We Have a Lot of Data

- 90% of the world’s data generated in last two years alone (2013)
- Sloan Sky Server stores 10s of TB per day
- Hadron Collider can generate 500 Exabytes per day
- How was all this data stored and accessed?
Then came the Web

• The internet is great, since it allows us to easily share and create data, and lots of it

• Bots can be used to automatically collect information from the web

• This is where Web Crawling and Web Scraping come into play
Crawling vs. Scraping

- **Web Crawling** is what Google, Yahoo, Bing! do
  - Collect web pages, index them to search

- **Web Scraping** collects data stored on web pages
  - Collect info presented on a page
  - The deep web is difficult to scrape…
Scraping Data

• How might one scrape data from a webpage?

• If there is data stored in the deep web, will a web scraper be able to find it?
People’s Heights Inc.

Some People

<table>
<thead>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ben</td>
<td>McCamish</td>
<td>5'11&quot;</td>
</tr>
<tr>
<td>Jonie</td>
<td>Jones</td>
<td>6'6&quot;</td>
</tr>
<tr>
<td>Johnny</td>
<td>Johnson</td>
<td>5'5&quot;</td>
</tr>
<tr>
<td>Smithy</td>
<td>Smith</td>
<td>4'4&quot;</td>
</tr>
</tbody>
</table>

• Suppose there is a company that shows people’s names and heights when you visit their webpage

• Unfortunately, they don’t have a database for their data and simply store it in the webpage.

• You are tasked with creating a database using their website as the source of all data
People’s Heights Inc.

- The first step is to grab the page in raw format, it happens to be in HTML
- Here is a snippet of the HTML storing the tables of data
- Let's write some pseudocode
- Be sure to think of how you might explore the page(s)
Some Dangers of Crawling

• A cluster running crawlers over a single domain might be confused with what?

• Some sites have a ToS that restricts the use of their data that is displayed on the website

• Your IP can get banned

✦ If you scrape data from their page, is it safe?
In this class...

- Wikipedia’s API can be used to access deep web

- One of the formats for files it returns is XML
  - We will take a look at this in a minute...
The Wiki API

• Why would we want to use the wikipedia API?

• We can use the API to pull up specific pages or search for different pages
Many actions to choose from: `opensearch` and `query`

What action do we want to perform: `action=opensearch`

Example:

```
```
Formatting

• Each option separated by a ‘&’

• We want to return XML files: format=xml

• Example:

• What exactly we are searching for using opensearch

• We want to search for: search=Computer

• Example:

Formatting

• Setting a limit to the search results returned

• We want to limits results: limit=5

• Example:
  
Let’s take a look at what this returns

Example:

Many More

• There are quite a few more options available

• Remember, only submit the requests in serial, unless you want to be blocked by the admins without warning

• Moving forward to another topic (or is it?)
### Flat document file

<table>
<thead>
<tr>
<th>Name</th>
<th>Last Name</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arya</td>
<td>Stark</td>
<td>4ft 3in</td>
</tr>
<tr>
<td>Jon</td>
<td>Snow</td>
<td>6ft 3in</td>
</tr>
<tr>
<td>Margaery</td>
<td>Tyrell</td>
<td>5ft 10in</td>
</tr>
</tbody>
</table>

- We have some flat file document containing names and heights, like the website
- We can see that each line contains:
  - `firstName, lastName, height`
How might we read this?

<table>
<thead>
<tr>
<th>Name</th>
<th>Line</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arya, Stark</td>
<td>Arya, Stark</td>
<td>4ft 3in</td>
</tr>
<tr>
<td>Jon, Snow</td>
<td>Jon, Snow</td>
<td>6ft 3in</td>
</tr>
<tr>
<td>Margaery, Tyrell</td>
<td>Margaery, Tyrell</td>
<td>5ft 10in</td>
</tr>
</tbody>
</table>

- A programmer might loop through line by line, splitting on the “,”:

  - **firstName = First Token**
  - **lastName = Second Token**
  - **Height = Third Token**
This can cause problems…

Arya, Stark, 4ft 3in
Jon, Snow, 6ft 3in
Margaery, Tyrell, 5ft 10in
Ben, John, McCamish, 5ft 11in

• A programmer might loop through line by line, splitting on the “,”:
  ✦ firstName = First Token
  ✦ lastName = Second Token
  ✦ Height = Third Token

• Adding this middle name ruins the tokens
This can be solved

• We already know of one way to solve this
  ★ Relational Databases, any others?

• The internet allows for the transfer of lots of data though, which can make transferring databases harder
  ★ Why?
Another Problem

• We have a schema X to store our data
  
  firstName, lastName, height

• We want to exchange our data with a website in Europe using a different schema
  
  name, meters, centimeters
How can we solve this?

- We could force them to change their schema
  - ★ What if there are more than two schemas?

