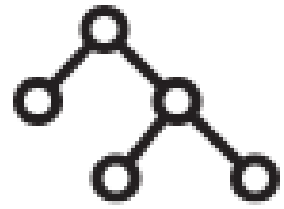




Module 11

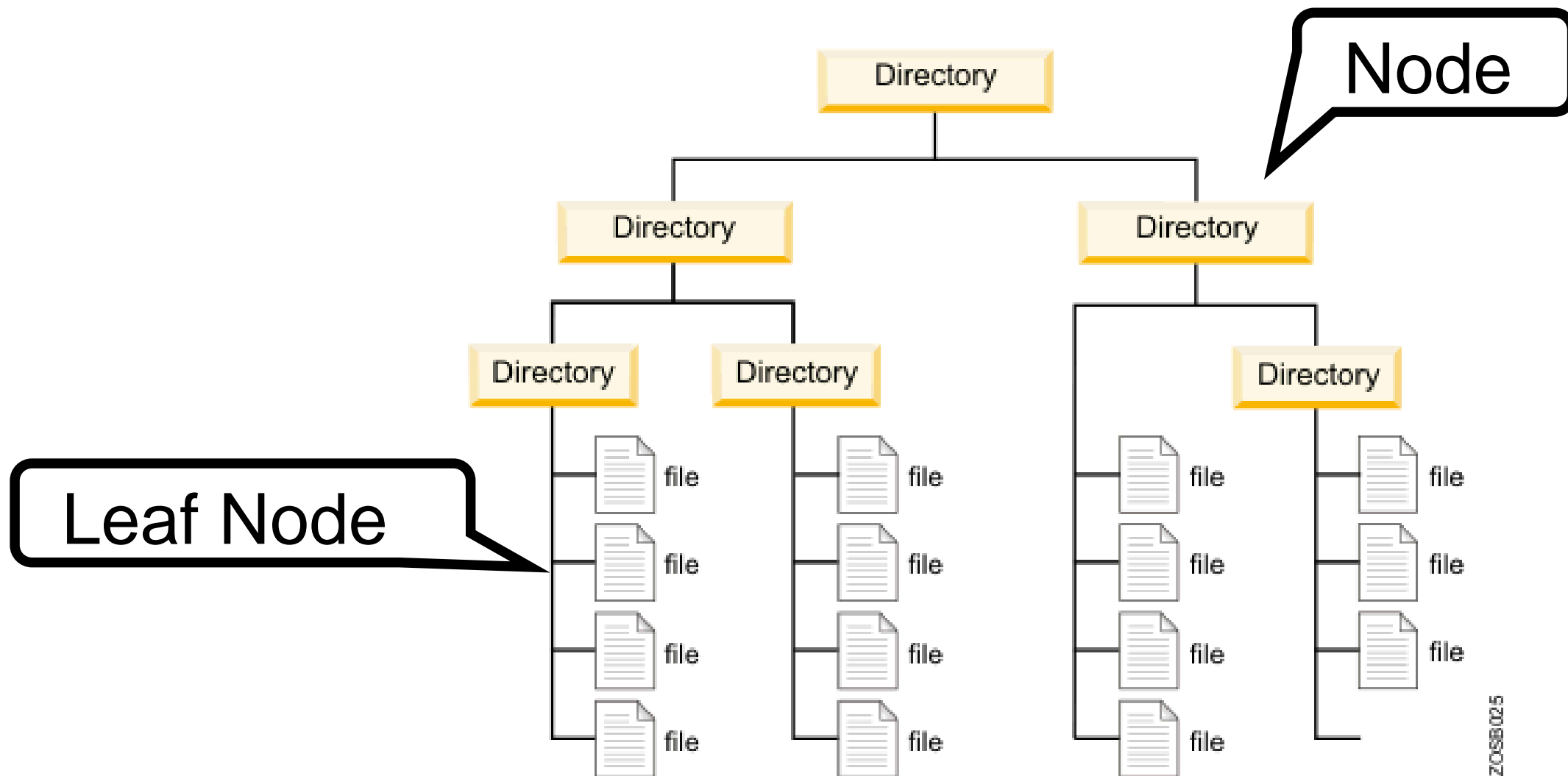
Tree-Structured data

CS 106 Winter 2020

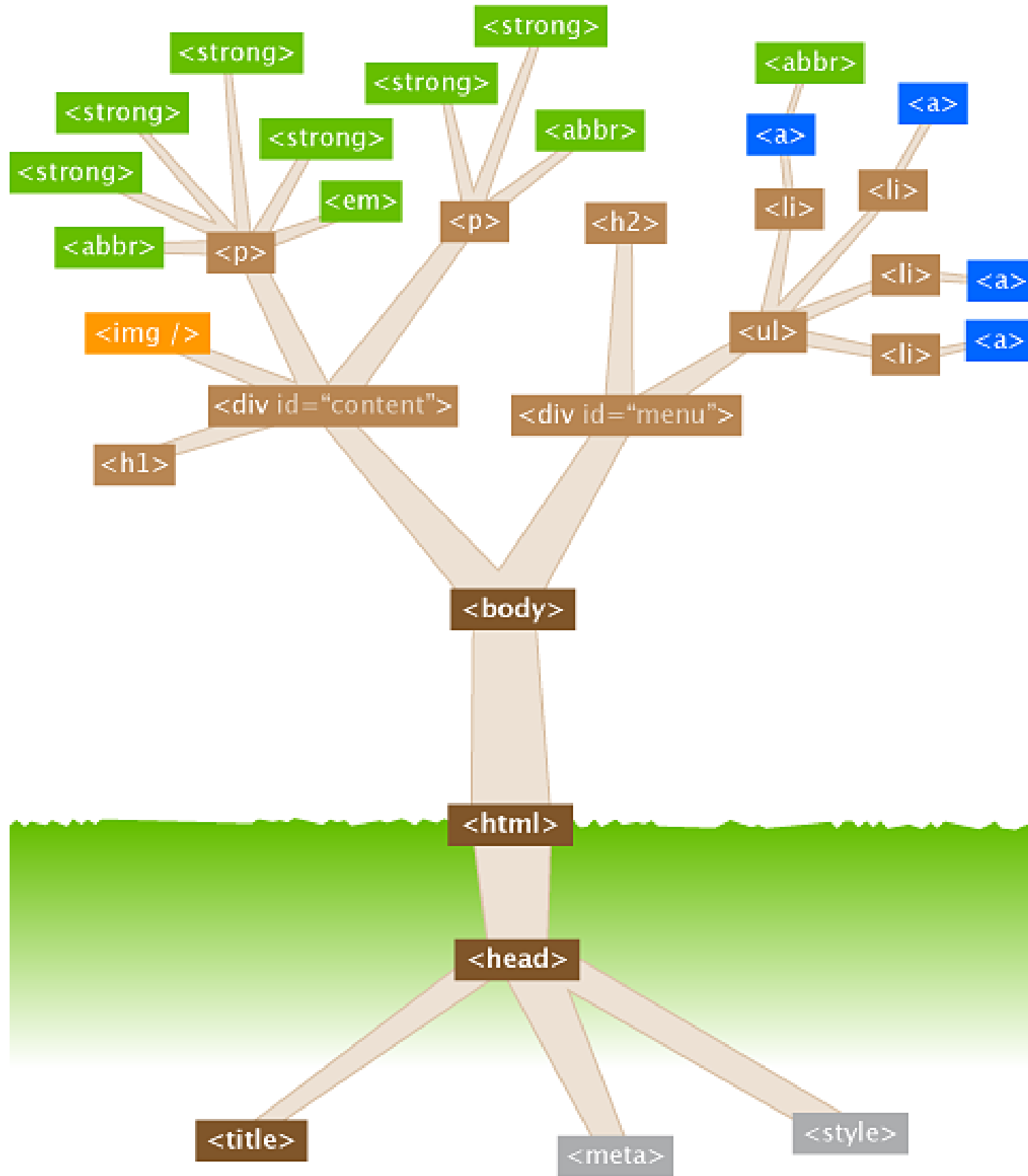


Trees

Some data is **hierarchical**: we think of each part (“node”) as “owning” or “enclosing” some sub-parts, down to some base level.







LIBRARY OF CONGRESS CLASSIFICATION

A GENERAL WORKS

AE Encyclopedias
AY Almanacs

B-BJ PHILOSOPHY

BF Psychology
BL-BX Religion

C HISTORY

