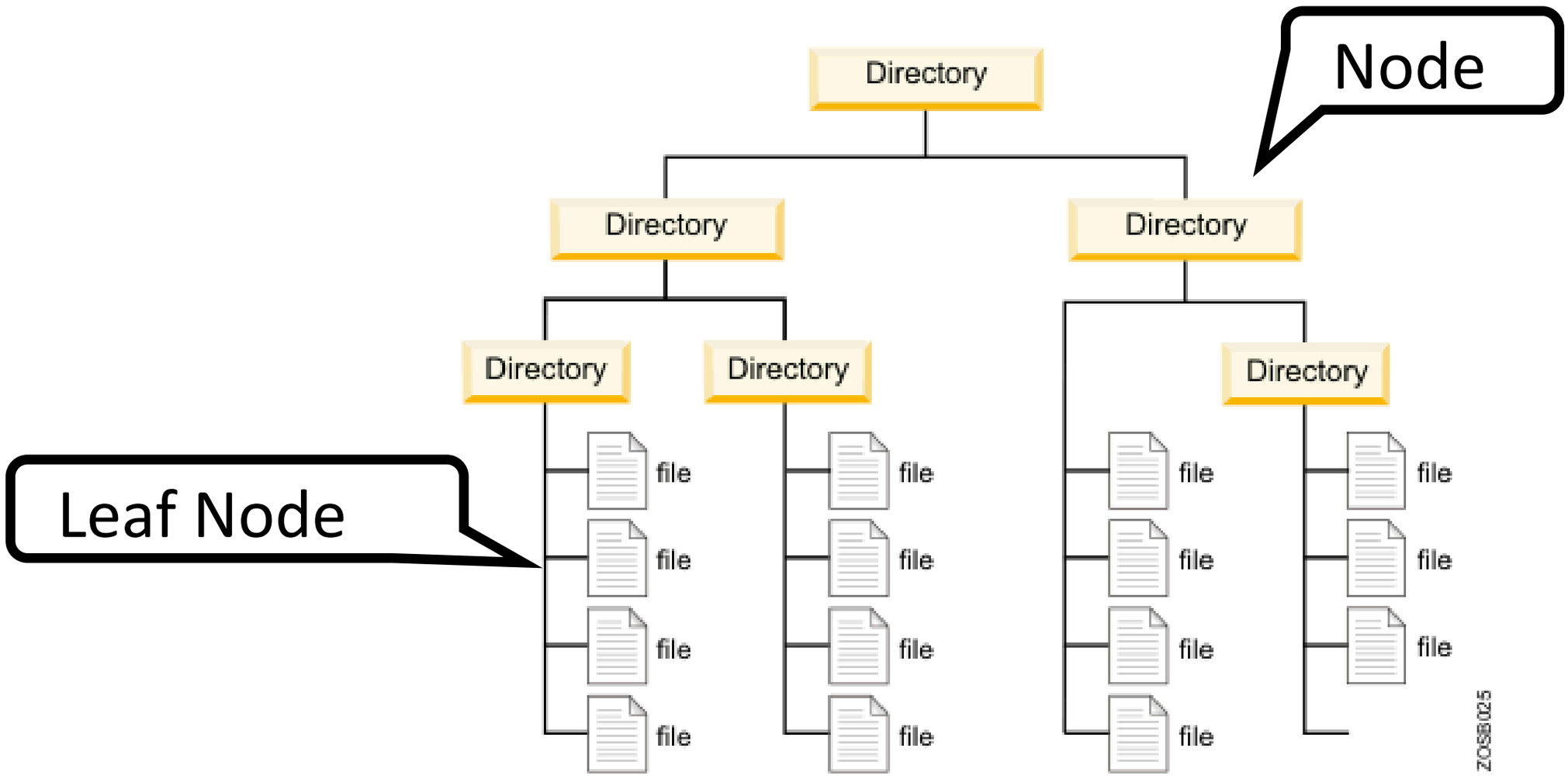
[Rosa Say on flickr]

Tree-Structured data (JSON)

