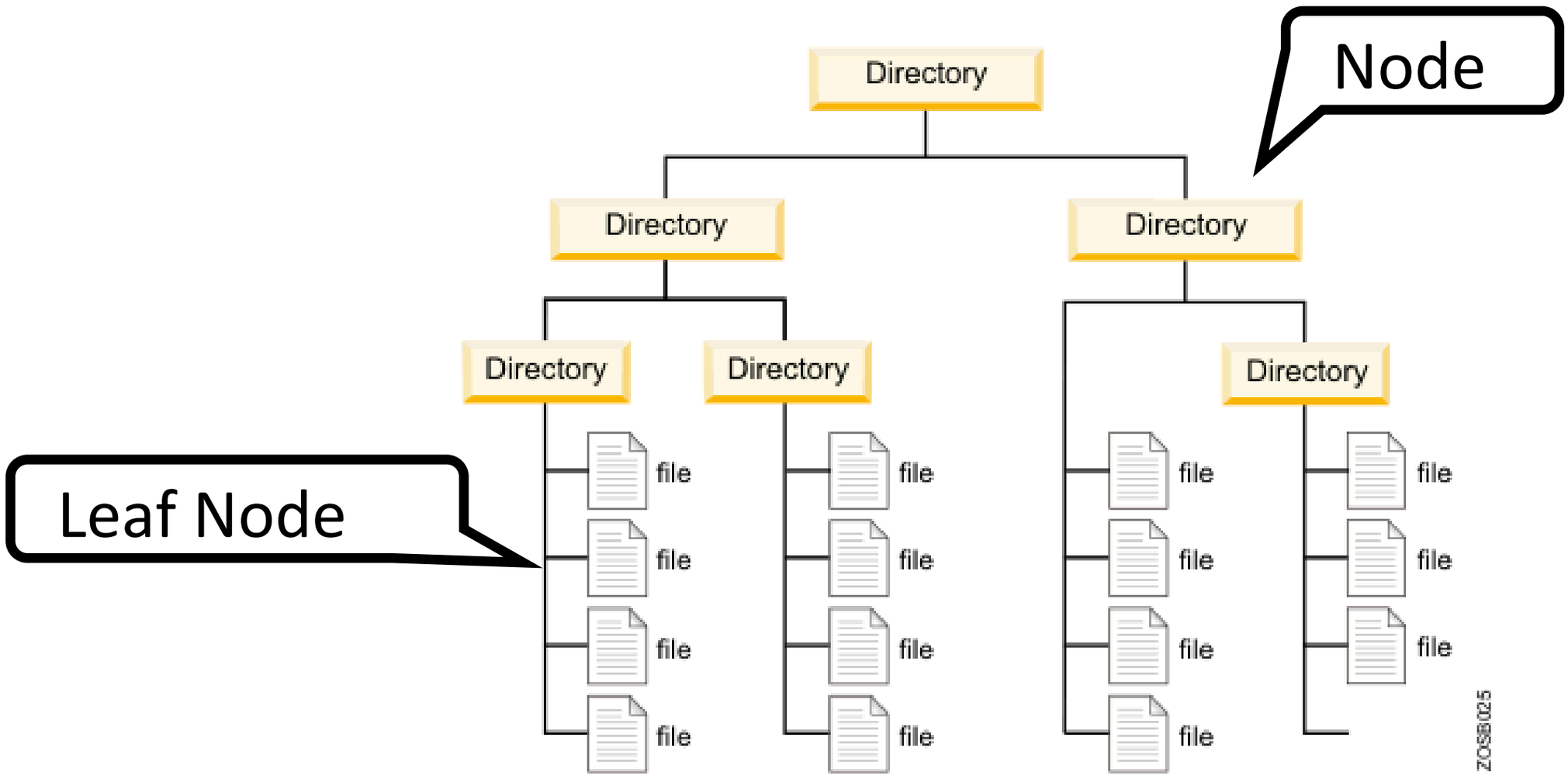
[Rosa Say on flickr]

Tree-Structured data (JSON)