CB History of Civilization
CC Archaeology
CT General Biography

D HISTORY

DA-DQ
DK Russian History
DS-DT

E U.S. HISTORY

E186 Colonial History
E456 Civil War
E740 Twentieth Century

F HISTORY OF THE AMERICAS

F1 State Histories
F381 Texas
F1001 Canada
F1201 Mexico. Latin America

G GEOGRAPHY

N FINE ARTS

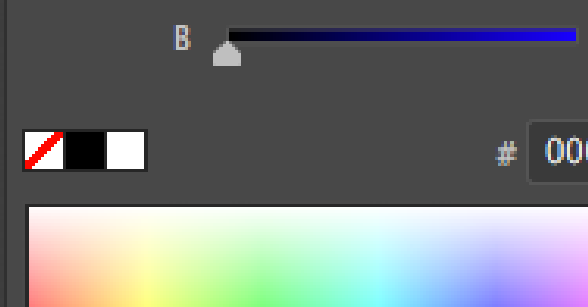
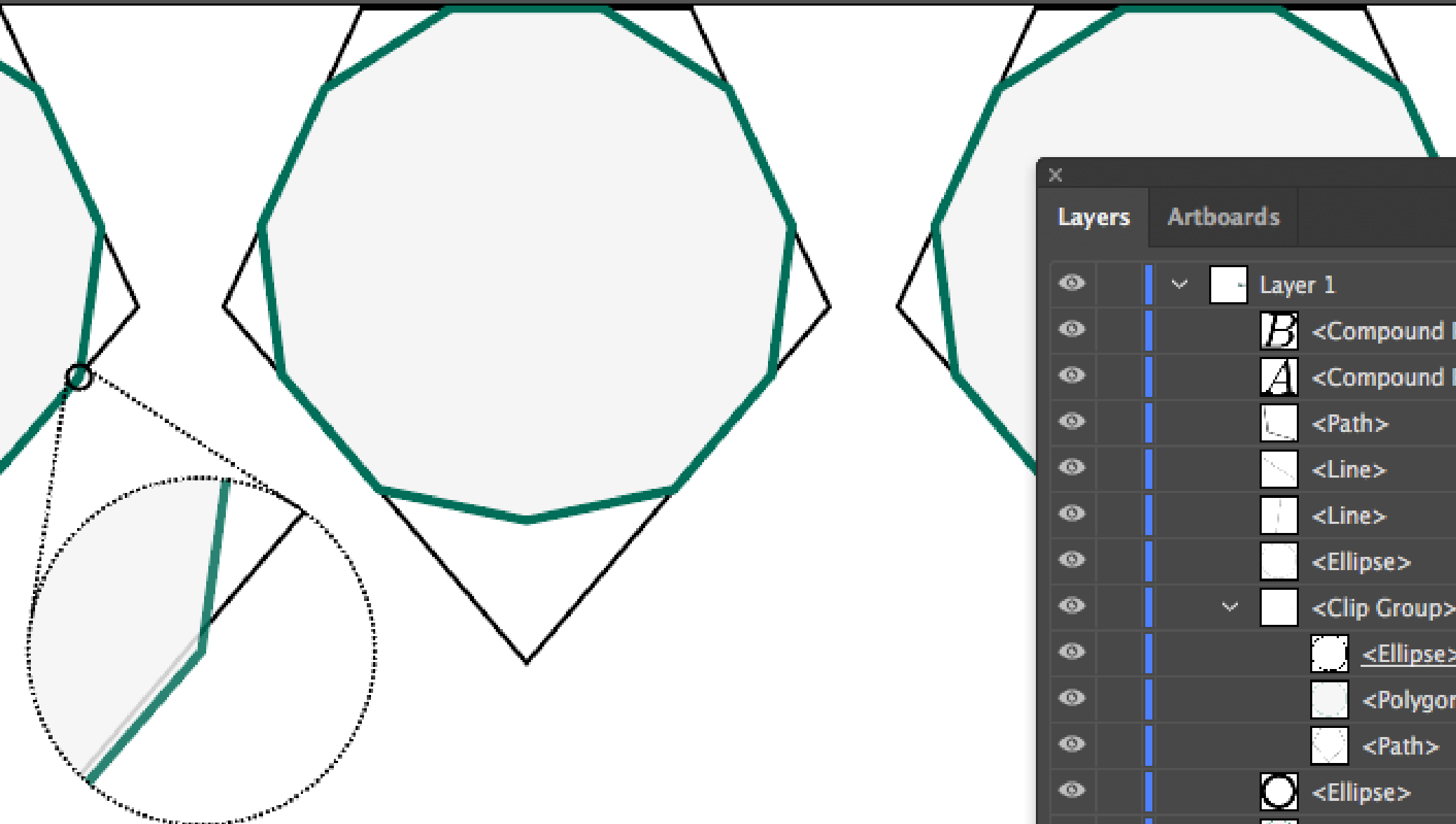
NA-NB Architecture. Sculpture
NC-NE Drawing, Painting, Prints
NK Crafts

