Module 11
Tree-Structured data
CS 106 Winter 2020
Some data is **hierarchical**: we think of each part ("node") as "owning" or "enclosing" some sub-parts, down to some base level.
<table>
<thead>
<tr>
<th>Library of Congress Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong> General Works</td>
</tr>
<tr>
<td>AE Encyclopedias</td>
</tr>
<tr>
<td>AY Almanacs</td>
</tr>
<tr>
<td><strong>B-BJ Philosophy</strong></td>
</tr>
<tr>
<td>BF Psychology</td>
</tr>
<tr>
<td>BL-BX Religion</td>
</tr>
<tr>
<td><strong>C</strong> History</td>
</tr>
<tr>
<td>CB History of Civilization</td>
</tr>
<tr>
<td>CC Archaeology</td>
</tr>
<tr>
<td>CT General Biography</td>
</tr>
<tr>
<td><strong>D</strong> History</td>
</tr>
<tr>
<td>DA-DQ</td>
</tr>
<tr>
<td>DK Russian History</td>
</tr>
<tr>
<td>DS-DT</td>
</tr>
<tr>
<td><strong>E</strong> U.S. History</td>
</tr>
<tr>
<td>E186 Colonial History</td>
</tr>
<tr>
<td>E456 Civil War</td>
</tr>
<tr>
<td>E740 Twentieth Century</td>
</tr>
<tr>
<td><strong>F</strong> History of the Americas</td>
</tr>
<tr>
<td>F1 State Histories</td>
</tr>
<tr>
<td>F381 Texas</td>
</tr>
<tr>
<td>F1001 Canada</td>
</tr>
<tr>
<td>F1201 Mexico, Latin America</td>
</tr>
<tr>
<td><strong>G</strong> Geography</td>
</tr>
<tr>
<td><strong>N</strong> Fine Arts</td>
</tr>
<tr>
<td>NA-NB Architecture, Sculpture</td>
</tr>
<tr>
<td>NC-NE Drawing, Painting, Prints</td>
</tr>
<tr>
<td>NK Crafts</td>
</tr>
<tr>
<td><strong>P</strong> Language and Literature</td>
</tr>
<tr>
<td>PA Classical Language, Literature</td>
</tr>
<tr>
<td>PC2001 French Language</td>
</tr>
<tr>
<td>PC4001 Spanish Language</td>
</tr>
<tr>
<td>PE English Language</td>
</tr>
<tr>
<td>PE1128 English as a Second Language</td>
</tr>
<tr>
<td>PF German Language</td>
</tr>
<tr>
<td>PL Japanese, Korean, Chinese Languages</td>
</tr>
<tr>
<td>PN Poetry, Theater, Speech, Journalism</td>
</tr>
<tr>
<td>PQ1 French Literature</td>
</tr>
<tr>
<td>PQ6001 Spanish Literature</td>
</tr>
<tr>
<td>PR British Literature</td>
</tr>
<tr>
<td>PS American Literature</td>
</tr>
<tr>
<td>PT German Literature</td>
</tr>
<tr>
<td>PZ Children's, Young Adult Literature</td>
</tr>
<tr>
<td><strong>Q</strong> Science</td>
</tr>
<tr>
<td>QA Mathematics</td>
</tr>
<tr>
<td>QB Astronomy</td>
</tr>
<tr>
<td>QC Physics</td>
</tr>
<tr>
<td>QD Chemistry</td>
</tr>
<tr>
<td>QE Geology</td>
</tr>
<tr>
<td>OH Natural History</td>
</tr>
</tbody>
</table>

**QD** | QA76 | Computer Science |
Sometimes, a node behaves like a **set of attributes**: it has a specific slot set aside for each kind of attribute.
Attributes can have sub-attributes and so on.
Sometimes, a node holds something more like a **sequence** of children.
Sometimes, a node holds something more like a **sequence** of children.
There are two standard ways that tree-structured data is passed around online:

- **XML**: eXtended Markup Language
- **JSON**: JavaScript Object Notation

Both are “simple” text-based formats for more or less arbitrary data.

