CS 217 Data Management and Information Processing

03 - Text Data and Data Organization

Text Representation

Hexadecimal notation

- Computer programmers often use **hex** notation to represent bit sequences.
- Hex takes four bits and represents them as one of sixteen characters:
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
- It’s the most convenient way for people to represent bit sequences (data):
  - binary 0010 1111 0001 0000 = 0x**2F10**
  - “0x” prefix is sometimes added to clarify that what follows is a hexadecimal number.

Decimal	Bits	Hex
0	0000	**0**
1	0001	**1**
2	0010	**2**
3	0011	**3**
4	0100	**4**
5	0101	**5**
6	0110	**6**
7	0111	**7**
8	1000	**8**
9	1001	**9**
10	1010	**A**
11	1011	**B**
12	1100	**C**
13	1101	**D**
14	1110	**E**
15	1111	**F**

Text encodings

- How do computers store text as ones and zeros?
- Early standard is called the American Standard Code for Information Interchange (ASCII)
  - Developed in the 1960s
  - Uses seven bits per character,
    but in practice each character is stored in 8 bits and the top bit is zero.
- ASCII text includes:
  - Lowercase letters, uppercase letters, numbers, punctuation, other symbols
  - Whitespace characters: space, tab, newline, carriage return
  - *Control characters*: null, line feed, vertical tab, bell, escape, delete, backspace, etc.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