P LANGUAGE AND LITERATURE

PA Classical Language, Literature
PC2001 French Language
PC4001 Spanish Language
PE English Language
PE1128 English as a Second Language
PF German Language
PL Japanese. Korean. Chinese Languages
PN Poetry. Theater. Speech. Journalism
PQ1 French Literature
PQ6001 Spanish Literature
PR British Literature
PS American Literature
PT German Literature
PZ Children's, Young Adult Literature

Q SCIENCE

QA Mathematics
QA76 Computer Science
QB Astronomy
QC Physics
QD Chemistry
QE Geology
QH Natural History



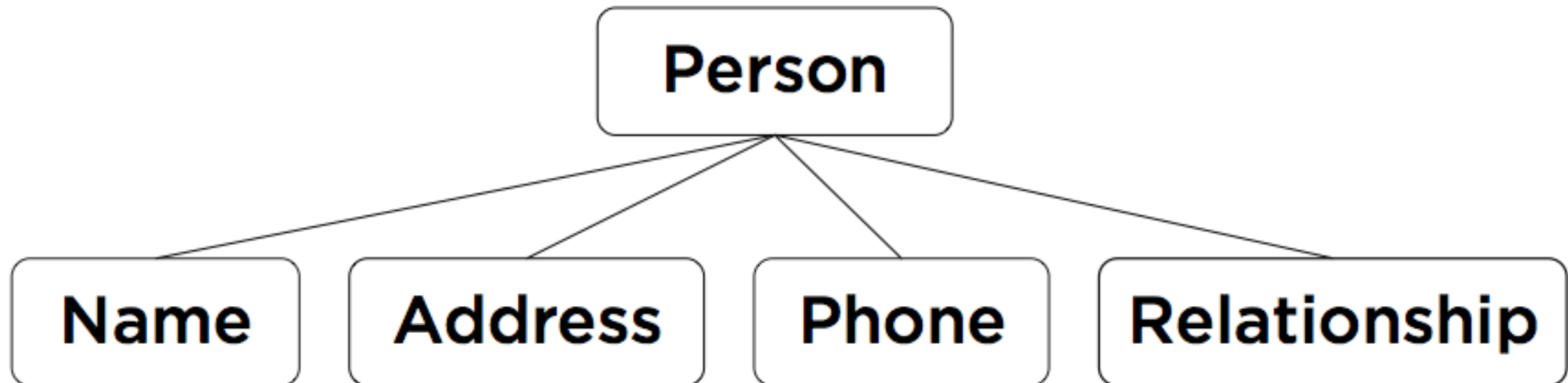
Layers Artboards

Visibility	Layer	Object Name	Lock
<input type="checkbox"/>	Layer 1		<input type="checkbox"/>
<input type="checkbox"/>		<Compound Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Compound Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Line>	<input type="checkbox"/>
<input type="checkbox"/>		<Line>	<input type="checkbox"/>
<input type="checkbox"/>		<Ellipse>	<input type="checkbox"/>
<input type="checkbox"/>		<input type="checkbox"/> <Clip Group>	<input type="checkbox"/>
<input type="checkbox"/>		<Ellipse>	<input type="checkbox"/>
<input type="checkbox"/>		<Polygon>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Ellipse>	<input type="checkbox"/>
<input type="checkbox"/>		<Polygon>	<input type="checkbox"/>
<input type="checkbox"/>		<input checked="" type="checkbox"/> <Group>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>
<input type="checkbox"/>		<input checked="" type="checkbox"/> <Group>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>
<input type="checkbox"/>		<Path>	<input type="checkbox"/>

