_x' + x['Number'].strip('.').strip(' ')
            print f
            savepath = targetdir + x['Number'].strip('.').strip(' ') + '_x' + x['Number'].strip('.').strip(' ')
            [X, header, footer, keywords] = NEA_Parser(f, Foo
            metadata = {}
            metadata['Header'] = '\n'.join(header)
            [title, units] = header[:2]
            notes = '\n'.join(header[2:-2])
            [owner, info] = header[-2:]
            metadata['title'] = title
            metadata['description'] = 'National Income and Pr
            metadata['Agency'] = 'DOC'
            metadata['Subagency'] = 'BEA'
            metadata['Type'] = 'National'
            metadata['Category'] = 'NIPA Tables'
            metadata['Section'] = x['Section']
            metadata['Table'] = '\n'.join(title.split(0:1]])
            metadata['Categories'] = '\n'.join(['Agency', 'Su
            metadata['units'] = units.strip('[]')
            metadata['notes'] = notes
            metadata['Owner'] = owner
            metadata['downloadedOn'] = info.split('Last')[0]
GovData Wrangling: The Parser Format

All the downloader / pre-parser functions have to take a single argument which is the directory in which the action is occurring.

The python decorator “activate” is used to express:

a) the files/dirs to be read as input to the step

b) the files/dirs as output of the step

```python
@activate(lambda x : 'http://www.bea.gov/national/nipaweb/csv/NIPATable.csv', lambda x : x[0])
def NIPA_downloader(maindir):
```

```python
@activate(lambda x : (x[0] + 'raw/',x[0] + 'manifest.tsv'), lambda x : x[0] + 'preparsed/')
def NIPA_preparser1(maindir):
```

```python
@activate(lambda x: x[0] + 'preparsed/',lambda x : (x[0] + '__PARSE__/',x[0] + '__metadata.pickle'))
def NIPA_preparser2(maindir):
```
GovData Wrangling: The Parser Format

All the downloader / pre-parser functions have to take a single argument which is the directory in which the action is occurring.

The python decorator “activate” is used to express:

a) the files/dirs to be read as input to the step
b) the files/dirs as output of the step

```python
@activate(lambda argtuple : 'http://www.bea.gov/national/nipaweb/csv/NIPATable.csv', lambda argtuple : argtuple[0])
def NIPA_downloader(maindir):

@activate(lambda argtuple : (argtuple[0] + '/raw/', argtuple[0] + '/manifest.tsv'), lambda argtuple : argtuple[0] + '/preparsed/')
def NIPA_preparser1(maindir):

@activate(lambda argtuple : argtuple[0] + '/preparsed/', lambda argtuple : (argtuple[0] + '__PARSE__/', argtuple[0] + '__metadata.pickle'))
def NIPA_preparser2(maindir):
```

`activate(lambda argtuple : Paths of Read inputs as a function of argtuple, lambda argtuple : Paths of Written outputs as a function of argtuple)`

All paths are relative to the ‘x[0]’, which is always assumed to be the maindir path.
GovData Wrangling: The Parser Format

All the downloader / pre-parser functions have to take a single argument which is the directory in which the action is occurring.

These particular declarations:

```python
@activate(lambda x : 'http://www.bea.gov/national/nipaweb/csv/NIPATable.csv', lambda x : x[0])
def NIPA_downloader(maindir):

@activate(lambda x : (x[0] + 'raw/', x[0] + 'manifest.tsv'), lambda x : x[0] + 'preparsed/'))
def NIPA_preparserr1(maindir):

@activate(lambda x : (x[0] + 'preparsed/', lambda x : (x[0] + '__PARSE__/', x[0] + '__metadata.pickle')))
def NIPA_preparserr2(maindir):
```

.... implicitly establish that:

a) the outputs of the “downloader” step is the whole output directory

b) the inputs of the “preparser1” step are the subdirectory “raw” and the file “manifest.tsv”, which better have been created by the downloader step; and its creates as output the subdirectory ‘preparsed/’

c) the inputs of the “preparser2” step are the subdirectory “preparsed” and it creates the directory ‘__PARSE__’ and file ‘__metadata.pickle’ as output
GovData Wrangling: The Parser Format

All the downloader / pre-parser functions have to take a single argument which is the directory in which the action is occurring.

These particular declarations:

```python
@activate(lambda x: 'http://www.bea.gov/national/nipaweb/csv/NIPATable.csv', lambda x: x[0])
def NIPA_downloader(maindir):

@activate(lambda x: (x[0] + 'raw/', x[0] + 'manifest.tsv'), lambda x: x[0] + 'preparsed/')
def NIPA_preparsr1(maindir):

@activate(lambda x: (x[0] + 'preparsed/', x[0] + '__PARSE__/', x[0] + '__metadata.pickle'))
def NIPA_preparsr2(maindir):
```

So step 1 creates inputs for step 2 which creates inputs for step 3.
GovData Wrangling: The Parser Format

collectionName: (str) dataset short name

class GovParser(object):
    def __init__(self, collectionName, parser, downloader = None, downloadProtocol = None, downloadArgs = None, downloadKwargs = None, parserArgs = None, parserKwargs = None, trigger = None, slicesCorrespondToIndexes=True, ID = None, incremental = False):

A terminal step of the downloader process must create a ‘__PARSE__’ subdirectory, where the clean data -- just prior to DB insertion -- lives.
GovData Wrangling: The Parser Format

class GovParser(object):
    def __init__(self, collectionName, parser, downloader = None, downloadProtocol = None, downloadArgs = None, downloadKwargs = None, parserArgs = None, parserKwargs = None, trigger = None, slicesCorrespondToIndexes = True, ID = None, incremental = False):

Parser = instance of subclass of govdata.core.DataIterator

Python iterator which:
    -- acts on downloaded, cleaned, data in __PARSE__ (and whatever else you have created in the “downloader” process)

    -- and whose “next” method returns ordered dictionary objects suitable to be inserted into MongoDB.
GovData Wrangling: The Parser Format

Parser = instance of subclass of `govdata.core.DataIterator`

```python
class DataIterator(object):
    def __iter__(self):
        return self

    def __getattr__(self, attr):
        try:
            V = self.metadata[''][attr]
        except KeyError:
            raise AttributeError, "Can't find attribute " + attr
        else:
            return V

    def set_source_metadata(self, source_metadata):
        self.metadata[''][source] = source_metadata['source']
```

All DataIterator parser must subclass `govdata.core.DataIterator` and in doing so, provide three additional methods: `__init__`, `refresh`, and `next`. 
GovData Wrangling: The Parser Format

Parser = instance of subclass ofgovdata.core.DataIterator

class CsvParser(DataIterator):
    def __init__(self, source):
        self.metadata = pickle.load(open(source + '__metadata.pickle'))

    def refresh(self, file):
        print('refreshing', file)
        self.Data = tb.tabarray(SVfile = file, verbosity = 0)
        self_IND = 0

    def next(self):
        if self_IND < len(self.Data):
            r = self.Data[self_IND]
            r = OrderedDict([[self.Data.dtype.names[i], float(xx) if isinstance(xx, float) else xx] for i, xx in r.items()])
            if 'subcollections' in r.keys():
                r['subcollections'] = r['subcollections'].split(',' , ')

            for k in self.columnGroups.get('timeColumns', []) + self.columnGroups.get('timeSeries', []):
                if k in r.keys():
                    r[k] = eval(r[k])

            self_IND += 1

            return r

        else:
            raise StopIteration

CsvParser is used to read data from CSVs.