- They can just export data to us
  - ✦ We would need a database admin to read it

- What if we knew the meaning of each element in the database and could parse through it?
This is XML

• XML (eXtensible Markup Language) does this

• Tree like structure, similar to HTML, except it has customizable tags and attributes

• You can create a standard way of storing and transferring data

✦ No need to change existing schemas or databases
✦ Easy to extend and modify
Joke Time!

• What is the object oriented way to become wealthy?
Joke Time!

- What is the object oriented way to become wealthy?
  - Inheritance!
<people>
  <person>
    <firstname>Arya</firstname>  <lastname>Stark</lastname>  
    <height>4ft 3in</height> 
  </person>
  <person>
    <firstname>Jon</firstname>  <lastname>Snow</lastname>  
    <height>6ft 3in</height> 
  </person>
  <person>
    <firstname>Margaery</firstname>  <lastname>Tyrell</lastname>  
    <height>5ft 10in</height> 
  </person>
</people>
<people>
  <person>
    <firstname>Arya</firstname>
    <lastname>Stark</lastname>
    <height>4ft 3in</height>
  </person>
  <person>
    <firstname>Jon</firstname>
    <lastname>Snow</lastname>
    <height>6ft 3in</height>
  </person>
  <person>
    <firstname>Margaery</firstname>
    <lastname>Tyrell</lastname>
    <height>5ft 10in</height>
  </person>
</people>
<people>
  <person>
    <firstname>Arya</firstname> <lastname>Stark</lastname>
    <height>4ft 3in</height>
  </person>
  <person>
    <firstname Jon</firstname> <lastname> Snow</lastname>
    <height>6ft 3in</height>
  </person>
  <person>
    <firstname>Margaery</firstname> <lastname> Tyrell</lastname>
    <height>5ft 10in</height>
  </person>
</people>
<people>

<person>
  <firstname>Arya</firstname>  <lastname>Stark</lastname>
  <feet>4</feet>  <inches>3</inches>
</person>

<person>
  <firstname>Jon</firstname>  <lastname>Snow</lastname>
  <feet>6</feet>  <inches>3</inches>
</person>

<person>
  <firstname>Margaery</firstname>  <lastname>Tyrell</lastname>
  <feet>5</feet>  <inches>10</inches>
</person>

</people>
<people>
  <person>Arya, Stark, 4ft, 3in</person>
  <person>Jon, Snow, 6ft, 3in</person>
  <person>Margaery, Tyrell, 5ft, 10in</person>
</people>

- This one is pretty much the same as the flat file
  - The point is though that it can be used to format the data however you want
  - This is perfect for transferring information in the web without concern for schema

- iTunes, Eclipse, and many more use XML
Customization

- XML is highly customizable beyond simple tags

<people>
  <person>
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
Customization

• XML is highly customizable beyond simple tags

```xml
<people>
  <person house="stark">
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
```
Customization

• XML is highly customizable beyond simple tags

```xml
<people>
  <person house="stark">
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
```
Customization

- An attribute has its value in double or single quotes

```xml
<people>
  <person house="stark">
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
```
Customization

• Elements can contain just about anything

• Can be named most anything as well

<people>
  <person house="stark">
    <firstname>Arya</firstname>
    <lastname>Stark</lastname>
    <feet>4</feet>
    <inches>3</inches>
  </person>
</people>

• What is a good format? Up to you, but let’s look at some good and bad examples
Customization

Not a good idea…

<people>
  <person house="stark" firstname="Arya"
    lastname="Stark" feet="4" inches="3">
  </person>
</people>

- Messy
- Not easy to expand or change
- No tree structure, not taking advantage of nesting
Customization

Getting better

<people>
  <person house="stark">
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>

• There is some nesting

• Does house="stark" have to be an attribute?

• How might we improve this even more?
Customization

Looks much better!

```xml
<people>
    <person>
        <house> stark </house>
        <firstname> Arya </firstname>
        <lastname> Stark </lastname>
        <height>
            <feet> 4 </feet>
            <inches> 3 </inches>
        </height>
    </person>
</people>
```

- When might you want to use attributes?
Customization

Using both Attributes and Elements

<people>
  <person characterID="539">
    <house> stark </house>
    <firstname>Arya</firstname>
    <lastname>Stark</lastname>
    <height>
      <feet>4</feet>
      <inches>3</inches>
    </height>
  </person>
</people>

• One idea might be for meta data
XPath

• XPath can be used to access information out of XML files

• What other system resembles the tree like structure of the XML file?
XPath

- XPath can be used to access information out of XML files

- What other system resembles the tree like structure of the XML file?
  - Directories on your computer!
XPath

• Suppose I want the names of all the people

• `/people/person/firstname/text()`

• This would return the first names of all the people in the format of text.

```
<people>
  <person>
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
```
XPath

• Suppose I want to get only the first person’s name

• /people/person[1]/firstname/text()
XPath

• You can also put on restrictions

• Only people at least 3 feet tall, their first names

• /people/person[feet >= 3]/firstname/text()

```
<people>
  <person>
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
```
XPath