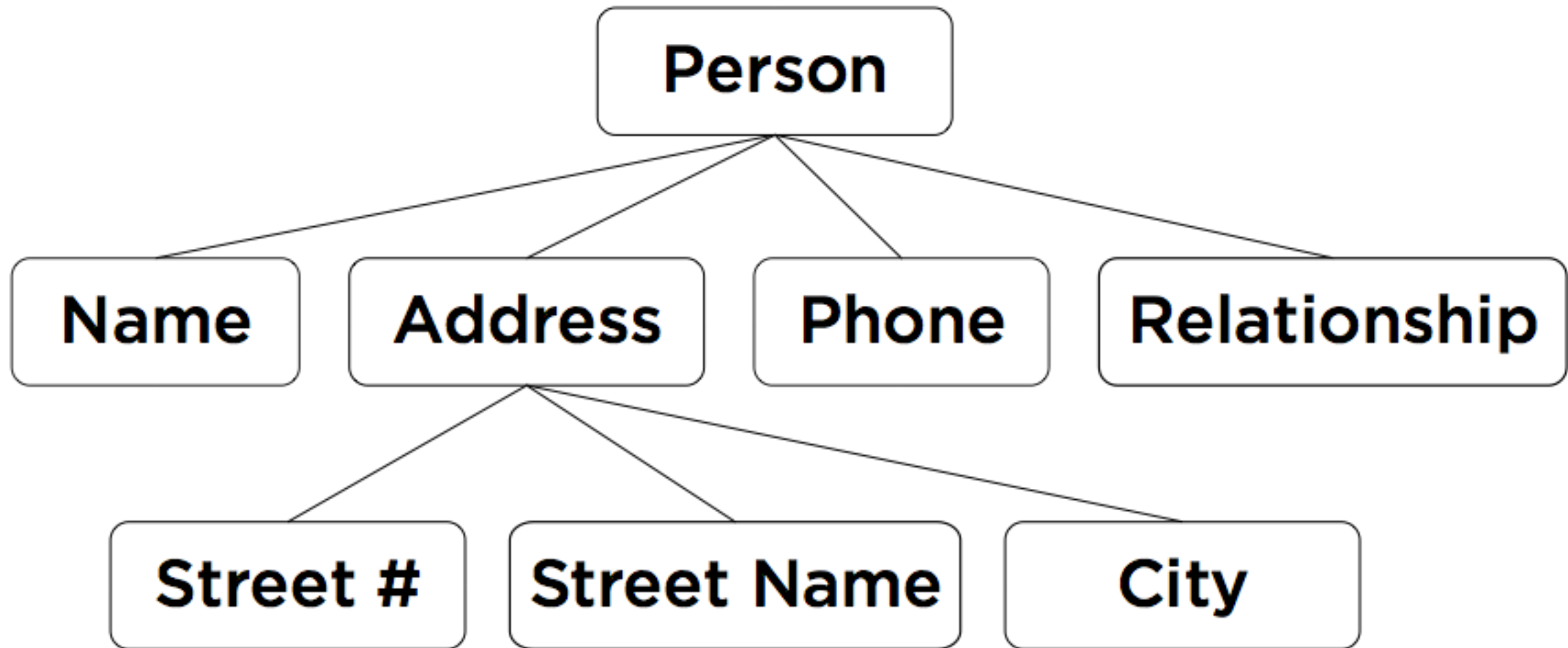
1 Layer

Search Copy Paste Delete

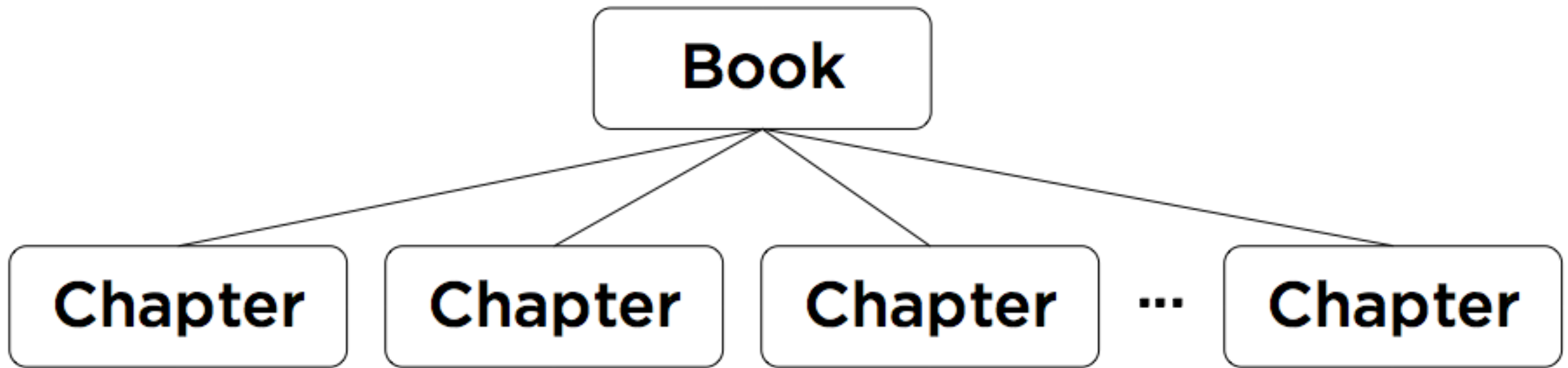
Sometimes, a node behaves like a **set of attributes**: it has a specific slot set aside for each kind of attribute.



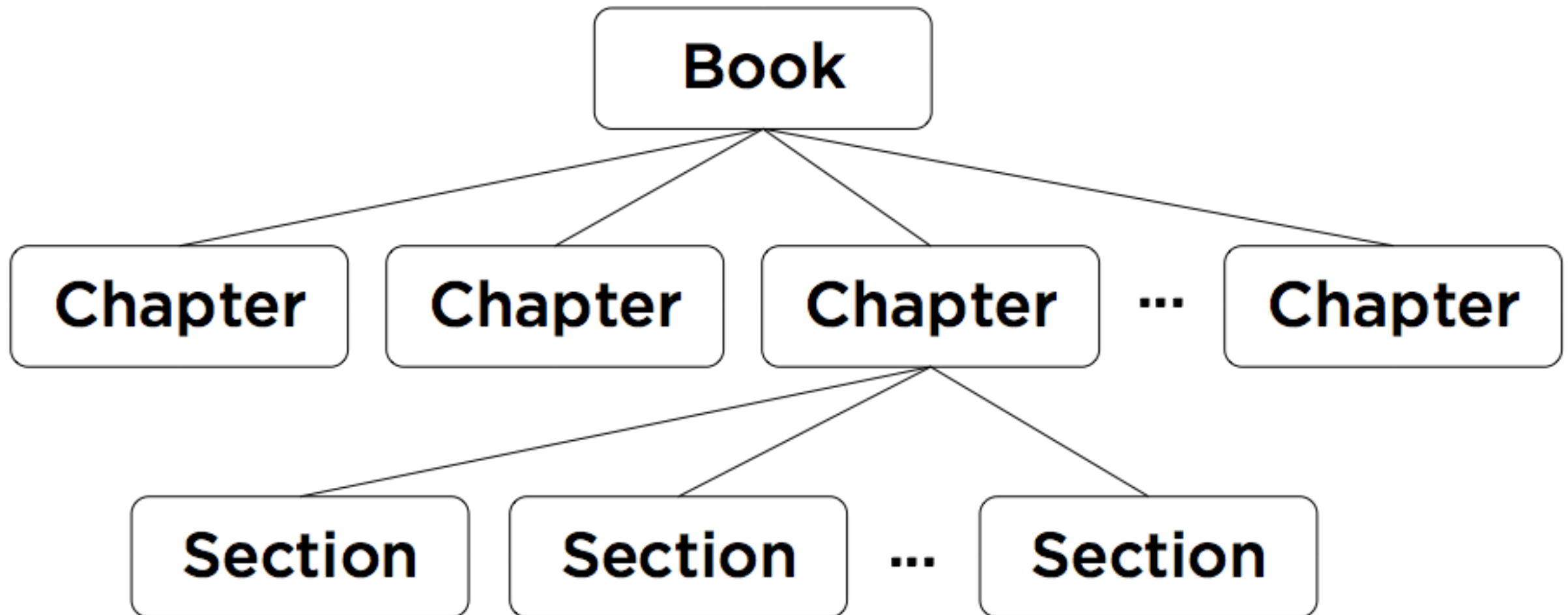
Attributes can have sub-attributes and so on.



Sometimes, a node holds something more like a **sequence** of children.



Sometimes, a node holds something more like a **sequence** of children.