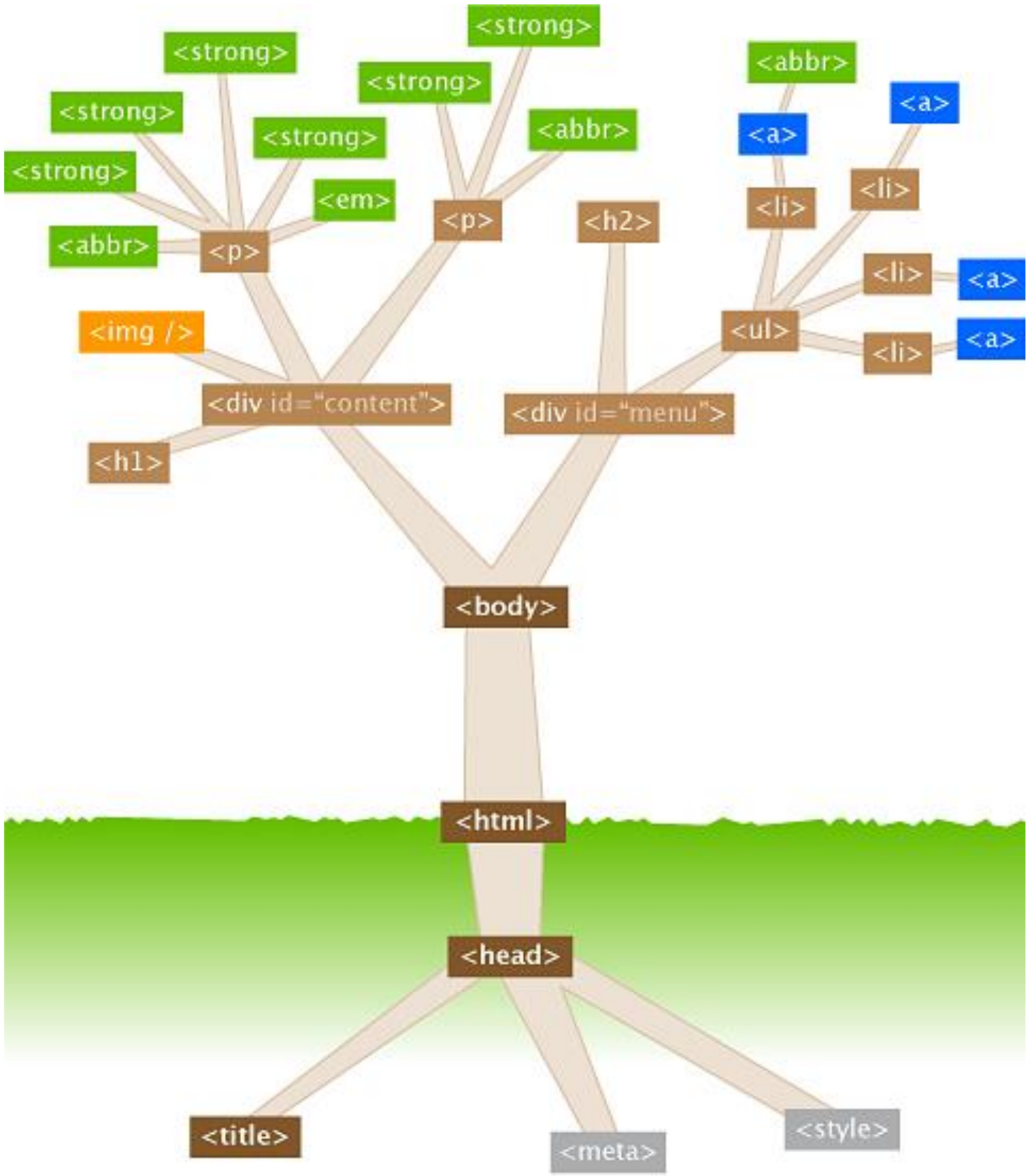
CS 106 Winter 2021

Trees

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




Home
Shop
Socks
Sports Socks
Business Socks
Shoes
Women's Shoes
Men's Shoes
About Us
Our History
Our People
Contact Us