- `//` selects any node matching the specifications
- `//person` will collect all the person elements
- `../` goes up a node

```xml
<people>
  <person>
    <firstname> Arya </firstname>
    <lastname> Stark </lastname>
    <feet> 4 </feet>
    <inches> 3 </inches>
  </person>
</people>
```
JSON

- **JavaScript Object Notation (JSON)**
- Easier compared to XML to read
- Often it is shorter to write
- Used more often than XML
Objects

• JSON contains objects
  ‣ Denoted by curly brackets “{“ and “}”

{“studentName”: “Bob”}
Strings

• Variable names are in quotes, so are strings

{"studentName": "Bob"}
Numbers

• Floats and Ints can be stored
  ‣ They do not use quotes for the value

{“studentAge”:105}
JSON Example

• Variables can also contain arrays
  ‣ Arrays can contain strings, objects, ints, floats, etc

• Denoted with square brackets “[“ and “]”

```json
{"people": [
    {"firstname":"Arya", "height":"4ft 3in"},
    {"firstname":"Jon", "height":"6ft 3in"},
    {"firstname":"Margaery", "height":"5ft 10in"},
    ]
}
```
We can retrieve values by indexing

Assume that this entire JSON file which is an object is just called ‘Starks’

To find ‘Arya’ we can enter:

```
Starks
```

```json
{"people": [
    {"firstname": "Arya", "height": "4ft 3in"},
    {"firstname": "Jon", "height": "6ft 3in"},
    {"firstname": "Margaery", "height": "5ft 10in"}
]}
```
JSON Example

• We can retrieve values by indexing
  ‣ Assume that this entire JSON file which is an object is just called ‘Starks’

• To find ‘Arya’ we can enter:
  ‣ Starks.people

```json
{"people": [  {"firstname": "Arya", "height": "4ft 3in"},  {"firstname": "Jon", "height": "6ft 3in"},  {"firstname": "Margaery", "height": "5ft 10in"}]
```
JSON Example

• We can retrieve values by indexing
  ‣ Assume that this entire JSON file which is an object is just called ‘Starks’

• To find ‘Arya’ we can enter:
  ‣ Starks.people[0]

```json
{
  "people": [
    {
      "firstname": "Arya", "height": "4ft 3in"
    },
    {
      "firstname": "Jon", "height": "6ft 3in"
    },
    {
      "firstname": "Margaery", "height": "5ft 10in"
    }
  ]
}
```
JSON Example

• We can retrieve values by indexing
  ‣ Assume that this entire JSON file which is an object is just called ‘Starks’

• To find ‘Arya’ we can enter:
  ‣ Starks.people[0].firstname

```json
{"people": [  {"firstname":"Arya", "height":"4ft 3in"},  {"firstname":"Jon", "height":"6ft 3in"},  {"firstname":"Margaery", "height":"5ft 10in"}, ]}
```
Let's find another

How can we find the height of Jon?

```json
{"people": [
  {
    "firstname": "Arya",  "height": "4ft 3in"},
  {
    "firstname": "Jon",  "height": "6ft 3in"},
  {
    "firstname": "Margaery",  "height": "5ft 10in"}
]}
```
JSON Example

- We can retrieve values by indexing
  - Assume that this entire JSON file which is an object is just called ‘Starks’

- To find the height of Jon we can enter:
  - Starks

```
{"people": [
  {"firstname": "Arya", "height": "4ft 3in"},
  {"firstname": "Jon", "height": "6ft 3in"},
  {"firstname": "Margaery", "height": "5ft 10in"}
]}
```
JSON Example

• We can retrieve values by indexing
  ‣ Assume that this entire JSON file which is an object is just called ‘Starks’

• To find the height of Jon we can enter:
  ‣ Starks.people

```json
{"people": [  {"firstname": "Arya", "height": "4ft 3in"},  {"firstname": "Jon", "height": "6ft 3in"},  {"firstname": "Margaery", "height": "5ft 10in"} ]}
```
JSON Example

- We can retrieve values by indexing
  - Assume that this entire JSON file which is an object is just called ‘Starks’

- To find the height of Jon we can enter:
  - Starks.people[1]

```json
{"people": [  {"firstname":"Arya", "height":"4ft 3in"},  {"firstname":"Jon", "height":"6ft 3in"},  {"firstname":"Margaery", "height":"5ft 10in"}, ] }
```
JSON Example

• We can retrieve values by indexing
  ‣ Assume that this entire JSON file which is an object is just called ‘Starks’

• To find the height of Jon we can enter:
  ‣ Starks.people[1].height
  ‣ In JavaScript, you can do Starks.people[1][“height”] as well

```json
{"people": [
  {"firstname":"Arya", "height":"4ft 3in"},
  {"firstname":"Jon", "height":"6ft 3in"},
  {"firstname":"Margaery", "height":"5ft 10in"},
]}
```
JSON and JavaScript

• It is quite easy to use these JSON files as objects

• You can loop through the arrays like any other java array object

• You can index into objects like a dictionary when using JavaScript