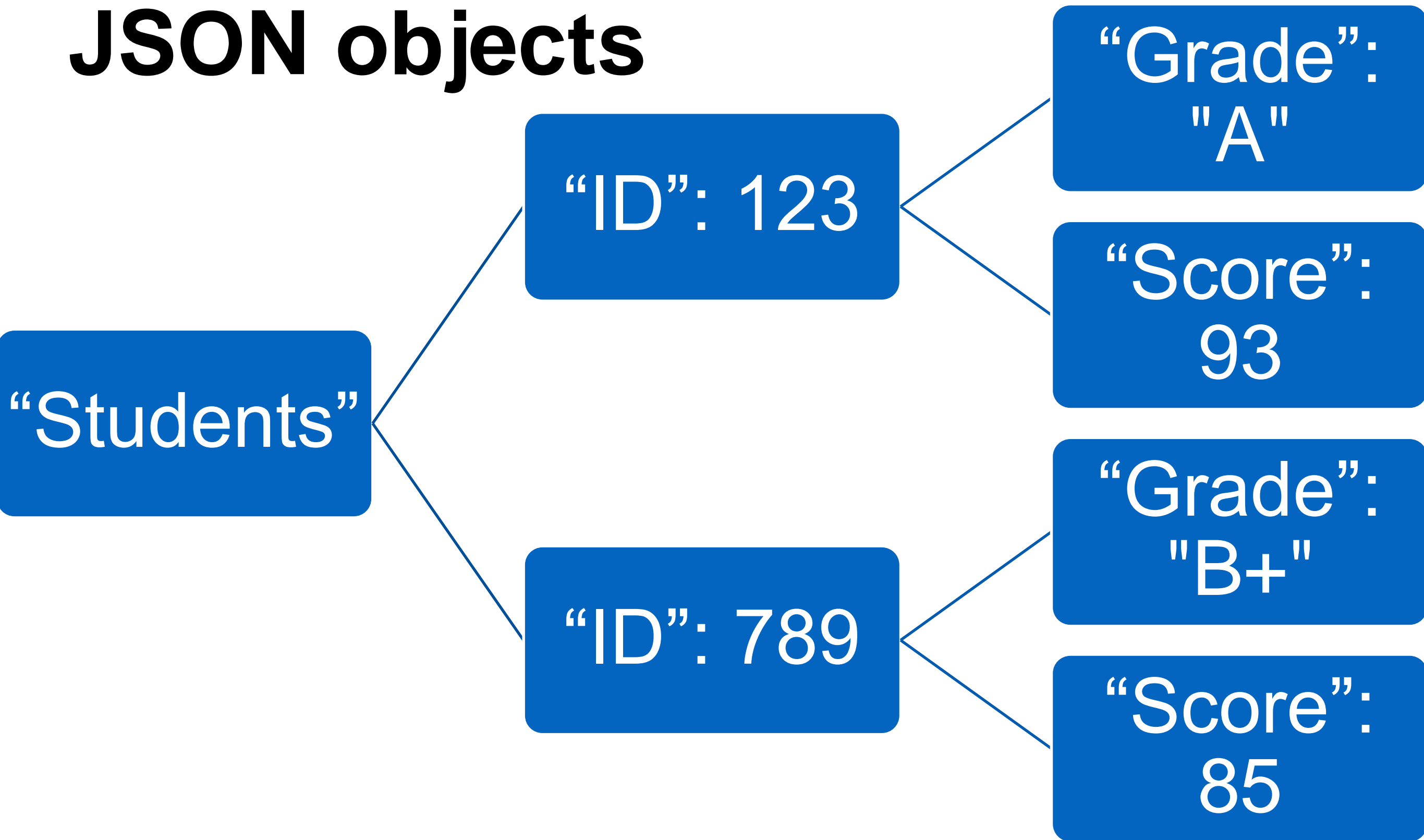
There are two standard ways that tree-structured data is passed around online:

- **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



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.

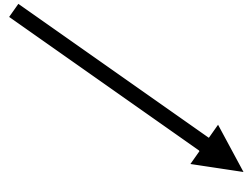
```
{  
  "Student ID": 123,  
  "Clicker": "78%",  
  "Assignments": "90%",  
  "Midterm": "91%",  
  "Final": "93%",  
}
```

JSON objects

The values in a JSON object can be pretty much anything. ints, floats, strings, arrays, arrays of arrays, even other JSON objects!


```
{
  "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
}
```

obj



```
{
  "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

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

```
... obj.key ...
```

```
... obj.fieldname...
```

```
... obj.phone ...
```

```
... obj.address ...
```

```
... obj.whatever[8] ...
```

```
... obj.classrooms[4].teacher_name ...
```



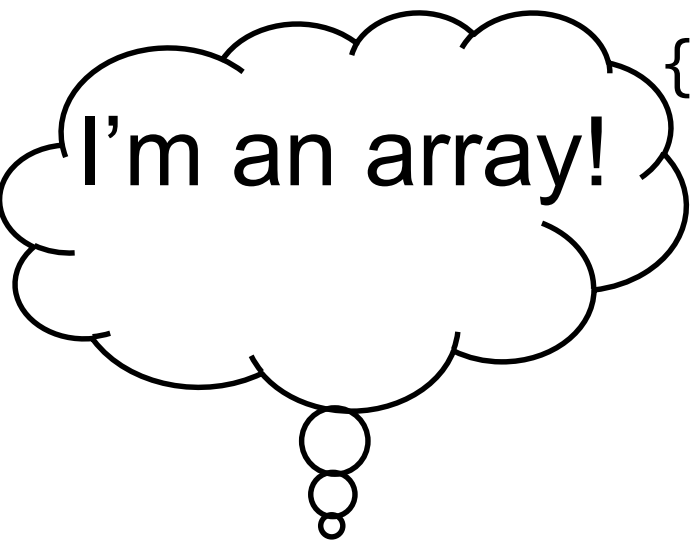
```
obj.firstName;
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "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
}
```

obj.age; ←


```
obj.phoneNumbers;
{
  "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
}
```

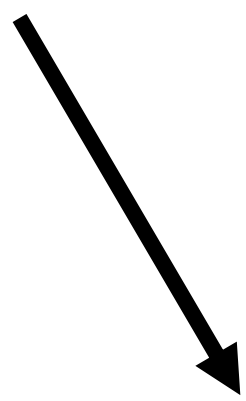




obj.phoneNumbers;