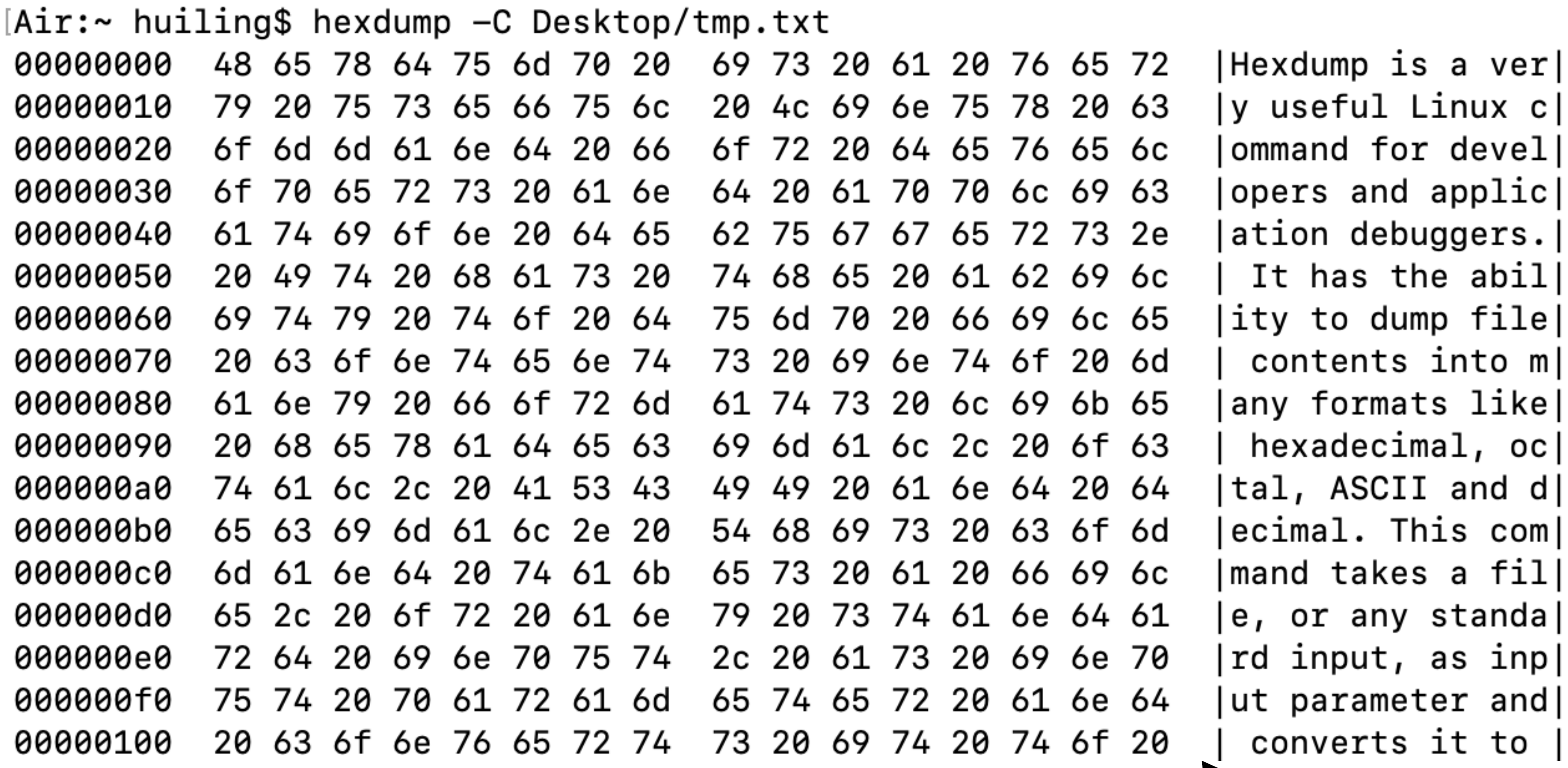
“Hello!” in ASCII

	H	e	l	l	o	!
hex	48	65	6C	6C	6F	21
binary	0100 1000	0110 0101	0110 1100	0110 1100	0110 1111	0010 0001

The ASCII table tells us that the letter “H” is represented by this eight-character bit sequence.
“H” **encodes** to 01001000.
01001000 **decodes** to “H”.

The character “l” has the same **encoding** whenever it appears in ASCII text.

Encoding a text file

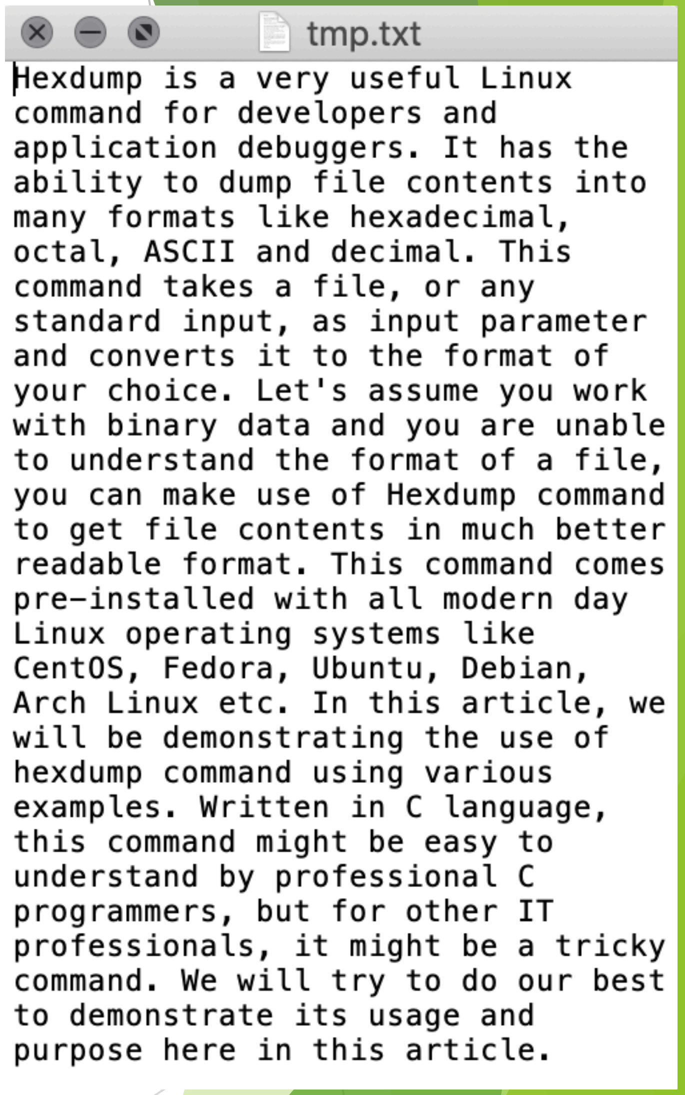


```
[Air:~ huiling$ hexdump -C Desktop/tmp.txt
00000000  48 65 78 64 75 6d 70 20  69 73 20 61 20 76 65 72  |Hexdump is a ver|
00000010  79 20 75 73 65 66 75 6c  20 4c 69 6e 75 78 20 63  |y useful Linux c|
00000020  6f 6d 6d 61 6e 64 20 66  6f 72 20 64 65 76 65 6c  |ommand for devel|
00000030  6f 70 65 72 73 20 61 6e  64 20 61 70 70 6c 69 63  |opers and applic|
00000040  61 74 69 6f 6e 20 64 65  62 75 67 67 65 72 73 2e  |ation debuggers.|
00000050  20 49 74 20 68 61 73 20  74 68 65 20 61 62 69 6c  | It has the abil|
00000060  69 74 79 20 74 6f 20 64  75 6d 70 20 66 69 6c 65  |ity to dump file|
00000070  20 63 6f 6e 74 65 6e 74  73 20 69 6e 74 6f 20 6d  | contents into m|
00000080  61 6e 79 20 66 6f 72 6d  61 74 73 20 6c 69 6b 65  |any formats like|
00000090  20 68 65 78 61 64 65 63  69 6d 61 6c 2c 20 6f 63  | hexadecimal, oc|
000000a0  74 61 6c 2c 20 41 53 43  49 49 20 61 6e 64 20 64  |tal, ASCII and d|
000000b0  65 63 69 6d 61 6c 2e 20  54 68 69 73 20 63 6f 6d  |ecimal. This com|
000000c0  6d 61 6e 64 20 74 61 6b  65 73 20 61 20 66 69 6c  |mand takes a fil|
000000d0  65 2c 20 6f 72 20 61 6e  79 20 73 74 61 6e 64 61  |e, or any standa|
000000e0  72 64 20 69 6e 70 75 74  2c 20 61 73 20 69 6e 70  |rd input, as inp|
000000f0  75 74 20 70 61 72 61 6d  65 74 65 72 20 61 6e 64  |ut parameter and|
00000100  20 63 6f 6e 76 65 72 74  73 20 69 74 20 74 6f 20  | converts it to |
```