__init__ reads initial metadata from a pickle object in the source folder of the clean data

refresh loads each data file to start reading from it.

next reads each line of data from a source file, one at a time.
GovData Wrangling: The Parser Format

Parser = instance of subclass of `govdata.core.DataIterator`

```python
class CsvParser(DataIterator):
    def __init__(self, source):
        self.metadata = pickle.load(open(source + '__metadata.pickle'))

    def refresh(self, file):
        print('refreshing', file)
        self.Data = tb.tabarray(SVfile = file, verbosity = 0)
        self.IND = 0

    def next(self):
        if self.IND < len(self.Data):
            r = self.Data[self.IND]
            r =OrderedDict([(self.Data.dtype.names[i], float(xx) if isinstance(xx, float) else xx) for i, xx in enumerate(r)])

            if 'subcollections' in r.keys():
                r['subcollections'] = r['subcollections'].split(',')

            for k in self.columnGroups.get('timeColumns', []) + self.columnGroups.get('timeColumns', []):
                if k in r.keys():
                    r[k] = eval(r[k])

            self.IND += 1

            return r

        else:
            raise StopIteration
```

CsvParser is used to read data from CSVs.

The iterator **must** provide a metadata object that defines the specific information described in the metadata schema before.
GovData Wrangling: The Parser Format

Parser = instance of subclass of govdata.core.DataIterator

```python
def checkMetadata(iterator):
    ""
    Sanity check on metadata.
    ""
    metadata = iterator.metadata
    assert isinstance(metadata, dict), 'Metadata is not a dictionary'
    assert '' in metadata.keys(), 'Metadata must contain a blank key'
    assert all(map(lambda x: isinstance(x, dict), metadata.keys())), 'Metadata keys must be dictionaries'
    assert all(map(lambda x: all(map(is_string_like, x.keys())), metadata.values())), 'Metadata values must be string-like'
    M = metadata['']
    assert isinstance(M.get('keywords'), list) and all(map(is_string_like, M.get('description'))), 'Metadata must contain string-like values'
    S = M.get('source')
    assert isinstance(S, list), 'Metadata must contain a list for source'
    try:
        S = OrderedDict(S)
    except:
        print 'Metadata source list in wrong format for
```
GovData Wrangling: The Parser Format

collectionName: (str) dataset short name
downloader = python list of (func,str) pairs

Different iterators can be used to make things more efficient.

After all -- using the CsvParser means you have multiple copies of the whole dataset sitting around, because you have to a) clean it, b) procedure clean csvs, and c) put it into the MongoDB.

The iterator acting on a file doesn’t even really have to see data -- just e.g. arguments that tell it how to get data from the web.
GovData Wrangling: The Parser Format

```python
class GovParser(object):
    def __init__(self,
        collectionName,
        downloader = None,
        parser,
        downloadProtocol = None,
        downloadArgs = None,
        downloadKwargs = None,
        parserArgs = None,
        parserKwargs = None,
        trigger = None,
        slicesCorrespondToIndexes = True,
        ID = None,
        incremental = False,
    ):
```

- **collectionName**: (str) dataset short name (source authentication)
- **downloader** = python list of (func,str) pairs (acquisition and cleaning steps)
- **parser** = subclass govdata.core.DataIterator (insertion step)

The other arguments are more specialized.
Module 5. Javascript
(needed background: basic programming)
How the Web Works Revisited

The real contents of the response.

It’s HTML.
The real contents of the response.

It’s HTML.

But it can also be Javascript.
Server computes the response.
Client computes the response.
How the Web Works (sort of)

So the modes are:

1) HTML  (server-side computation)
2) Javascript (client-side computation)
3) AJAX  (constant communication)

Ajax (programming)

From Wikipedia, the free encyclopedia

For other uses, see Ajax (disambiguation).

Ajax (pronounced /ædʒɪks/; shorthand for Asynchronous JavaScript and XML)\(^1\) is a group of interrelated web development methods used on the client-side to create interactive web applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Data is usually retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not needed, and the requests need not be asynchronous.\(^2\)

Like DHTML and LAMP, Ajax is not one technology, but a group of technologies. Ajax uses a combination of HTML and CSS to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources

2. Use good JS libraries to make JS not suck

3. Use cutting-edge visualization properties of HTML5 for good visualizations

4. Modularize your widgets so that they can talk in generic ways.
1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources
1. Use pure JavaScript; not too much reliance on AJAX; just call general APIs for data sources

```html
<html>
<head>
<script>
// define some functions
// a rarely as possibly, call general-purpose APIs
// to obtain data as JSON or XML
</script>
</head>
<body>
<!-- static structure here -->
</body>
</html>
```
1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources.

Q: What would an alternative look like:

A: You’d rely on server-side computation for:
   - data transformations
   - layout and templating (django / rails)
   - responding to little AJAX requests

Instead, we recommend:
   - do data transformations & event handling in JS
   - do layout in static HTML + templating
   - do AJAX just for “real” data sources
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources.

   Advantages:
   
   1) offloads computations
   
   2) one single, VERY portable file

   Disadvantages:
   
   1) makes everything public (not a disadvantage for us)
   
   2) JS sucks, wouldn’t you rather write your business logic in python?

   *which leads to our second principle* ...
2. Use good JS libraries to make JS not suck
2. Use good JS libraries to make JS not suck

i. jQuery (practically part of stdlib)
   
   object selecting & event handling

ii. underscore.js
   
   better basic data structures (like arrays)

iii. requireJS
   
   modularization, dependency tracking, minification

iv. jQuery.address
   
   deep linking
Javascript: jQuery

i. jQuery (practically part of stdlib)

http://jquery.com/
Javascript: jQuery

The basic thing that jQuery does is it makes JS a great language in which to build simple GUIs.