watershedcreative.com/naked/html-tree.html

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

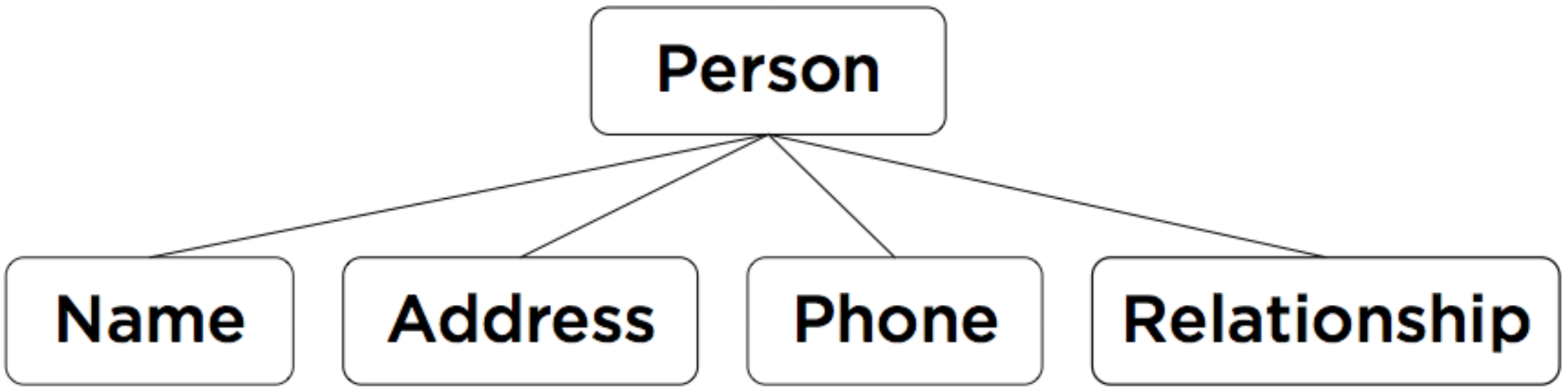
P LANGUAGE AND LITERATURE

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

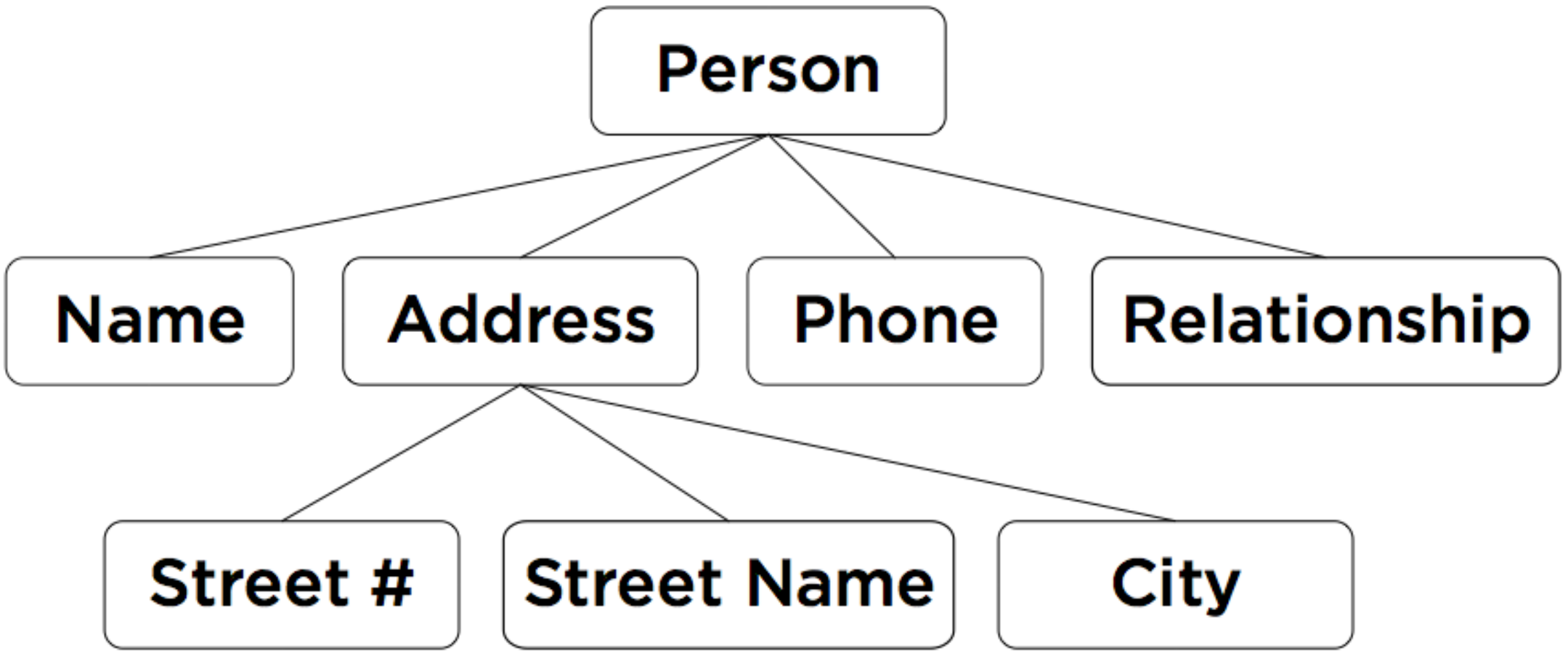
Q SCIENCE