Data bits in the file, shown in hex notation for brevity.

(from "hexdump -C" command)

ASCII encoding translates each byte to a character



```
Hexdump is a very useful Linux
command for developers and
application debuggers. It has the
ability to dump file contents into
many formats like hexadecimal,
octal, ASCII and decimal. This
command takes a file, or any
standard input, as input parameter
and converts it to the format of
your choice. Let's assume you work
with binary data and you are unable
to understand the format of a file,
you can make use of Hexdump command
to get file contents in much better
readable format. This command comes
pre-installed with all modern day
Linux operating systems like
CentOS, Fedora, Ubuntu, Debian,
Arch Linux etc. In this article, we
will be demonstrating the use of
hexdump command using various
examples. Written in C language,
this command might be easy to
understand by professional C
programmers, but for other IT
professionals, it might be a tricky
command. We will try to do our best
to demonstrate its usage and
purpose here in this article.
```

Appearance in text editor

What about other characters we might need?

- ¿Español?, 中文, Ελληνικά
- 😀📟🍟⚽
- Different currency symbols
- Even American English uses “weird punctuation” sometimes.
- A single 8-bit (1 byte) will not be enough to store all the possible characters.

UTF-8 to the rescue!

- UTF-8 is now the most common text encoding.
- The latest version includes 136,690 symbols, and more can be added.
  - Can eventually be expanded to more than two million characters
- It’s a **variable-length** encoding
  - Characters are represented with one, two, three, or four bytes.
  - ASCII is fixed-length of one byte.
- Backward-compatible with ASCII
  - ASCII text is also valid UTF-8
  - Previous version of Unicode (such as UTF-16) were not widely adopted due to incompatibility with ASCII.

Variable length character encoding with UTF-8

1st byte	2nd byte	3rd byte	4th byte	# of free bits
`0... ....`				7 (ASCII)
`110. ....`	`10.. ....`			11
`1110 ....`	`10.. ....`	`10.. ....`		16
`1111 0...`	`10.. ....`	`10.. ....`	`10.. ....`	21

- Single-byte characters are identical to ASCII
- First byte tells you how many total bytes to expect
- Every “extra” byte starts with “10”
  - If you start reading in the middle of a character you’ll know it.
  - It’s very easy to know where each new character starts.

Decimal numbers in text

- We can store decimal numbers using the chars [0-9.eE\-]
- For example:
  - “12” = “1” + “2” = 0x 31 32 = 0011 0001 0011 0010
  - “12.2e-4” = “1” + “2” + “.” + “2” + “e” + “-” + “4”
    = 0x 31 32 2E 32 65 2D 34
    = 0011 0001 0011 0010 0010 1110 0011 0010 0110 0101 0010 1101 0011 0100
- These text-based encodings are **inefficient** because they only make use of a small subset of the characters.
- However, they are easy to read and machine-independent.
- A general-purpose compressor like “zip” works well on text.
- Other numeric encodings introduced in the last lecture work directly with the bits, not with text.