Four basic ideas:

i. selecting objects in the page.

ii. handling interactive events in the page.

iii. manipulating (changing) the page.

iv. doing AJAX to communicate off the page.

The combination of these four is powerful, and jQuery makes them easy in JS.
CDN HOSTED JQUERY

A number of large enterprises provide hosted copies of jQuery on existing CDN networks that are available for public use. Below are links to the CDN-hosted copies of jQuery that you may hotlink to.

```html
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
</head>
<body>
<!-- static structure here -->
</body>
</html>
```
Javascript: jQuery

```html
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
</head>
<body>
<!-- static structure here -->
</body>
</html>
```
JAVASCRIPT: JQUERY

http://docs.jquery.com/Main_Page

JQUERY API REFERENCE

- jQuery Core
- Selectors
- Attributes
- Traversing
- Manipulation
- CSS
- Events
- Effects
- Ajax
- Utilities
- Internals

There are a number of alternative resources for browsing the API.
Javascript: jQuery

```html
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
</head>
<body>
<div class="box" id="box1">Object 1</div>
<div class="box" id="box2">Object 2</div>
</body>
</html>
```
The Basic Syntax:

```javascript
$(selector string).eventName(function(e) {
    //event response code

});
```

This syntax says:

1. find all the objects in the page that match the selector string
2. and then, bind the event response code to those objects, to be called whenever “eventName” event actually occurs
Javascript: jQuery

The Basic Syntax:

\( $(\text{nearly any JS object}) \)

= that something, “wrapped” as a jQuery object, with all the good methods jQuery provides

\( $(\text{selector string}) \)

= a list of all the things that match the selector string on the page, returned as jQuery objects
Javascript: jQuery

The Basic Syntax:

$(selector string)

$(“tag”)
   Get me all objects whose tag is “tag”

$(“.cname”)
   Get me all objects whose class attribute is “cname”

$(“#idval”)
   Get me all objects whose id attribute is “idval”
Javascript: jQuery

The Basic Syntax:

\$(selector string)

\$(“tag”)
Get me all objects whose tag is “tag”  \$(“div”)

\$(“.cname”)
Get me all objects whose class attribute is “cname”  \$(“.box”)

\$(“#idval”)
Get me all objects whose id attribute is “idval”  \$(“#box1”)

$\(\text{\$\(\text{\texttt{\"\texttt{div}\"}}\))}\)
Javascript: jQuery

```html
<body>
  <div class="box" id="box1">Object 1</div>
  <div class="box" id="box2">Object 2</div>
</body>
```

```javascript
$(".box")
```
Javascript: jQuery

```html
<body>
  <div class="box" id="box1">Object 1</div>
  <div class="box" id="box2">Object 2</div>
</body>
```

jQuery:

```javascript
$("#box1")
```

```
> $(".box")
  [<div class="box" id="box1">Object 1</div>, <div class="box" id="box2">Object 2</div>]
> $("#box1")
  [<div class="box" id="box1">Object 1</div>]
> 
```
Javascript: jQuery

Lots of other kinds of selector methods:

- **All Selector ("***)**
  Selects all elements.

- **:animated Selector**
  Select all elements that are in the progress of an animation at the time the selector is run.

- **Attribute Contains Prefix Selector [name |="value"]**
  Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-).

- **Attribute Contains Selector [name="value"]**
  Selects elements that have the specified attribute with a value containing the a given substring.

- **Attribute Contains Word Selector [name-="value"]**
  Selects elements that have the specified attribute with a value containing a given word, delimited by spaces.

- **Attribute Ends With Selector [name$="value"]**
  Selects elements that have the specified attribute with a value ending exactly with a given string. The comparison is case sensitive.

- **Attribute Equals Selector [name="value"]**
  Selects elements that have the specified attribute with a value exactly equal to a certain value.

- **Attribute Not Equal Selector [name!="value"]**
  Selects elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value.

- **Attribute Starts With Selector [name^="value"]**
  Selects elements that have the specified attribute with a value beginning exactly with a given string.

- **:button Selector**
  Selects all button elements and elements of type button.

- **:checkbox Selector**
  Selects all elements of type checkbox.

- **:checked Selector**
  Matches all elements that are checked.

- **:eq() Selector**
  Select the element at index n within the matched set.

- **:even Selector**
  Selects even elements, zero-indexed. See also odd.

- **:file Selector**
  Selects all elements of type file.

- **:first-child Selector**
  Selects all elements that are the first child of their parent.

- **:first Selector**
  Selects the first matched element.

- **:gt() Selector**
  Select all elements at an index greater than n index within the matched set.

- **Has Attribute Selector [name]**
  Selects elements that have the specified attribute, with any value.

- **:has() Selector**
  Selects elements which contain at least one element that matches the specified selector.

- **:header Selector**
  Selects all elements that are headers, like h1, h2, h3 and so on.

- **:hidden Selector**
  Selects all elements that are hidden.

- **ID Selector ("#id")**
  Selects a single element with the given id attribute.

- **:image Selector**
  Selects all elements of type Image.

- **:input Selector**
  Selects all elements of type input.

- **:link Selector**
  Selects all elements of type link.

- **:not() Selector**
  Selects all elements that don't match the specified selector.

- **:nth-child Selectors**
  Select elements based on their position in a series.

- **:nth-last-child Selectors**
  Select elements based on their position from the end of a series.

- **:nth-last-of-type Selectors**
  Select elements based on their type and position from the end of a series.

- **:nth-of-type Selectors**
  Select elements based on their type and position in a series.

- **:only-child Selector**
  Selects elements that are the only child of their parent.

- **:only-of-type Selector**
  Selects elements that are the only element of a given type within their parent.

- **:parent Selector**
  Selects all parent elements of the matched set.

- **:password Selector**
  Selects all password elements.

- **:radio Selector**
  Selects all radio elements.

- **:root Selector**
  Selects the root element (like body).

- **:selected Selector**
  Matches all elements that are selected.

- **:source Selector**
  Selects all source elements.

- **:span Selector**
  Selects all span elements.

- **:text Selector**
  Selects all text content.

- **:textarea Selector**
  Selects all textarea elements.

- **:type Selector**
  Selects all elements of a given type.

- **:undefined Selector**
  Selects all elements that don't have a type.

- **:valid Selector**
  Selects all valid elements.

- **:visible Selector**
  Selects all visible elements.

- **:voiceover Selector**
  Selects all voiceover elements.

- **:wheel Selector**
  Selects all elements that are related to the wheel event.