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

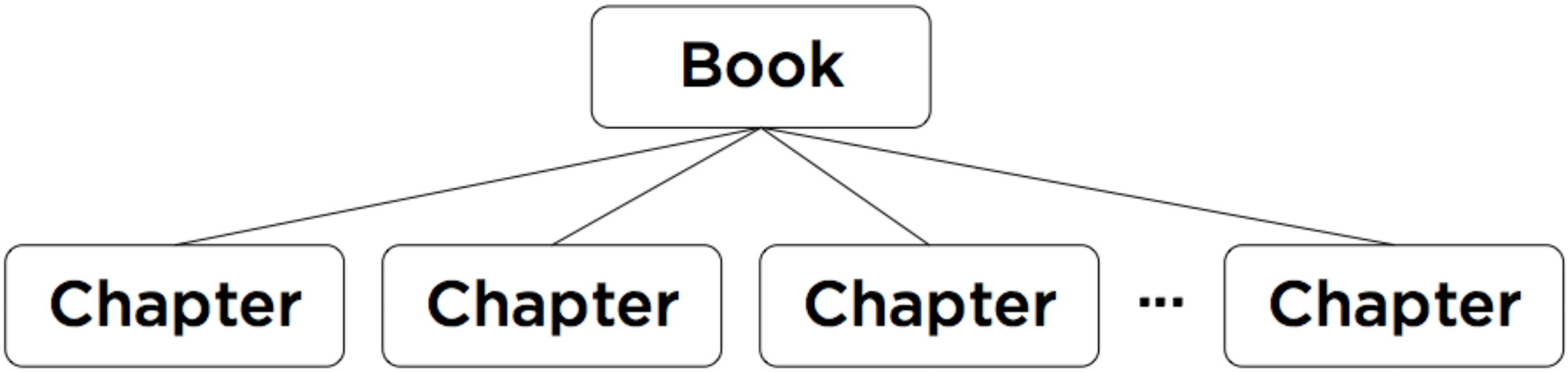
Sometimes, a node behaves like a **set of attributes**: it has a specific slot set aside for each kind of attribute.



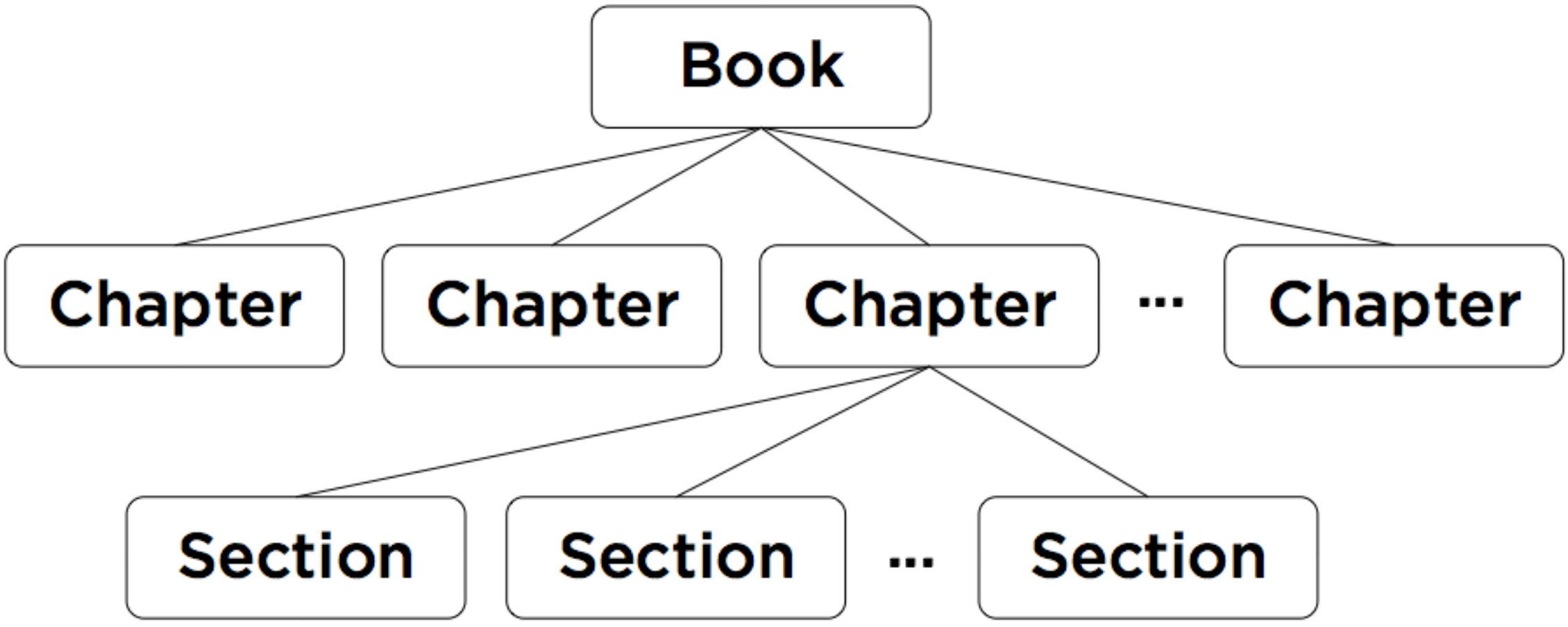
Attributes can have sub-attributes and so on.