Both are accommodated for in the p5 library. We’ll use JSON because it’s nicer to read.
**JSON objects**

A **JSON Object** is a comma-separated list of key:value pairs, enclosed in curly braces. It behaves like a dictionary! It maps string keys to arbitrary values.

```json
{
    "Student ID": 123,
    "Clicker": "78%",
    "Assignments": "90%",
    "Midterm": "91%",
    "Final": "93%",
}
```
JSON objects

The values in a JSON object can be pretty much anything. ints, floats, strings, arrays, arrays of arrays, even other JSON objects!
null
{ "firstName": "John", "lastName": "Smith", "age": 35, "address": { "streetAddress": "51 Strange Street", "city": "Kitchener", "province": "ON", "postalCode": "N3K 1E7" }, "phoneNumbers": [ { "type": "home", "number": "519 555-1234" }, { "type": "mobile", "number": "226 555-4567" } ], "children": [ "Eunice", "Murgatroyd" ], "spouse": null }
Getting JSON Objects

let stuff = loadJSON( "filename.json" );

Read the contents of the file into a JSONObject.
Working with JS Objects

```javascript
let obj = loadJSON("address.json");

... obj.key ...
... obj.fieldname...
... obj.phone ...
... obj.address ...
... obj.whatever[8] ...
... obj.classrooms[4].teacher_name ...
```
obj.firstName;
{ "firstName": "John", "lastName": "Smith", "age": 35, "address": { "streetAddress": "51 Strange Street", "city": "Kitchener", "province": "ON", "postalCode": "N3K 1E7" }, "phoneNumbers": [ { "type": "home", "number": "519 555-1234" }, { "type": "mobile", "number": "226 555-4567" } ], "children": [ "Eunice", "Murgatroyd" ], "spouse": null }
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 35,
    "address": {
        "streetAddress": "51 Strange Street",
        "city": "Kitchener",
        "province": "ON",
        "postalCode": "N3K 1E7"
    },
    "phoneNumbers": [
    {
        "type": "home",
        "number": "519 555-1234"
    },
    {
        "type": "mobile",
        "number": "226 555-4567"
    }]
},
"children": ["Eunice", "Murgatroyd"],
"spouse": null
}
```json
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 35,
    "address": {
        "streetAddress": "51 Strange Street",
        "city": "Kitchener",
        "province": "ON",
        "postalCode": "N3K 1E7"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "519 555-1234"
        },
        {
            "type": "mobile",
            "number": "226 555-4567"
        }
    ],
    "children": ["Eunice", "Murgatroyd"],
    "spouse": null
}
```
{  
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type": "mobile",
      "number": "226 555-4567"
    }
  ],
  "children": ["Eunice", "Murgatroyd"],
  "spouse": null
}
```json
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 35,
    "address": {
        "streetAddress": "51 Strange Street",
        "city": "Kitchener",
        "province": "ON",
        "postalCode": "N3K 1E7"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "519 555-1234"
        },
        {
            "type": "mobile",
            "number": "226 555-4567"
        }],
    "children": ["Eunice", "Murgatroyd"],
    "spouse": null
}
```
```json
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type": "mobile",
      "number": "226 555-4567"
    }
  ],
  "children": ["Eunice", "Murgatroyd"],
  "spouse": null
}
```
Example: counting files
Example: RSS Feeds

A podcast is rich media, such as audio or video, distributed via RSS. Feeds like this one provide updates whenever there is new content. FeedBurner makes it easy to receive content updates in popular podcatchers.

Learn more about syndication and FeedBurner...