- **:xhtml Selector**
  Selects all XHTML elements.
The Basic Syntax: doing stuff with selected results

\[
$(\text{selector string}).\text{method} ( \text{args} )
\]

... means, “apply the method to ALL the objects that match the selector string.”

\[
$(\text{selector string}).\text{eventName} ( \text{args} )
\]

... means, “bind the event handler specified by ‘args’ to ALL objects matching the selector string.”
The Basic Syntax: where “doing stuff” == “binding event”

\[
\$("div").eventName(function(e)\
\{
    //event response code
\});
\]

This syntax says:

i. find all the “div”s in the page

ii. bind the event response to be called whenever “eventName” event actually occurs on a “div”
Javascript: jQuery

```javascript
$(document).ready(function() {
    $('div').click(function() {
        alert("HERE")
    });
});
```
Javascript: jQuery

Lots of event handlers:

- `.bind()`
  Attach a handler to an event for the elements.

- `.blur()`
  Bind an event handler to the "blur" JavaScript event, or trigger that event.

- `.change()`
  Bind an event handler to the "change" JavaScript event, or trigger that event.

- `.click()`
  Bind an event handler to the "click" JavaScript event, or trigger that event.

- `.dblclick()`
  Bind an event handler to the "dblclick" JavaScript event, or trigger that event.

- `.delegate()`
  Attach a handler to one or more events for all elements that match on a specific set of root elements.

- `.die()`
  Remove all event handlers previously attached using `.live()`.

- `.error()`
  Bind an event handler to the "error" JavaScript event.

- `.event.currentTarget`
  The current DOM element within the event bubbling phase.

- `.event.data`
  The optional data passed to jQuery.fn.bind when the current event exec.

- `.event.preventDefault()`
  Returns `true` if the `event`'s default action was prevented.

- `.event.stopPropagation()`
  Prevents the event from bubbling up the DOM tree, preventing any parent handle event.

- `.event.target`
  The DOM element that initiated the event.

- `.event.timeStamp`
  The difference in milliseconds between the time an event is triggered and January 1, 1970.

- `.event.type`
  Describes the nature of the event.

- `.event.which`
  For key or button events, this attribute indicates the specific button or key that was pressed.

- `.focus()`
  Bind an event handler to the "focus" JavaScript event, or trigger that event.

- `.focusin()`
  Bind an event handler to the "focusin" JavaScript event.

- `.focusout()`
  Bind an event handler to the "focusout" JavaScript event.

- `.mousedown()`
  Bind an event handler to the "mousedown" JavaScript event, or trigger that event.

- `.mouseenter()`
  Bind an event handler to be fired when the mouse enters an element.

- `.mouseleave()`
  Bind an event handler to be fired when the mouse leaves an element.

- `.mousemove()`
  Bind an event handler to the "mousemove" JavaScript event, or trigger that event.

- `.mouseout()`
  Bind an event handler to the "mouseout" JavaScript event, or trigger that event.

- `.mouseover()`
  Bind an event handler to the "mouseover" JavaScript event, or trigger that event.

- `.mouseup()`
  Bind an event handler to the "mouseup" JavaScript event, or trigger that event.

- `.one()`
  Attach a handler to an event for the elements. The handler is executed only once.

- `.jQuery.proxy()`
  Takes a function and returns a new one that will always have a particular context.

- `.ready()`
  Specify a function to execute when the DOM is fully loaded.

- `.resize()`
  Bind an event handler to the "resize" JavaScript event, or trigger that event.

- `.scroll()`
  Bind an event handler to the "scroll" JavaScript event, or trigger that event.
**Javascript: jQuery**

jQuery does page manipulations:

```
$(selector string).eventName(function(e){
    $(selector).manipulateSomehow();
});
```
Javascript: jQuery
jQuery does page manipulations:

“When you click on box1, add “hello” text to box 2.”


“When you click on box1, add “hello” text to box 2. But if you click on box 2, remove all the hello text objects.”
Javascript: jQuery

jQuery does page manipulations:

“When you click on box1, add “hello” text to box 2.

But if you click on box 2, remove all the hello text objects ... and count them.”

```html
<script>
$(document).ready(function(){
    $('#box1').click(function(){
        $('#box2').append('<div class="hbox">hello</div>');
    });
    $('#box2').click(function(){
        var hboxes = $('.hbox');
        var l = hboxes.length;
        $('.hbox').remove();
        $('#box2').append('<br/>Removed " + l + " things.");
    });
});
</script>
```
Javascript: jQuery

Lots of manipulations.

- **.addClass()**: Adds the specified class(es) to each of the set of matched elements.

- **.after()**: Insert content, specified by the parameter, after each element in the set.

- **.append()**: Insert content, specified by the parameter, to the end of each element in the set.

- **.appendTo()**: Insert every element in the set of matched elements to the end of the target.

- **.attr()**: Get the value of an attribute for the first element in the set of matched elements.

- **.before()**: Insert content, specified by the parameter, before each element in the set.

- **.clone()**: Create a deep copy of the set of matched elements.

- **.css()**: Get the value of a style property for the first element in the set of matched elements.

- **.detach()**: Remove the set of matched elements from the DOM.

- **.empty()**: Remove all child nodes of the set of matched elements from the DOM.

- **.hasClass()**: Determine whether any of the matched elements are assigned the given class.

- **.height()**: Get the current computed height for the first element in the set of matched elements.

- **.html()**: Get the HTML contents of the first element in the set of matched elements.

- **.innerHeight()**: Get the current computed height for the first element in the set of matched elements.

- **.outerWidth()**: Get the current computed width for the first element in the set of matched elements.

- **.position()**: Get the current coordinates of the first element in the set of matched elements.

- **.prepend()**: Insert content, specified by the parameter, to the beginning of each element in the set.

- **.prependTo()**: Insert every element in the set of matched elements to the beginning of the target.

- **.remove()**: Remove the set of matched elements from the DOM.

- **.removeAttr()**: Remove an attribute from each element in the set of matched elements.

- **.removeClass()**: Remove a single class, multiple classes, or all classes from each element.

- **.replaceAll()**: Replace each target element with the set of matched elements.

- **.replaceWith()**: Replace each element in the set of matched elements with the provided content.

- **.scrollLeft()**: Get the current horizontal position of the scroll bar for the first element in the set of matched elements.

- **.scrollTop()**: Get the current vertical position of the scroll bar for the first element in the set of matched elements.