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.

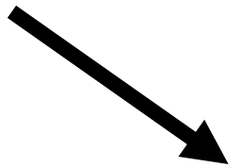
```
{
    “Student ID”: 123,
    “Clicker”: "78%”,
    “Assignments”: “90%”,
    “Midterm”: “91%”,
    “Final”: “93%”
 }
```

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



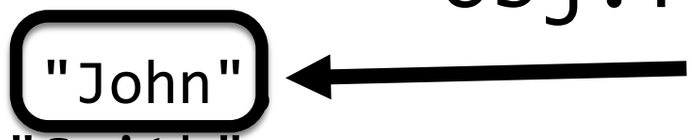
```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type":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

Getting JSON Objects

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

Read the contents of the file into a JSONObject.

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;

```
{
  "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;

I'm an array!

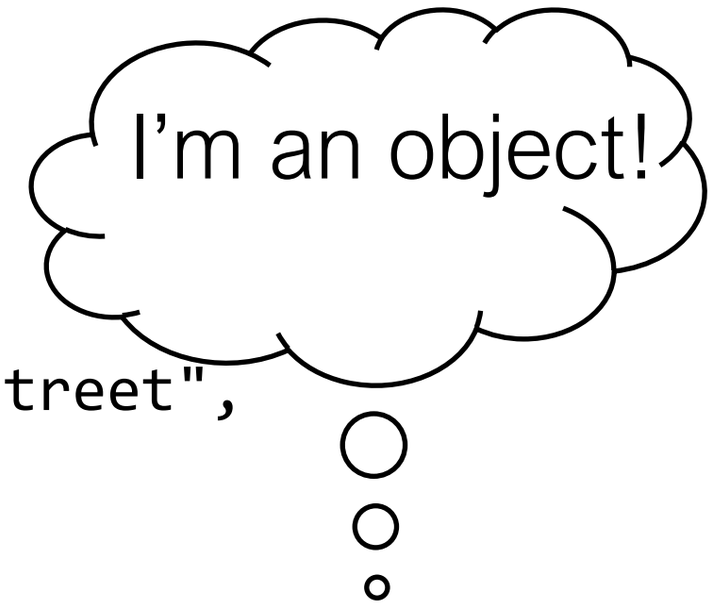
obj.phoneNumbers;

```
{
"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
}
```

```
{
  "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];

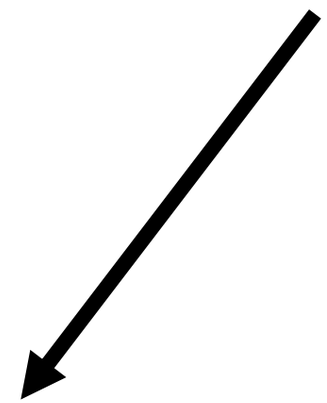
```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type":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