Data Organization

Data Files

- A computer **file** is a container for data, and files have:
  - A ***path*** (sequence of folders and a filename):

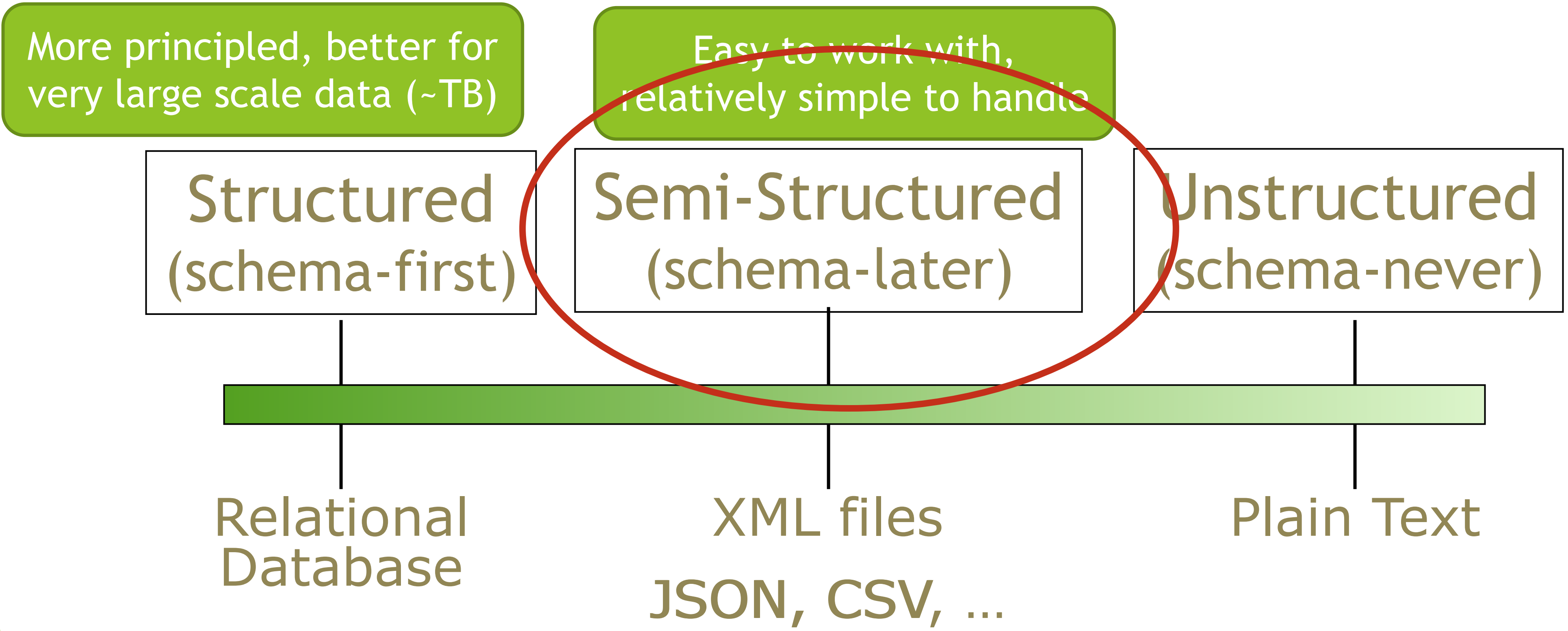
    `/Users/Documents/my_data.csv`
  - A sequence of data bytes “in” the file (8 bits = 1 byte):

    `00010101 10110101 11010101 11010010 10100011 01010101 1111011 …`
  - Other metadata like *permissions*, depending on the filesystem type.
- Files are:
  - **Persistent**, meaning that they remain in the computer after it is rebooted
  - **Sharable** by other programs running on the computer
- Thus, files allow programs to
  - Save their own data
  - Share data with other programs on the same computer
  - Transfer data between computers
- Databases are a more powerful alternative to plain files (“flat files”), but they are not as *portable*.
  - we still use flat files to exchange bulk data.