- **.text()**: Get the combined text contents of each element in the set of matched elements.
Javascript: jQuery

jQuery does AJAX

\[
\$(\text{selector string}).\text{eventName}(function(e)\
\{ \text{var send\_data = get\_data(e);}\
\text{\$.ajax({}
\text{url: "http://my\_api\_url",}
\text{data : send\_data}
\text{success : doSomethingWithResults}
\text{})});
\})
\]

where \text{doSomethingWithResults} is a function that does jQuery manipulations and \text{get\_data} computes arguments from data about the event.
jQuery does AJAX

“When you click on box1, ask the API how many records there are in BEA_NIPA collection and show the result in box2.”
so that the search only happens when you press “Enter”

Here’s where the input box is put in. It’s a standard HTML thing.
2. Use good JS libraries to make JS not suck

   i. jQuery (practically part of stdlib)

       object selecting & event handling

   ii. underscore.js

       better basic data structures (like arrays)

   iii. requireJS

       modularization, dependency tracking, minification

   iv. jQuery.address

       deep linking
Javascrip: Underscore

ii. underscore.js    better basic data structures (like arrays)

Underscore.js

Underscore is a utility-belt library for JavaScript that provides a lot of the functional programming support that you would expect in Prototype.js (or Ruby), but without extending any of the built-in JavaScript objects. It's the tie to go along with jQuery's tux.

Well, the basic idea is that:

a. Real programming languages have good data structures.

b. So Javascript should have them too.
ii. underscore.js  better basic data structures (like arrays)

Object-Oriented and Functional Styles

Collections
each, map, reduce, reduceRight, detect, select, reject, all, any, include, invoke, pluck, max, min, sortBy, sortedIndex, toArray, size

Arrays
first, rest, last, compact, flatten, without, uniq, intersect, zip, indexOf, lastIndexOf, range

Functions
bind,/bindAll, memoize, delay, defer, throttle, debounce, wrap, compose

Objects
keys, values, functions, extend, clone, tap, isEqual, isEmpty, isElement, isArray, isArguments, isFunction, isString, isNumber, isBoolean, isDate, isRegExp, isNaN, isNull, isUndefined

Utility
noConflict, identity, times, mixin, uniqueld, template

Chaining
chain, value
Javascript: Underscore

ii. underscore.js  

better basic data structures (like arrays)

_.each([1, 2, 3], function(num){ alert(num); });
=> alerts each number in turn...
_.each({one : 1, two : 2, three : 3}, function(num, key){ alert(num); });
=> alerts each number in turn...

_.all([true, 1, null, 'yes']);
=> false

_.intersect([1, 2, 3], [101, 2, 1, 10], [2, 1]);
=> [1, 2]

_.zip(['moe', 'larry', 'curly'], [30, 40, 50], [true, false, false])
=> [['moe', 30, true], ['larry', 40, false], ['curly', 50, false]]

_.keys({one : 1, two : 2, three : 3});
=> ['one', 'two', 'three']
Javascrip: Underscore

ii. underscore.js   better basic data structures (like arrays)

And the “right” idea of equality for regular situations.

```javascript
var moe = {name: 'moe', luckyNumbers: [13, 27, 34]};
var clone = {name: 'moe', luckyNumbers: [13, 27, 34]};
moe === clone;
=> false
_.isEqual(moe, clone);
=> true
```

Underscore.strings, an Underscore extension that adds functions for string-manipulation: trim, startsWith, contains, capitalize, reverse, sprintf, and more.
2. Use good JS libraries to make JS not suck

i. jQuery (practically part of stdlib)
   object selecting & event handling

ii. underscore.js
   better basic data structures (like arrays)

iii. requireJS
    modularization, dependency tracking, minification

iv. jQuery.address
    deep linking
The basic idea is that:

a. Real programming languages modularize code.

b. So Javascript should too.

requireJS is a little like Javascript’s answer to the idea of “import” in python
Javascript: requireJS

iii. requireJS modularization, dependency tracking, minification

http://requirejs.org/

/* ---

RequireJS is a JavaScript file and module loader. It is optimized for in-browser use, but it can be used in other JavaScript environments, like Rhino and Node. Using a modular script loader like RequireJS will improve the speed and quality of your code.

IE 6+ .......... compatible ✔
Firefox 2+ .... compatible ✔
Safari 3.2+ ..... compatible ✔
Chrome 3+ .... compatible ✔
Opera 10+ ...... compatible ✔

Get started then check out the API.

--- */
**Javascript: requireJS**

iii. requireJS  modularization, dependency tracking, minification

```
<html>
<head>
    <script data-main="main" src="scripts/require.js"></script>
</head>
<body>
    <!-- static contents here -->
</body>
</html>
```

```
require(['helper/util'], function() {
    require.read(function() {
        //When this is called, stuff defined in
        //helper.utils is available
        //But it doesn't enter a global namespace.
    });
});
```
iii. requireJS  modularization, dependency tracking, minification

```javascript
require(['http://gov.thedata.org/data?collectionName=BEA_NIPA&action=count&callback=define'],
function(count){
    console.log(count)
};
});

define(['/path/to/mod1','/path/to/mod2'],function(mod1,mod2){
    var newFunc(a,b){
        return mod1.func1(a,b) + mod2.constant
    }
});
```
When the page has Javascript, the message contains the JS source code. So you want it to be short.
Javascript: requireJS

iii. requireJS  modularization, dependency tracking, minification

OPTIMIZE

Once you are finished doing development and want to deploy your code for your end users, you can use the optimization to combine the JavaScript files together and minify it. In the example above, it can combine main.js and helper/util.js into one file and minify it using Google's Closure Compiler.

```bash
.../requirejs/build/build.sh app.build.js
```
Javascript: requireJS

```bash
../requirejs/build/build.sh app.build.js
```
Javascript: requireJS

When the page has Javascript, the message contains the JS source code. So you want it to be short.

And readability doesn’t count.
Javascript: Principles

2. Use good JS libraries to make JS not suck

i. jQuery *(practically part of stdlib)*
   - object selecting & event handling

ii. underscore.js
   - better basic data structures (like arrays)

iii. requireJS
   - modularization, dependency tracking, minification

iv. jQuery.address
   - deep linking
Deep linking
From Wikipedia, the free encyclopedia

On the World Wide Web, deep linking is making a hyperlink that points to a specific page or image on a website, instead of that website's main or home page. Such links are called deep links.

Deep linking and web technologies

Websites which are built on web technologies such as Adobe Flash and AJAX often do not support deep linking. This can result in usability problems for people visiting such websites. For example, visitors to these websites may be unable to save bookmarks to individual pages or states of the site, web browser forward and back buttons may not work as expected, and use of the browser's refresh button may return the user to the initial page.
Deep linking and web technologies

Websites which are built on web technologies such as Adobe Flash and AJAX often do not support deep linking. This can result in usability problems for people visiting such websites. For example, visitors to these websites may be unable to save bookmarks to individual pages or states of the site, web browser forward and back buttons may not work as expected, and use of the browser’s refresh button may return the user to the initial page.