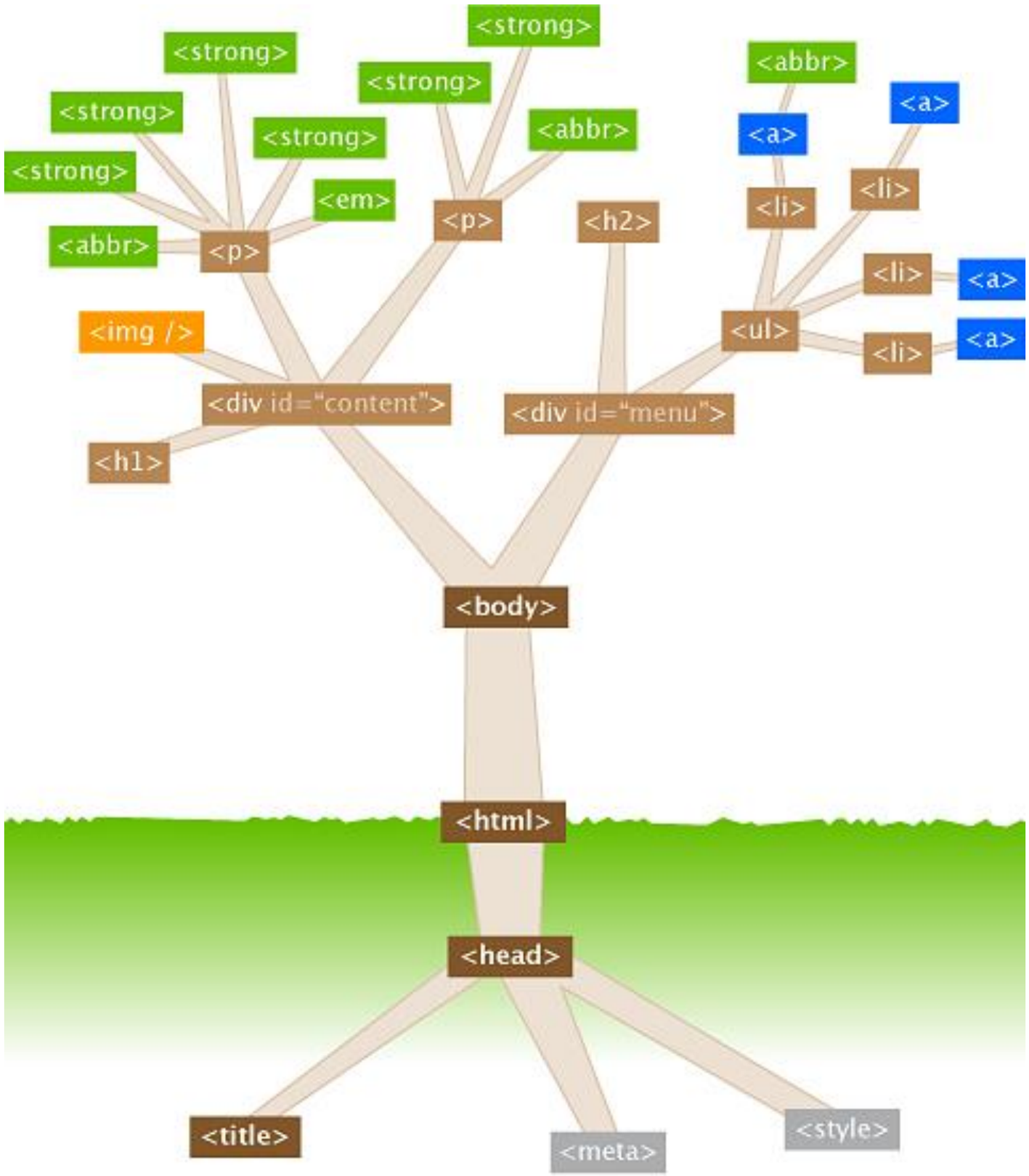
CS 106 Winter 2021

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

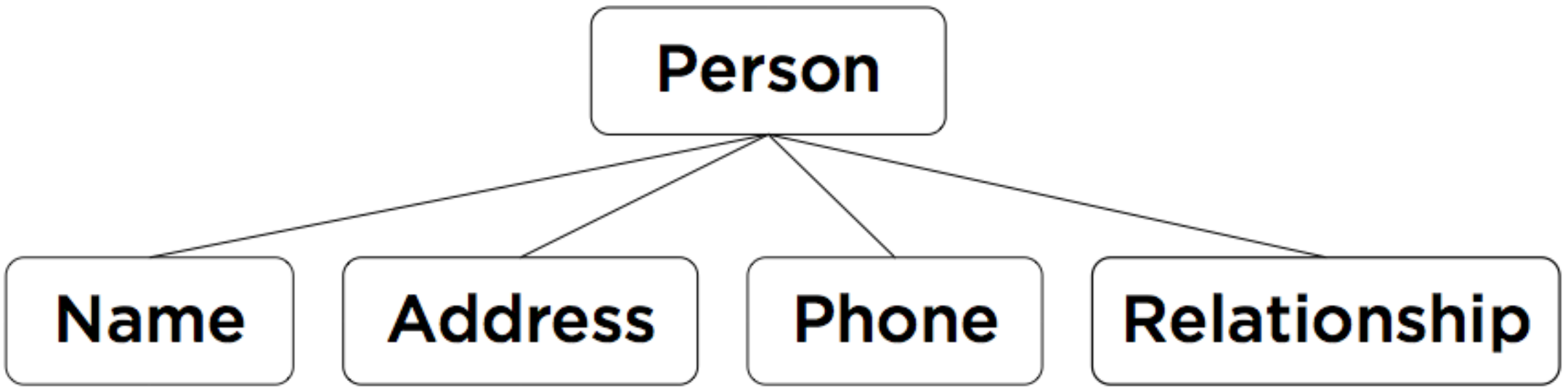
P LANGUAGE AND LITERATURE