```
{  
  "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  
}
```

A rounded rectangular box highlighting the "phoneNumbers" array in the JSON code above. The array contains two objects: one for a home phone number and one for a mobile phone number.

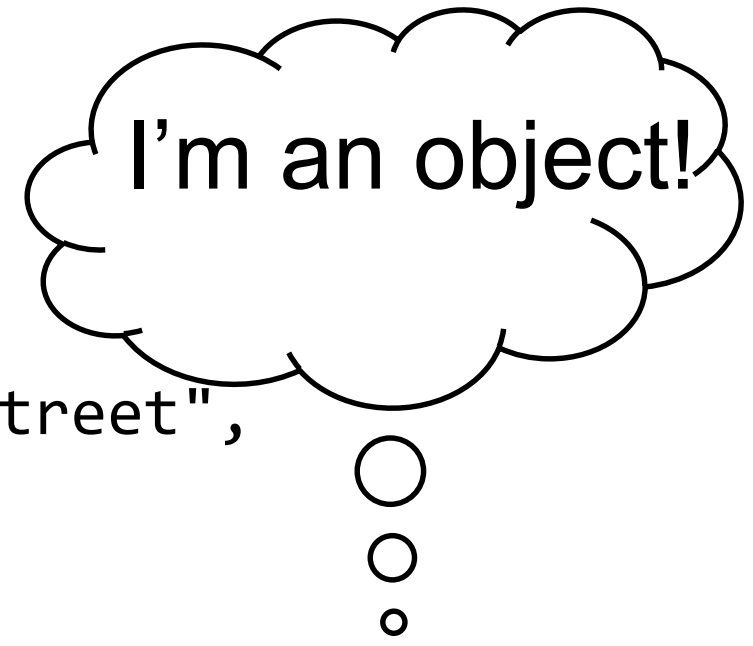


```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type": "mobile",
      "number": "226 555-4567"
    }
  ],
  "children": ["Eunice", "Murgatroyd"],
  "spouse": null
}
```

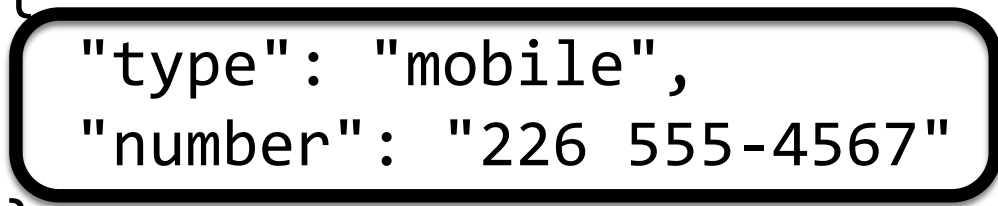
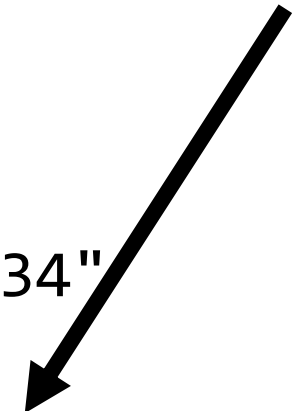
obj.phoneNumbers[1];



```
{
  "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
}
```

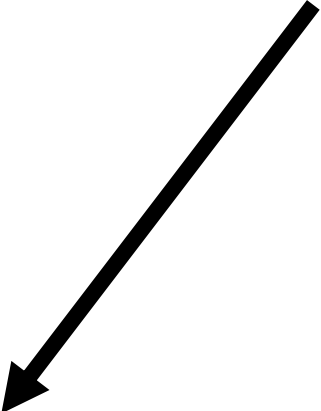


obj.phoneNumbers[1];

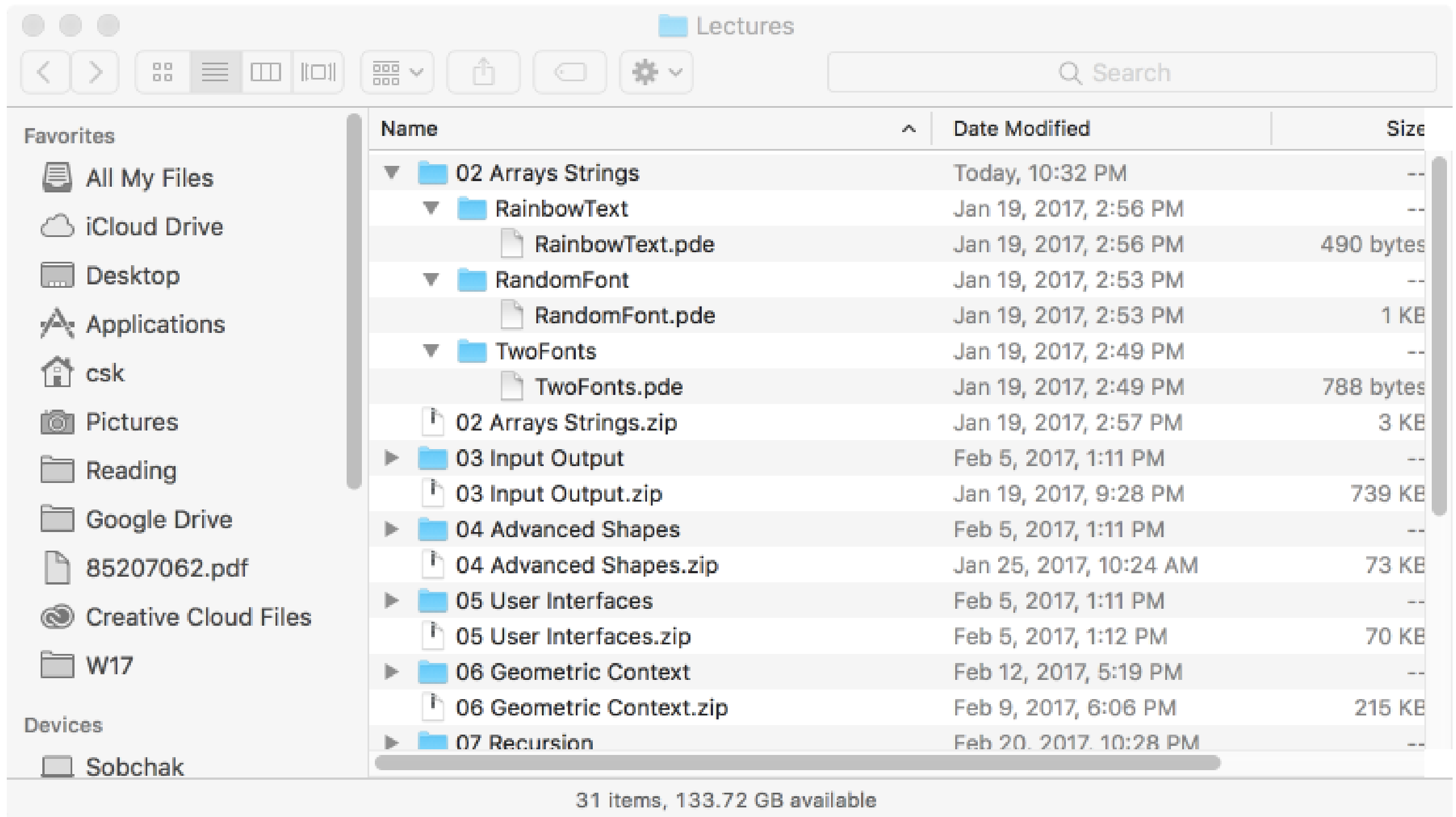