JohnSmith.json


```
let obj = {};

function preload() {
   obj = loadJSON("JohnSmith.json");
}
```

https://openprocessing.org/sketch/1130707

JohnSmith.json


```
function setup() {

   noCanvas();

   createElement("H1", "Practice with a JSON Object");
   createP();

   let fName = obj.firstName;
   createP(fName);

let lName = obj.lastName;
createP(lName);

let fullName = fName + " " + lName;
createP(fullName);
```

JohnSmith.json


```
let age = obj.age;
createP(age);
```

JohnSmith.json


```
// "address" is an object within the object "obj".
// An object within an object.

let addr = obj.address;
createP(addr);
print(addr);

let addrStreet = obj.address.streetAddress;
createP(addrStreet);

let addrCity = obj.address.city;
createP(addrCity);

let addrProvince = obj.address.province;
createP(addrProvince);

let addrPostal = obj.address.postalCode;
createP(addrPostal);
```

```
// "phoneNumbers" is an array of objects within the object "obj".
// Each of the "PhoneNumbers" has a "type" and a "number".

let pNumbers = obj.phoneNumbers;
createP(pNumbers);

let phone1Type = obj.phoneNumbers[0].type;
createP(phone1Type);
let phone1Number = obj.phoneNumbers[0].number;
createP(phone1Number);

let phone2Type = obj.phoneNumbers[1].type;
createP(phone2Type);
let phone2Number = obj.phoneNumbers[1].number;
createP(phone2Number);


for (i = 0; i < pNumbers.length; i++) {
   createP(pNumbers[i].type);
   createP(pNumbers[i].number);
}
```

```
// "kids" is an array of strings within the object "obj".
let kids = obj.children;
for (i = 0; i < kids.length; i++) {
   createP(kids[i]);
}

let partner = obj.spouse;
createP(partner);

createP("The End")
}
```

References

- Daniel Shiffman videos:
  - 10.2: What is JSON? Part I - p5.js Tutorial
    - https://www.youtube.com/watch?v=_NFkzw6oFtQ
  - 10.3: What is JSON? Part II - p5.js Tutorial
    - https://www.youtube.com/watch?v=118sDpLOClw
- Excellent introduction to JSON objects
- Remember: He uses “var” rather than “let”.

API

Application Program Interface

- Used to get data from a server
- For example:
  - openweathermap.org provides weather in a JSON file
  - We write a sketch to request the weather data:
    - Using loadJSON
    - Similar to loading a JSON file from within Open Processing such as “openJSON(‘dogs.json’);”
- We then display something based on the data in the JSON file

Get yourself a free API

from openweathermap.org

- Go to: https://openweathermap.org/
- Select “API” at the top
- Click on “Please sign up and use our fast and easy-to-work weather APIs for free.”
- Follow the sign up steps and you’ll get a key which allows you to use the openweathermap.org API.
  - Your key will look something like this:
    - `01b0f58045147663b1ae918d34d88b4`

OpenWeatherMap1

```
let weather = {};

function preload() {
    weather = loadJSON(
      'https://api.openweathermap.org/data/2.5/
      weather?q=Toronto&APPID=011b0f58045147663b1ea518d34d88b4');
}

function setup() {
    createCanvas(400, 200);
    print(weather);
    print("Wind speed in Toronto is: " + weather.wind.speed);
}
```

https://openprocessing.org/sketch/1137956#

OpenWeatherMap1

```
let weather = {};
```

A variable to hold the JSON object

```
function preload() {
    weather = loadJSON(
      'https://api.openweathermap.org/data/2.5/
      weather?q=Toronto&APPID=011b0f58045147663b1ea518d34d88b4');
}

function setup() {
    createCanvas(400, 200);
    print(weather);
    print("Wind speed in Toronto is: " + weather.wind.speed);
}
```

OpenWeatherMap1

API to get the JSON object

```
let weather = {};

function preload() {
    weather = loadJSON(
      'https://api.openweathermap.org/data/2.5/
      weather?q=Toronto&APPID=011b0f58045147663b1ea518d34d88b4');
}

function setup() {
    createCanvas(400, 200);
    print(weather);
    print("Wind speed in Toronto is: " + weather.wind.speed);
}
```

Note: This key is not valid. You need to get your own free key.