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

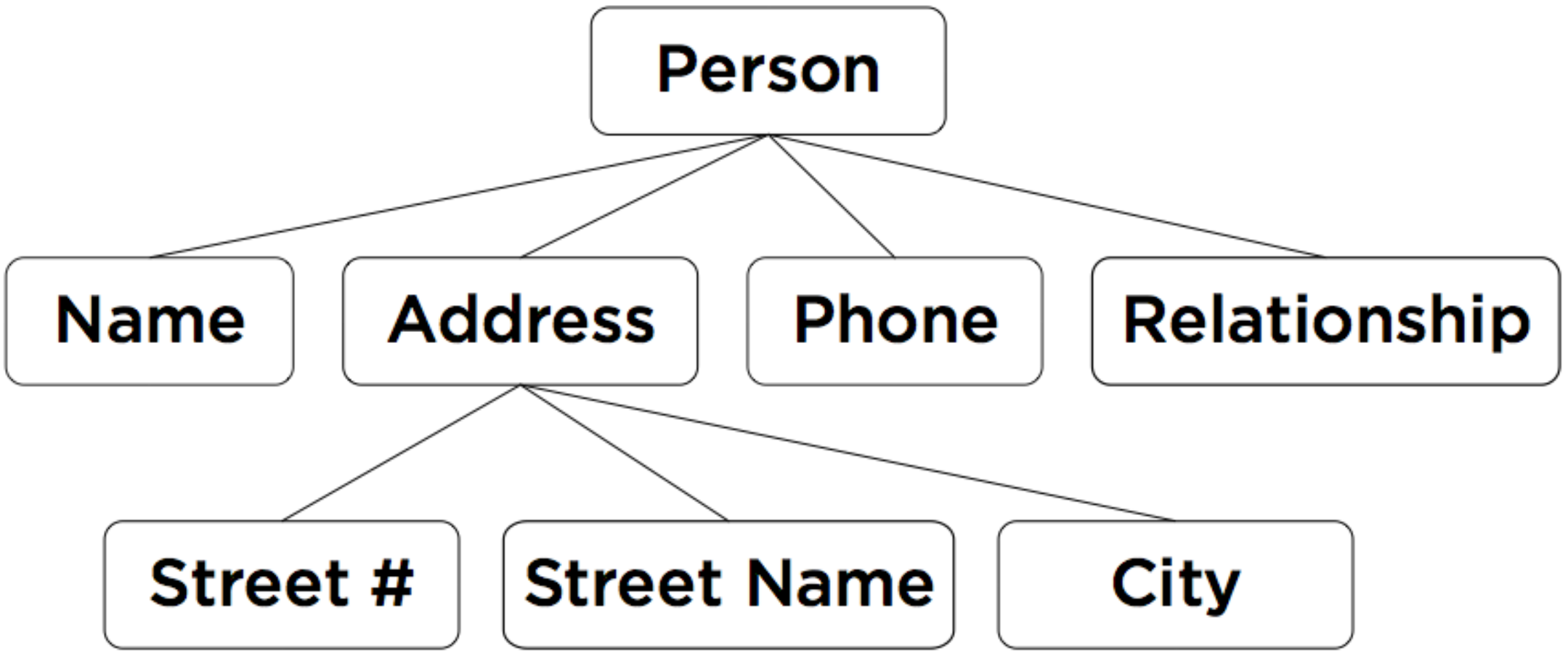
Q SCIENCE

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

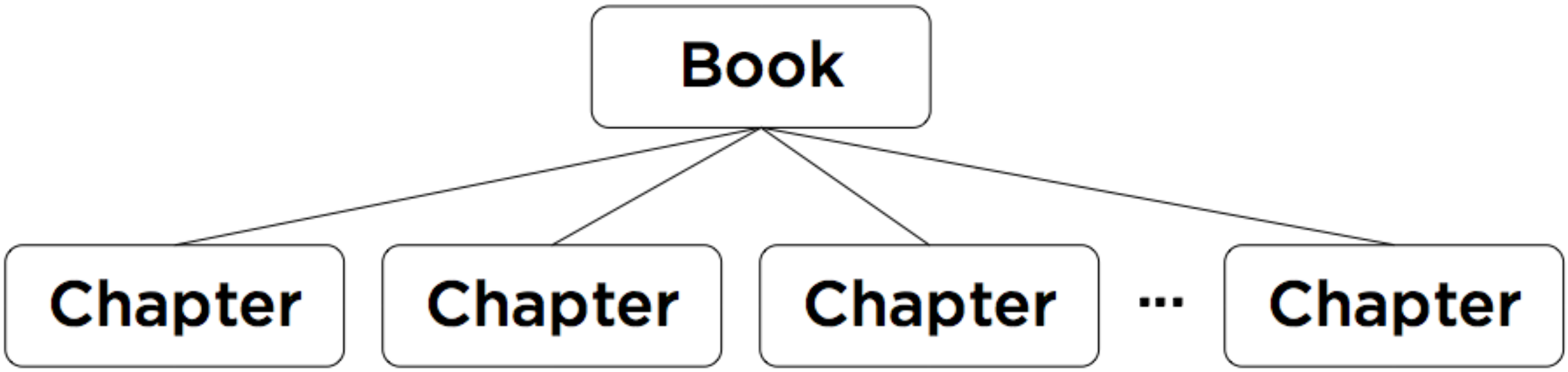