```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type": "mobile",
      "number": "226 555-4567"
    }
  ],
  "children": ["Eunice", "Murgatroyd"],
  "spouse": null
}
```

obj.phoneNumbers[1].number;



Example: counting files



The screenshot shows a macOS Finder window titled "Lectures". The left sidebar displays "Favorites" and "Devices". The main pane shows a list of files and folders with columns for "Name", "Date Modified", and "Size".

Name	Date Modified	Size
02 Arrays Strings	Today, 10:32 PM	--
RainbowText	Jan 19, 2017, 2:56 PM	--
RainbowText.pde	Jan 19, 2017, 2:56 PM	490 bytes
RandomFont	Jan 19, 2017, 2:53 PM	--
RandomFont.pde	Jan 19, 2017, 2:53 PM	1 KB
TwoFonts	Jan 19, 2017, 2:49 PM	--
TwoFonts.pde	Jan 19, 2017, 2:49 PM	788 bytes
02 Arrays Strings.zip	Jan 19, 2017, 2:57 PM	3 KB
03 Input Output	Feb 5, 2017, 1:11 PM	--
03 Input Output.zip	Jan 19, 2017, 9:28 PM	739 KB
04 Advanced Shapes	Feb 5, 2017, 1:11 PM	--
04 Advanced Shapes.zip	Jan 25, 2017, 10:24 AM	73 KB
05 User Interfaces	Feb 5, 2017, 1:11 PM	--
05 User Interfaces.zip	Feb 5, 2017, 1:12 PM	70 KB
06 Geometric Context	Feb 12, 2017, 5:19 PM	--
06 Geometric Context.zip	Feb 9, 2017, 6:06 PM	215 KB
07 Recursion	Feb 20, 2017, 10:28 PM	--

31 items, 133.72 GB available


```
"name": ".",  
"children": [  
  {  
    "type": "file",  
    "name": ".DS_Store"  
  },  
  {  
    "type": "directory",  
    "name": "02 Arrays Strings",  
    "children": [  
      {  
        "type": "directory",  
        "name": "RainbowText",  
        "children": [  
          {  
            "type": "file",  
            "name": "RainbowText.pde"  
          }  
        ]  
      }  
    ]  
  },  
  {  
    "type": "directory",  
    "name": "RandomFont",  
    "children": [  
      {  
        "type": "file"
```



Example: RSS Feeds

Radiolab

A **podcast** powered by FeedBurner

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...](#)

Subscribe Now!

...with web-based podcatchers. Click your choice below:



...with iTunes:

[Add to iTunes](#)

...with something else (copy this address):

Get more info on other podcatchers:



[View Feed XML](#)

See: rss2json.com

feeds.wnyc.org/radiolab

```
<?xml-stylesheet type="text/css" media="screen" href="http://feeds.wnyc.org/~d/styles/itemcon
<rss xmlns:atom="http://www.w3.org/2005/Atom" xmlns:itunes="http://www.itunes.com/dtds/po
<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
Radiolab is heard around the country on more than 500 member stations. Check your local sta
Embed the Radiolab widget on your blog or website.
```

Radiolab is supported, in part, by the Alfred P. Sloan Foundation, enhancing public understand
All press inquiries may be directed to Jennifer Houlihan Roussel at (646) 829-4497.</descriptio

```
<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/rs
<feedburner:info xmlns:feedburner="http://rssnamespace.org/feedburner/ext/1.0" uri="radi
<atom10:link xmlns:atom10="http://www.w3.org/2005/Atom" rel="hub" href="http://pubsub
<media:copyright>© WNYC</media:copyright>
<media:thumbnail url="https://media2.wnyc.org/i/1400/1400/I/80/1/Radiolab-wnycstudios.j
<media:keywords>Science,Technology,Philosophy,Education,radiolab,jad,abumrad,krulw
<media:category scheme="http://www.itunes.com/dtds/podcast-1.0.dtd">Science & M
<media:category scheme="http://www.itunes.com/dtds/podcast-1.0.dtd">Society & C
<media:category scheme="http://www.itunes.com/dtds/podcast-1.0.dtd">Education</medi
<itunes:author>WNYC Studios</itunes:author>
<itunes:image href="https://media2.wnyc.org/i/1400/1400/I/80/1/Radiolab-wnycstudios.jpg
<itunes:keywords>Science,Technology,Philosophy,Education,radiolab,jad,abumrad,krulw
```

stations. Check your local station for airtimes. Embed the Radiolab widget on your blog modern world. More information about Sloan at www.sloan.org. All press inquiries may