Usually, DB-backed pages have URLs that look like this:

http://base_address/api_path/?queryString

where queryString is like “key1=val1&key2=val2”

The Javascript actions don’t get registered in this scheme.
Javascript: Deep Linking

iv. jQuery.address deep linking

http://www.asual.com/jquery/address/

jQuery Address - Deep linking for the masses

The jQuery Address plugin provides powerful deep linking capabilities and allows the creation of unique virtual addresses that can point to a website section or an application state. It enables a number of important capabilities including:

- Bookmarking in a browser or social website
- Sending links via email or instant messenger
- Finding specific content using the major search engines
- Utilizing browser history and reload buttons
Javascript: Deep Linking

Basic idea:

- make sure every type of “state change” in the JS page is encoded in the URL, via jQuery events (make sure to separate the way that the api variables and the JS states are encoded)

- load up the page based on parsing the encoded URL, where the API sees the variables and the JS gets the state encoding

- use a tool like jQuery.address to do the encoding & decoding
Javascript: Deep Linking

http://base_address/#/api_path/?queryString#encoded_state

```javascript
require(['"find","jquery","jquery.address"'],
    function(find) {
      $.address.init(function(e) {
        e).externalChange(function(e) {
          var state = $.address.jsonhash();
          state = state || {};
          var params = $.address.parameters();
          if (e.path === "show") { // show
            console.log("SHOW");
            show.load(params, state);
          } else { // find
            $.address.path('find');
            find.load(params, state);
          }
        });
    });
```
<table>
<thead>
<tr>
<th>Javascript: Deep Linking</th>
</tr>
</thead>
</table>

**The GovData Project**

|--------------------|--------------------------------------------|-------------------------------------------|

*Department of Commerce > Bureau of Economic Analysis > National Economic Accounts > NIPA Tables (2681 Slices)*
HTML5

From Wikipedia, the free encyclopedia

HTML5 is the next major revision of the HTML standard, currently under development.

<nav>  <footer>

<audio>  <video>
<canvas>

inline svg

emphasis on easing JS integration
Scalable Vector Graphics

From Wikipedia, the free encyclopedia

"SVG" redirects here. For other uses, see SVG (disambiguation).

Scalable Vector Graphics (SVG) is a family of specifications of an XML-based file format for describing two-dimensional vector graphics, both static and dynamic (i.e. interactive or animated).
# Javascript + SVG

Variety of Elements to combine.

## SVG Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Defines a hyperlink</td>
</tr>
<tr>
<td>altGlyph</td>
<td>Allows control over glyphs used to render particular character data (e.g., for music symbols or Asian text)</td>
</tr>
<tr>
<td>altGlyphDef</td>
<td>Defines a set of glyph substitutions (e.g., for music symbols or Asian text)</td>
</tr>
<tr>
<td>altGlyphItem</td>
<td>Defines a candidate set of glyph substitutions (e.g., for music symbols or Asian text)</td>
</tr>
<tr>
<td>animate</td>
<td>Animates an attribute or property over time</td>
</tr>
<tr>
<td>animateColor</td>
<td>Specifies a color transformation over time</td>
</tr>
<tr>
<td>animateMotion</td>
<td>Causes an element to move along a motion path</td>
</tr>
<tr>
<td>animateTransform</td>
<td>Animates a transformation attribute on an element</td>
</tr>
<tr>
<td>circle</td>
<td>Defines a circle</td>
</tr>
<tr>
<td>clipPath</td>
<td></td>
</tr>
<tr>
<td>color-profile</td>
<td>Specifies a color profile description</td>
</tr>
<tr>
<td>cursor</td>
<td>Defines a platform-independent cursor</td>
</tr>
<tr>
<td>definition-src</td>
<td>Defines a separate font definition resource</td>
</tr>
<tr>
<td>defs</td>
<td>A container for referenced elements</td>
</tr>
<tr>
<td>disc</td>
<td>A text-only description for elements in SVG - not displayed as part of the graphics. User agents may display the text as a tooltip</td>
</tr>
<tr>
<td>ellipse</td>
<td>Defines an ellipse</td>
</tr>
<tr>
<td>feBlend</td>
<td>SVG Filter. Composites two objects together using different blending modes</td>
</tr>
<tr>
<td>feColorMatrix</td>
<td>SVG Filter. Applies a matrix transformation</td>
</tr>
<tr>
<td>feComponentTransfer</td>
<td>SVG Filter. Performs component-wise remapping of data</td>
</tr>
<tr>
<td>feComposite</td>
<td>SVG Filter.</td>
</tr>
<tr>
<td>feConvolveMatrix</td>
<td>SVG Filter.</td>
</tr>
<tr>
<td>feDiffuseLighting</td>
<td>SVG Filter.</td>
</tr>
<tr>
<td>feDropShadow</td>
<td>SVG Filter.</td>
</tr>
<tr>
<td>filter</td>
<td>SVG Filter.</td>
</tr>
<tr>
<td>font-face-url</td>
<td>A container element for grouping together related elements</td>
</tr>
<tr>
<td>glyph</td>
<td>Defines the graphics for a given glyph</td>
</tr>
<tr>
<td>glyphRef</td>
<td>Defines a possible glyph to use</td>
</tr>
<tr>
<td>hixem</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td></td>
</tr>
<tr>
<td>line</td>
<td>Defines a line</td>
</tr>
<tr>
<td>linearGradient</td>
<td>Defines a linear gradient</td>
</tr>
<tr>
<td>marker</td>
<td></td>
</tr>
<tr>
<td>mask</td>
<td></td>
</tr>
<tr>
<td>metadata</td>
<td>Specifies metadata</td>
</tr>
<tr>
<td>missing-glyph</td>
<td></td>
</tr>
<tr>
<td>mpath</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>Defines a path</td>
</tr>
<tr>
<td>pattern</td>
<td></td>
</tr>
<tr>
<td>polygon</td>
<td>Defines a closed shape that consists of a set of connected straight lines</td>
</tr>
<tr>
<td>polyline</td>
<td>Defines a set of connected straight lines</td>
</tr>
<tr>
<td>radialGradient</td>
<td>Defines a radial gradient</td>
</tr>
<tr>
<td>rect</td>
<td>Defines a rectangle</td>
</tr>
<tr>
<td>script</td>
<td>Container for scripts (e.g., ECMAScript)</td>
</tr>
<tr>
<td>set</td>
<td>Sets the value of an attribute for a specified duration</td>
</tr>
<tr>
<td>stop</td>
<td></td>
</tr>
<tr>
<td>style</td>
<td>Allows style sheets to be embedded directly within SVG content</td>
</tr>
<tr>
<td>svg</td>
<td>Defines an SVG document fragment</td>
</tr>
<tr>
<td>switch</td>
<td></td>
</tr>
<tr>
<td>symbol</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td></td>
</tr>
<tr>
<td>textPath</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>A text-only description for elements in SVG - not displayed as part of the graphics. User agents may display the text as a tooltip</td>
</tr>
<tr>
<td>tref</td>
<td></td>
</tr>
<tr>
<td>tsfpan</td>
<td></td>
</tr>
<tr>
<td>use</td>
<td></td>
</tr>
<tr>
<td>view</td>
<td></td>
</tr>
<tr>
<td>v kern</td>
<td></td>
</tr>
</tbody>
</table>
Javascript + SVG