Sometimes, a node behaves like a **set of attributes**: it has a specific slot set aside for each kind of attribute.



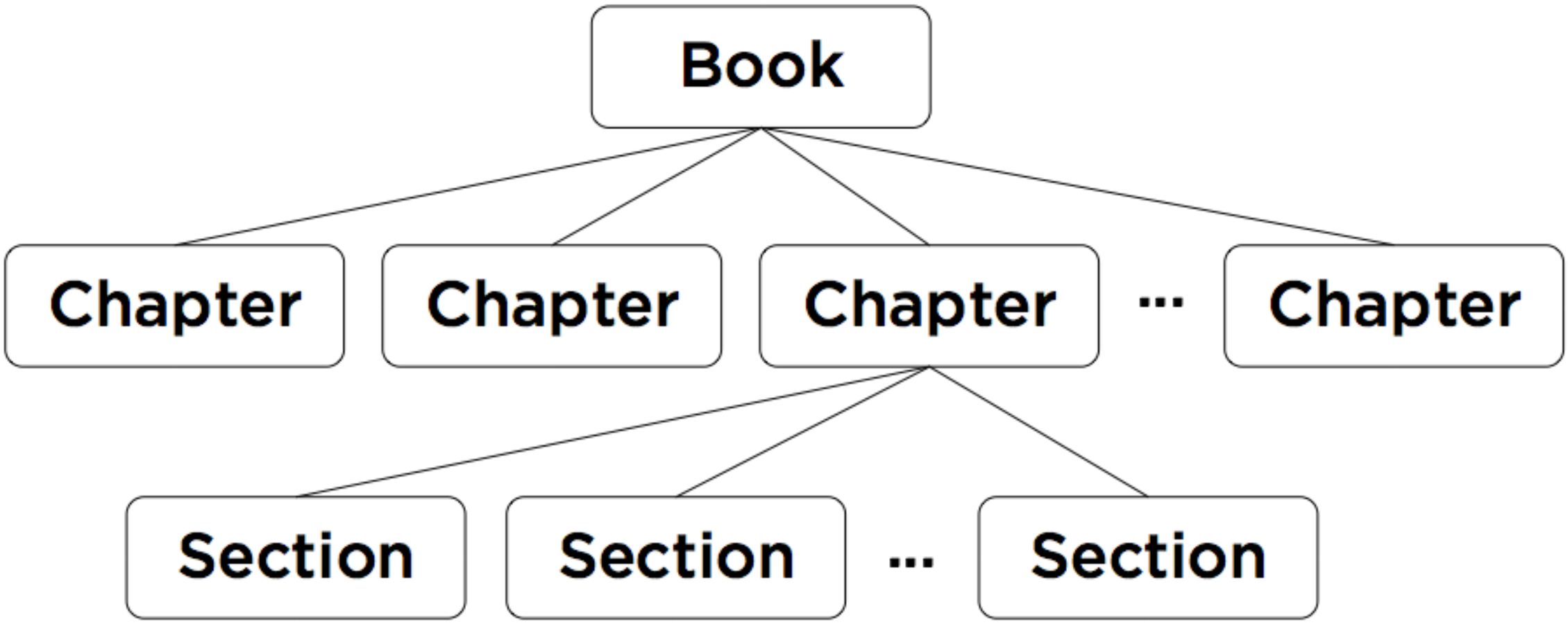
Attributes can have sub-attributes and so on.