See: rss2json.com

feeds.wnyc.org/radiolab
  <channel>
    <title>Radiolab</title>
    <link>http://www.radiolab.org/series/podcasts/</link>
    <description>Radiolab is a show about curiosity. Where sound illuminates ideas, and the boundaries blur between science, technology, philosophy, and education. Radiolab is heard around the country on more than 500 NPR member stations. Check your local station for airtimes.</description>
    <language>en-us</language>
    <lastBuildDate>Fri, 24 Mar 2017 01:00:00 -0400</lastBuildDate>
    <ttl>600</ttl>
    <itunes:explicit>no</itunes:explicit>
    <atom10:link xmlns:atom10="http://www.w3.org/2005/Atom" rel="self" type="application/rss+xml"/>
    <feedburner:info xmlns:feedburner="http://rssnamespace.org/feedburner/ext/1.0" uri="radiolab.wnycstudios.com/feeds/wnyc%3Aradiolab.rss"/>
    <media:copyright>© WNYC Studios</media:copyright>
    <media:thumbnail url="https://media2.wnyc.org/i/1400/1400/l/80/1/Radiolab-wnycstudios.jpg"/>
    <media:keywords>Science, Technology, Philosophy, Education, radiolab, jad, abumrad, krulwich, Radio, Lab, wnyc, studios</media:keywords>
    <media:category scheme="http://www.itunes.com/dtds/podcast-1.0.dtd">Science &amp; Medicine/Natural Sciences</media:category>
    <media:category scheme="http://www.itunes.com/dtds/podcast-1.0.dtd">Society &amp; Culture</media:category>
    <media:category scheme="http://www.itunes.com/dtds/podcast-1.0.dtd">Education</media:category>
    <itunes:author>WNYC Studios</itunes:author>
    <itunes:keywords>Science, Technology, Philosophy, Education, radiolab, jad, abumrad, krulwich, Radio, Lab, wnyc, studios</itunes:keywords>
  </channel>
</rss>
We again join Ben Montgomery, reporter at the Tampa Bay Times, as he looks at data on every person...
Going live

All load functions accept URLs as parameters in addition to file names!

loadStrings()
loadImage()
loadTable()
loadJSON()
Functions like loadStrings() and loadImage() allow you to access fixed content over the internet. loadJSON() is more like calling a function over the web.
Welcome to Open Data API

Hello and Heads up! (September 18th, 2017)

Hello!

I wanted to let you know that effective immediately the Open Data API project at Waterloo is under a new team. We're looking forward to understanding what exists now, getting feedback from current users, and having a clear plan to communicate before moving forward. We’d like to continue to build on the great work done by those before us that made this project possible.

That said, we think it's safe to say that for the immediate future we will be doing essentially no new work on this platform. We’ll fix only critical issues and work on a plan going forward. This blog will be used to communicate the motivation for changes, any new resources and practices, and the ongoing work going forward. We hope you can bear with us during the transition.
Example: classrooms

The UW API supports requests like “what courses are scheduled in a given classroom?”

GET /buildings/{building}/{room}/courses.{format}

https://api.uwaterloo.ca/v2/buildings/STC/0040/courses.json
Most online APIs require you to register for a key.
The Google APIs

Google offers dozens of APIs for web designers and developers.

Some are specifically related to popular Google products, like Gmail and Analytics, while others are more specialized and aren’t part of public programs.

All are free to use, of course. You can view all of Google’s APIs and code tools on their site directory.

- **Feed API** - The Google Feed API lets you download any public feed (including RSS, Media RSS, and Atom) and then combine them into mashups. It simplifies the mashup process by using JavaScript rather than more complex server-side coding.

- **Places API** - Google Places is a large directory of local businesses and attractions all

  website, as well as display check-ins by users.
Example: Square vs Circles

Game Description:
1. Game generates a circle and a square per frame using random function.
2. If the circle is generated on the left half of the screen, the score of circles is incremented by 1.
3. If the square is generated on the right half of the screen, the score of squares is incremented by 1.
4. Scores and the current winner is displayed on the screen.
5. User can control the speed of the game through a slider.
6. Both scores (circle and square) and the fps of the game are loaded from a JSON.
7. User can save the score and settings in a JSON file by pressing the save button.
8. User can copy and paste the saved json to the data folder to start the game from the saved point onwards.