http://www.w3schools.com/svg/svg_examples.asp

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <rect width="300" height="100"
    style="fill:rgb(0,0,255);stroke-width:1;
    stroke:rgb(0,0,0)"/>
</svg>
```
Javascript + SVG

http://www.w3schools.com/svg/svg_examples.asp

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <path d="M153 334
C153 334 151 334 151 334
C151 339 153 344 156 344
C164 344 171 339 171 334
C171 322 164 314 156 314
C142 314 131 322 131 334
C131 350 142 364 156 364
C175 364 191 350 191 334
C191 311 175 294 156 294
C131 294 111 311 111 334
C111 361 131 384 156 384
C186 384 211 361 211 334
C211 300 186 274 156 274"
style="fill:white;stroke:red;stroke-width:2"/>
</svg>
```
<xml version="1.0" standalone="no">
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
</svg>
Javascript + SVG

http://www.w3schools.com/svg/svg_examples.asp

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<defs>
<radialGradient id="grey_blue" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
<stop offset="0%" style="stop-color:rgb(200,200,200);stop-opacity:0"/>
<stop offset="100%" style="stop-color:rgb(0,0,255);stop-opacity:1"/>
</radialGradient>
</defs>
<ellipse cx="230" cy="200" rx="110" ry="100"
style="fill:url(#grey_blue)"/>
</svg>
```
Javascript + SVG

PITA to write SVG directly. Use, e.g. RaphaelJS

http://raphaeljs.com/

Raphaël—JavaScript Library

What is it?
Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

Also SVG is not supported in IE until 9. But Raphael renders to IE’s equivalent ..
Javascript + SVG

```javascript
// Creates canvas 320 x 200 at 10, 50
var paper = Raphael(10, 50, 320, 200);

// Creates circle at x = 50, y = 40, with radius 10
var circle = paper.circle(50, 40, 10);
// Sets the fill attribute of the circle
circle.attr("fill", "#f00");

// Sets the stroke attribute of the circle
circle.attr("stroke", "#fff");
```
Javascript + SVG: protoviz

http://vis.stanford.edu/protovis/

Protovis

A GRAPHICAL APPROACH TO VISUALIZATION

Protovis composes custom views of data with simple marks such as bars and dots. Unlike low-level graphics libraries that quickly become tedious for visualization, Protovis defines marks through dynamic properties that encode data, allowing inheritance, scales and layouts to simplify construction.

Protovis is free and open-source, provided under the BSD License. It uses JavaScript and SVG for web-native visualizations; no plugin required (though you will need a modern web browser)! Although programming experience is helpful, Protovis is mostly declarative and designed to be learned by example.

This project is led by Mike Bostock and Jeff Heer of the Stanford Visualization Group, with significant help from Vadim Ogievetsky. We welcome your contributions and suggestions.
Javascript + SVG: protoviz

http://vis.stanford.edu/protovis/

**Protovis**

A GRAPHICAL APPROACH TO VISUALIZATION

Protovis composes custom views of data with simple marks such as bars and dots. Unlike low-level graphics libraries that quickly become tedious for visualization, Protovis defines marks through dynamic properties that encode data, allowing inheritance, scales and layouts to simplify construction.

Protovis is free and open-source, provided under the BSD License. It uses JavaScript and SVG for web-native visualizations; no plugin required (though you will need a modern web browser)! Although programming experience is helpful, Protovis is mostly declarative and designed to be learned by example.

This project is led by Mike Bostock and Jeff Heer of the Stanford Visualization Group, with significant help from Vadim Ogievetsky. We welcome your contributions and suggestions.
Javascript + SVG: protoviz

http://vis.stanford.edu/protovis/

**Gallery**

Enjoy these sample visualizations built with Protovis. For any example, use your browser to view the source or the backing dataset. If you've built something cool with Protovis, let us know or send a merge request, and we'll add it to the gallery!

**Conventional**

While Protovis is designed for custom visualization, it is still easy to create many standard chart types. These simpler examples serve as an introduction to the language, demonstrating key abstractions such as quantitative and ordinal scales, while hinting at more advanced features, including stack layout.

- Area Charts
- Bar & Column Charts
- Scatterplots
- Pie & Donut Charts
- Line & Step Charts
- Stacked Charts
- Grouped Charts
Javascript + SVG: protoviz

- Anderson's Flowers
- Becker's Barley
- Bertin's Hotel
- Streamgraphs

- Sparklines
- Bullet Charts
- Bubble Charts
- Sizing the Horizon

- Candlestick Charts
- Bertin's Antibiotics
- Nightingale's Rose
- Playfair's Wheat

- Gas & Driving
- Seattle Weather
- Marey's Trains
- Stemplots
Javascript + SVG: protoviz

Maps

Protovis offers two avenues of visualizing geospatial data: build on top of existing browser-based map tools (such as Google Maps or OpenLayers), or use our own geo scales for custom visualization design.
Canvas element

From Wikipedia, the free encyclopedia

The **canvas** element is part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It is a low level, procedural model that updates a bit map and does not have a built in scene graph.

**Bitmap instead of vector**