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.

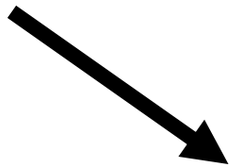
```
{
    “Student ID”: 123,
    “Clicker”: "78%”,
    “Assignments”: “90%”,
    “Midterm”: “91%”,
    “Final”: “93%”
 }
```

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



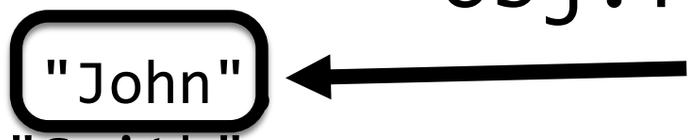
```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type":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

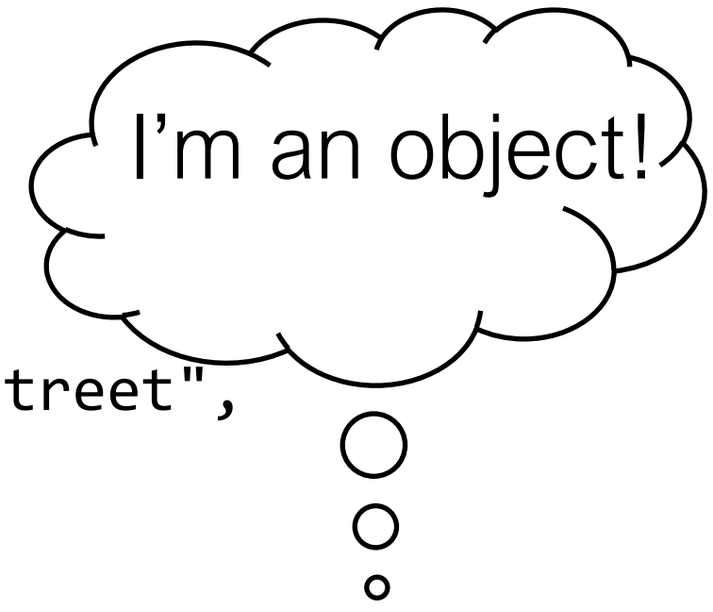
obj.phoneNumbers;

```
{
"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];

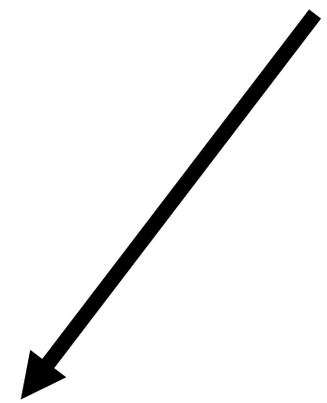
```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 35,
  "address": {
    "streetAddress": "51 Strange Street",
    "city": "Kitchener",
    "province": "ON",
    "postalCode": "N3K 1E7"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "519 555-1234"
    },
    {
      "type":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