OpenWeatherMap1

```
let weather = {};

function preload() {
    weather = loadJSON(
      'https://api.openweathermap.org/data/2.5/
      weather?q=Toronto&APPID=011b0f58045147663b1ea518d34d88b4');
}
```

```
function setup() {
    createCanvas(400, 200);
    print(weather);
    print("Wind speed in Toronto is: " + weather.wind.speed);
}
```

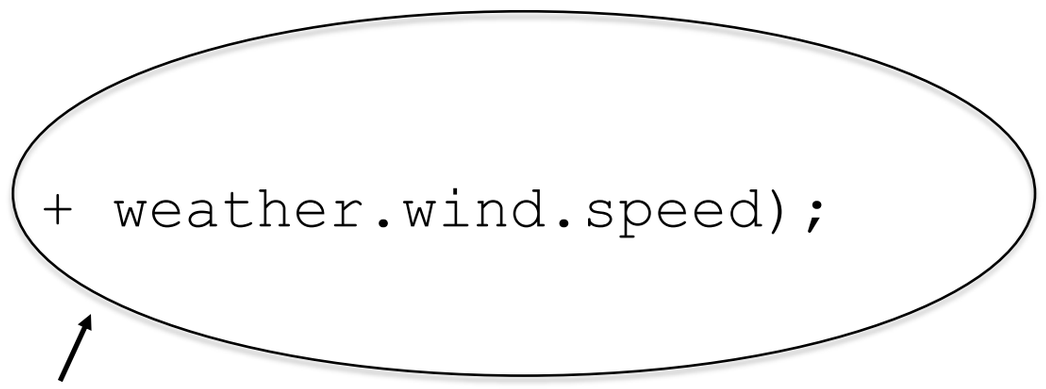
Print out the JSON just to show that a JSON object was successfully returned by loadJSON

OpenWeatherMap1

```
let weather = {};

function preload() {
    weather = loadJSON(
      'https://api.openweathermap.org/data/2.5/
      weather?q=Toronto&APPID=011b0f58045147663b1ea518d34d88b4');
}
```

```
function setup() {
    createCanvas(400, 200);
    print(weather);
    print("Wind speed in Toronto is: " + weather.wind.speed);
}
```



Print out one value just to show something being used from the JSON object

Wind speed is here in the JSON file


```
Object
  coord: Object {"lon":-79.4163,"lat":43.7001}
  weather: Array [{"id":801,"main":"Clouds","description":"few clouds","…
  base: stations
  main: Object {"temp":274.6,"feels_like":271.04,"temp_min":273.71,"te…
  visibility: 10000
  wind: Object {"speed":1.57,"deg":40,"gust":2.06}
  clouds: Object {"all":14}
  dt: 1615944959
  sys: Object {"type":1,"id":718,"country":"CA","sunrise":1615894060,…
  timezone: -14400
  id: 6167865
  name: Toronto
  cod: 200
Wind speed in Toronto is: 1.57
```


Wind speed is printed here

Let’s look at the API request in more detail

```
LoadJSON(
      'https://api.openweathermap.org/data/2.5/
      weather?q=Toronto&APPID=011b0f58045147663b1ea518d34d88b4');
}
```

- openweathermap.org must tell us what to specify in our API request. They specify: https://api.openweathermap.org/data/2.5/weather
- The question mark means “here comes a query”
- openweathermap.org tells us that we can use “q=“ to search for a certain city.
- openweathermap.org says that we must sign up for a free APPID and include that in our query.
- The ampersand is an “and” to string two queries together.

Sample API documentation from openweathermap.org

API call

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid=
{API key}
```

```
api.openweathermap.org/data/2.5/weather?q={city name},{state
code}&appid={API key}
```

```
api.openweathermap.org/data/2.5/weather?q={city name},{state
code},{country code}&appid={API key}
```

Parameters

q	required	City name, state code and country code divided by comma, use ISO 3166 country codes. You can specify the parameter not only in English. In this case, the API response should be returned in the same language as the language of requested location name if the location is in our predefined list of more than 200,000 locations.

Sample API documentation from openweathermap.org

`appid`	required	Your unique API key (you can always find it on your account page under the "API key" tab)
`mode`	optional	Response format. Possible values are `xml` and `html`. If you don't use the `mode` parameter format is JSON by default. Learn more
`units`	optional	Units of measurement. `standard`, `metric` and `imperial` units are available. If you do not use the `units` parameter, `standard` units will be applied by default. Learn more
`lang`	optional	You can use this parameter to get the output in your language. Learn more

Examples of API calls:

```
api.openweathermap.org/data/2.5/weather?q=London&appid={API key}
```

OpenWeatherMap2

- London,ON,CA
- units=metric

```
let weather = {};

function preload() {
    weather = loadJSON(
        'https://api.openweathermap.org/data/2.5
         /weather?q=London,ON,CA&
         APPID=001b0f58045147663b1ea518d34d88b4&units=metric');
}

function setup() {
    createCanvas(400, 200);
    print(weather);
    text("Celcius temperature in London Ontario is: " +
    weather.main.temp, 20, 20);
}
```

https://openprocessing.org/sketch/1137277

Sample API documentation from openweathermap.org

Parameters

`lang`	optional	Language code

Examples of API calls

```
http://api.openweathermap.org/data/2.5/weather?
id=524901&lang=fr&appid={API key}
```

We support the following languages that you can use with the corresponded lang v

- `af` Afrikaans
- `al` Albanian
- `ar` Arabic
- `az` Azerbaijani
- `bg` Bulgarian

OpenWeatherMap3
Try different languages

- lang=ar
- lang=fr
- lang=zh_cn

```
let weather = {};

function preload() {
   weather = loadJSON(

   'https://api.openweathermap.org/data/2.5/weather?lang=ar&
q=London,ON,CA&APPID=001b0f58045147663b1ea518d34d88b4&units=me
tric');
}

function setup() {
   createCanvas(400, 200);
   print(weather);
   text("Celcius temperature in London Ontario is: " +
   weather.main.temp, 20, 20);
}
```

https://openprocessing.org/sketch/1137983#

Let’s look at the API request again

```
weather = loadJSON(

'https://api.openweathermap.org

/data/2.5/Weather

?lang=ar&

q=London,ON,CA&

APPID=001b0f58045147663b1ea518d34d88b4&

units=metric');
```

Let’s Practice

- Display the following:
  - Waterloo, Ontario
  - “feels_like” temperature (in Celsius)
- noCanvas()
  - Create an “H1” header
  - Create a “createP” to hold the temperature
    - Set the paragraph font-size to 24px

OpenWeatherMap4

Feels Like in Metric

-1.51

```
let weather = {};

function preload() {
    weather = loadJSON(

    'https://api.openweathermap.org/data/2.5/weather?&q=Waterloo,ON,CA&APPID=001b0f58045147663b1ea518d34d88b4&units=metric');
}

function setup() {
    noCanvas();
    print(weather);
    createElement("H1", "Feels Like in Metric")
    let p = createP(weather.main.feels_like);
    p.style("font-size", "24px");
}
```

https://openprocessing.org/sketch/1138039

Let’s Practice a Bit More

- Visibility

```
print(weather.visibility);
```

- Longitude

```
print(weather.coord.lon);
```

- The weather description (this one is confusing)

```
print(weather.weather[0].description);
```

- Let’s deal with the dates. JavaScript stores dates as number of milliseconds since January 01, 1970, 00:00:00 UTC (Universal Time Coordinated).

```
let date = new Date(obj.dt * 1000);
let month = date.toLocaleDateString("en-US", {month: "long"});
let day = date.toLocaleDateString("en-US", {day: "numeric"});
print(" " + month + "           " + day);
```

Date example: https://openprocessing.org/sketch/1140627#

More Time Examples
(Credit: Scott Larter)

```
function setup() {
    let unix_timestamp = 1616092816;

    let date = new Date(unix_timestamp * 1000);

    let human_date_format = date.toLocaleDateString() // 3/18/2021

    // other local date formats
    //let human_date_format = date.toLocaleDateString("en-US") // 3/18/2021
    //let human_date_format = date.toLocaleDateString("en-CA") // 2021-03-18

    let month_str = date.toLocaleDateString("en-US", {month: "long"});
    let month_num = date.toLocaleDateString("en-US", {month: "numeric"});
    let day = date.toLocaleDateString("en-US", {day: "numeric"});

    print("Full date: " + human_date_format);
    print("Month string: " + month_str);
    print("Month num: " + month_num);
    print("Day: " + day);
}
```

https://openprocessing.org/sketch/1140641

References

- Daniel Shiffman video:
  - 10.5: Working with APIs in Javascript p5 https://www.youtube.com/watch?v=ecT4206I_WI