```html
<canvas> is a new HTML element which can be used to draw graphics using scripting (usually JavaScript). It is a fast interface to the canvas underlying technology,
```

- Rectangles
- Paths
- Lines
- Arcs
- Ellipses

**MoveTo**

**Transformations**

**Animations** (sort of)

*fast for many objects, but you don’t have DOM control.*
<html>
<head>
<script type="application/javascript">
function draw() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext) {
        var ctx = canvas.getContext("2d");

        ctx.fillStyle = "rgb(200,0,0)";
        ctx.fillRect(10, 10, 55, 50);

        ctx.fillStyle = "rgba(0, 0, 200, 0.5)"
        ctx.fillRect(30, 30, 55, 50);
    }
}
</script>
</head>
<body onload="draw();">
<canvas id="canvas" width="150" height="150"></canvas>
</body>
</html>
<html>
<head>
<script type="application/javascript">
function draw() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        ctx.beginPath();
        ctx.arc(75,75,50,0,Math.PI*2,false); // Outer circle
        ctx.moveTo(110,75);
        ctx.arc(75,75,35,0,Math.PI,false); // Mouth (clockwise)
        ctx.moveTo(65,65);
        ctx.arc(60,65,5,0,Math.PI*2,false); // Left eye
        ctx.moveTo(95,65);
        ctx.arc(90,65,5,0,Math.PI*2,false); // Right eye
        ctx.stroke();
    }
}
</script>
</head>
<body onload="draw();">
    <canvas id="canvas" width="150" height="150"></canvas>
</body>
</html>
<script type="application/javascript">
function draw() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        for(var i=0;i<4;i++) {
            for(var j=0;j<3;j++) {
                ctx.beginPath();
                var x = 25+j*50;  // x coordinate
                var y = 25+i*50;  // y coordinate
                var radius = 20; // Arc radius
                var startAngle = 0; // Starting point
                var endAngle = Math.PI+(Math.PI*j)/2; // End point on circle
                var anticlockwise = i%2==0 ? false : true; // clockwise or anticlockwise

                ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise);

                if (i>1) {
                    ctx.fill();
                } else {
                    ctx.stroke();
                }
            }
        }
    }
}</script>
```html
<html>
<head>
<script type="application/javascript">
function draw () {
    var ctx = document.getElementById('canvas').getContext('2d');
    ctx.fillRect(0, 0, 300, 300);
    for (var i=0; i<3; i++) {
        for (var j=0; j<3; j++) {
            ctx.save();
            ctx.strokeStyle = "#9CFF00";
            ctx.translate(50 + j * 100, 50 + i * 100);
            drawSpirograph(ctx, 20 * (j+2)/(j+1), -8*(i+3)/(i+1), 10);
            ctx.restore();
        }
    }
}
function drawSpirograph(ctx, R, r, i) {
    var x1 = R - i;
    var y1 = 0;
    var r = 1;
    ctx.beginPath();
    ctx.moveTo(x1, y1);
    do {
        if (i > 20000) break;
        var x2 = (R + r + Math.cos(i * Math.PI / 72)) - (r + 0) * Math.cos((R + r + i) * Math.PI / 72);
        var y2 = (R + r + Math.sin(i * Math.PI / 72)) - (r + 0) * Math.sin((R + r + i) * Math.PI / 72);
        ctx.lineTo(x2, y2);
        x1 = x2;
        y1 = y2;
        i++;
    } while (x2 != R - 0 && y2 != 0);
    ctx.stroke();
}
</script>
</head>
<body onload="draw();">
<canvas id="canvas" width="350" height="350"></canvas>
</body>
</html>
```
Javascript + Canvas

**globalCompositeOperation**

We can not only draw new shapes behind existing shapes but we can also use it to mask off certain areas, clear sections from the canvas (not limited to rectangles like the `clearRect` method does) and more.

globalCompositeOperation = type

type is a string representing any one of twelve compositing operations. Each of the available types is described below.

**Note:** In all of the examples below the blue square is drawn first and referred to as 'existing canvas content'. The red circle is drawn second and referred to as 'new shape'.

- **source-over (default)**
  This is the default setting and draws new shapes on top of the existing canvas content.

- **source-in**
  The new shape is drawn only where both the new shape and the destination canvas overlap. Everything else is made transparent.

- **source-out**
  The new shape is drawn where it doesn't overlap the existing canvas content.

- **source-atop**
  The new shape is only drawn where it overlaps the existing canvas content.

- **destination-over**
  New shapes are drawn behind the existing canvas content.

- **destination-in**
  The existing canvas content is kept where both the new shape and existing canvas content overlap. Everything else is made transparent.

- **destination-out**
  The existing content is kept where it doesn't overlap the new shape.

- **destination-atop**
  The existing canvas is only kept where it overlaps the new shape.
Javascript + Canvas: processingJS

http://processingjs.org/

Processing is an open source programming language and environment for people who want to create images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context, Processing also has evolved into a tool for generating finished professional work. Today, there are tens of thousands of software developers, artists, designers, educators, and students who use Processing as a learning platform, design space, and creative tool.
Javascript + Canvas: processingJS

Exhibition

Galactic Inbox
by Paul at Monocubed
This is a keyboard-driven game written by the GMall team.
Links: @monocubed, monocubed.com

facebook-privacy
by Matt McKeon
Visualization based on the evolution of privacy on Facebook.
Links: @mattmckeon, thvcl.com

ABSTRACT01.js
by Marius Watz
abstract01.js is a reworking of an old 2D Processing sketch from 2003.
Links: Marius Watz

JS NINJA
by Allister MacDonald
This is a mouse-driven game that allows your ninja to build up their chi-foam.
Links: Hyper-Metrix.com

askken.heroku.com
by Michael Aufler at Quasipartikel Labs
Visualization based on search engines.
Links: @_mqi, quasipartikel.at

BitTorrent Visualizations
by Chris Lee
A simplified BitTorrent visualization originally written by Abram Stem, updated by Jeff Awcock. Read Jeff Awcock’s blog.

Twitch
by Casey Reas
Twitch is a series of minimal

Ball Droppings
by Josh Nimoy
Turn your sound up. Draw lines.

Wiki Visualizations
by Matt Ryall
A small collection of animated...
Javascript + random: google vis API

http://code.google.com/apis/charttools
Welcome to the SlickGrid!

Update: the trunk has been switched to 2.0 alpha. The latest stable release is 1.4.3 (tag).

What it is

Do you use SlickGrid?
Add your site to the Used By!

Quite simply, SlickGrid is a JavaScript grid/spreadsheet component. It is an advanced component and is going to be a bit more difficult to learn and configure, but once you realize its full potential, it will blow your mind!

https://github.com/mleibman/SlickGrid/wiki/Examples
Javascript: Examples

Everyone
Sleeping, eating, working and watching television take up about two-thirds of the average day.
Javascript: Examples

http://www.babynamewizard.com/voyager
Javascript: Examples

http://oakland.crimespotting.org/
Visualization: Blogs & Examples

http://www.visualcomplexity.com