```
"image": "https://media2.wnyc.org/i/1400/1400/I/80/1/Radiolab-wnycstudios.jpg"
}
"items": [
  {
    "title": "Shots Fired: Part 2"
    "pubDate": "2017-03-24 05:00:00"
    "link": "http://www.radiolab.org/story/shots-fired-part-2/"
    "guid": "http://www.radiolab.org/story/shots-fired-part-2/"
    "author": "WNYC Studios"
    "thumbnail": "https://media2.wnyc.org/i/130/130/c/80/1/3957814193_6fd835e7c0_o.jpg"
    "description": "We again join Ben Montgomery, reporter at the Tampa Bay Times, as he
    "content": "We again join Ben Montgomery, reporter at the Tampa Bay Times, as he
    "enclosure": {
      "link": "https://www.podtrac.com/pts/redirect.mp3/audio.wnyc.org/rl_extras/rl_extr
      "type": "audio/mpeg"
      "duration": 1766
```

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*.

OPEN DATA API

[Open Data API home](#)[Register for an API key](#)[Contact us](#)

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 api.uwaterloo.ca and fix only critical issues. The motivation for

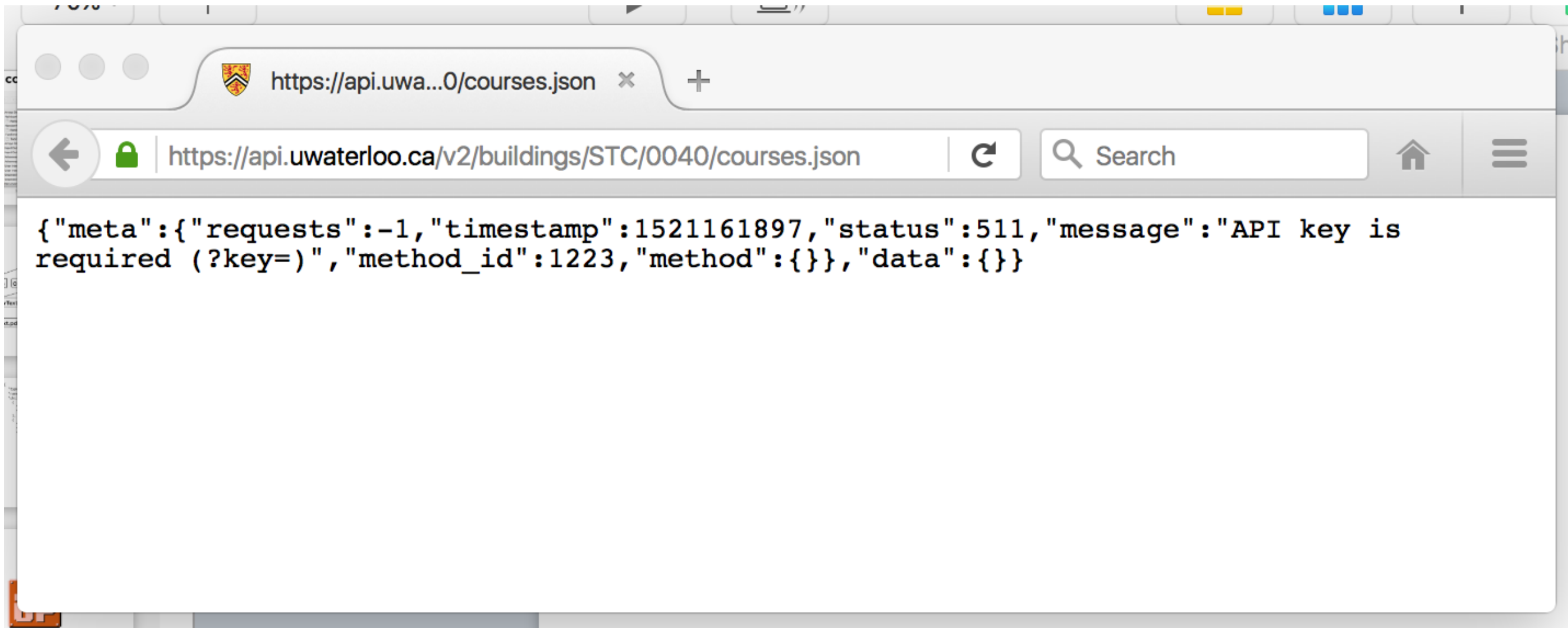
Example: classrooms

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.

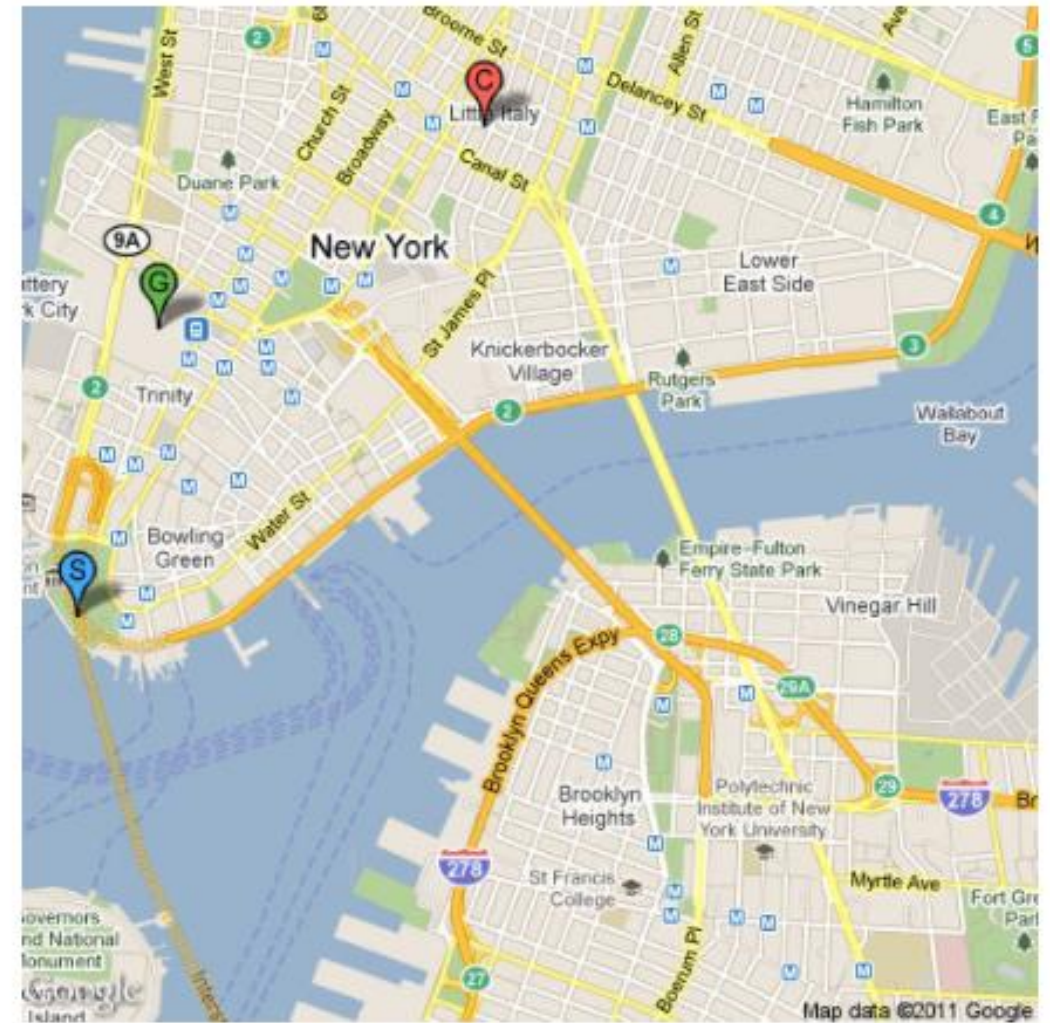


The Google APIs

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

www.webdesignerdepot.com/2011/07/40-useful-apis-for-web-designers-and-developers/

websites, 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.