Standard data file formats

- The filename *extension* conventionally determines the ***file format***.
  - Tells us how to interpret the sequence of bits in the file
- Some file formats use **human-readable** ASCII or UTF-8 **text**.
  - `txt, csv, json, xml`
- More efficient file formats represent data directly in **binary** form.
  - `mat` (matlab), `RData, sqlite, jpg, zip`
- Some files use both formats in two stages:
  - human-readable files that have been compressed to a binary format:
  - `xlsx, docx, csv.gz, txt.gz`

Data File Organization Spectrum



More principled, better for very large scale data (~TB)

Easy to work with, relatively simple to handle

Structured (schema-first)

Semi-Structured (schema-later)

Unstructured (schema-never)

Relational Database

XML files

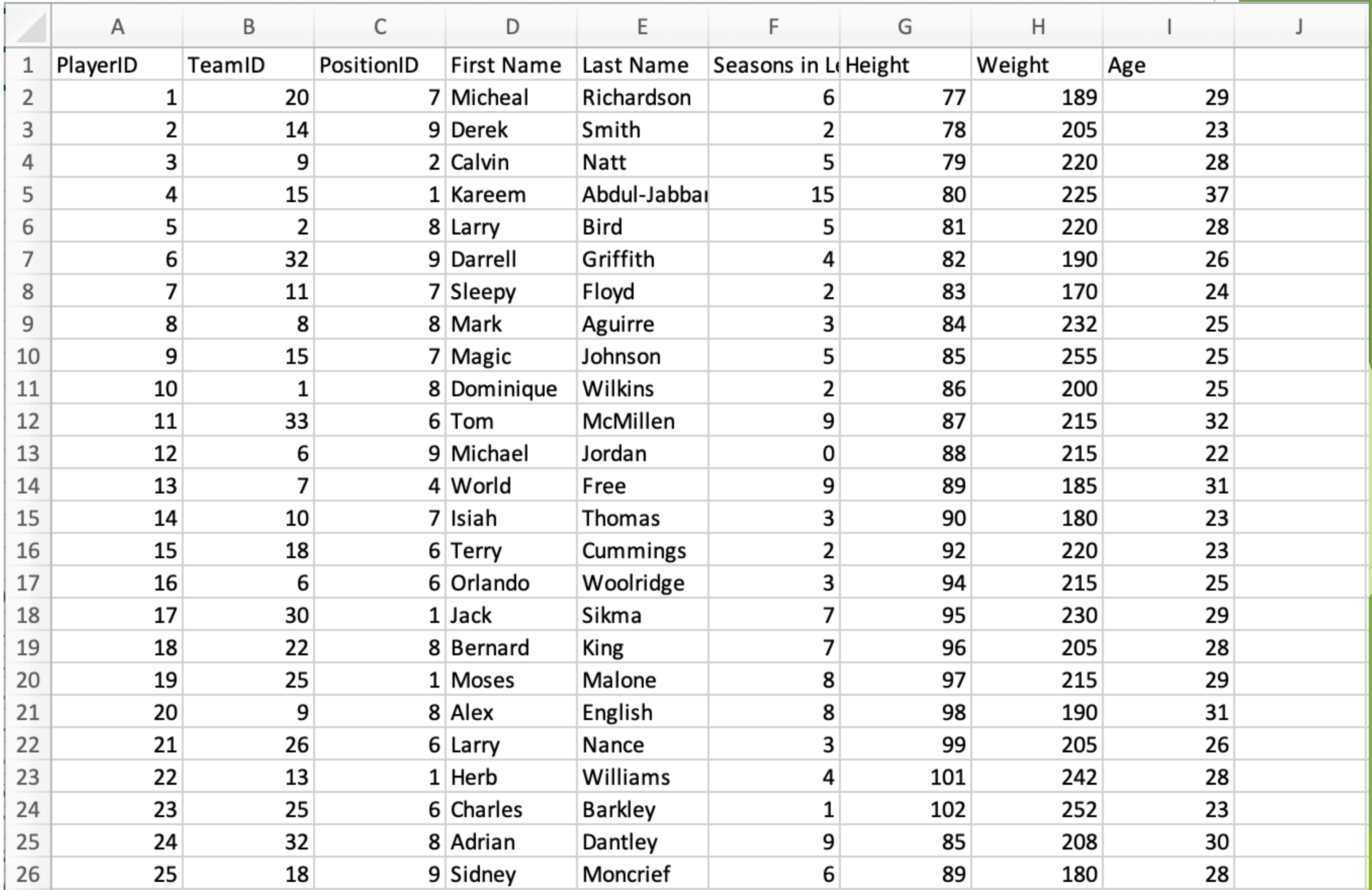
JSON, CSV, …

Plain Text

Comma Separated Values (CSV)

- CSV is a simple text format for storing tabular data (spreadsheets)
- Each row is represented on one line of text
- Columns are separated by commas
- Values can be enclosed in double quotes ("...") if necessary
  - For example, if value includes comma or newline characters
  - Double quotes within a text value must be “escaped” by using two double quotes
- Values can be empty by having nothing between the commas

NBA_player_of_the_week.csv viewed in Excel



	A	B	C	D	E	F	G	H	I	J
1	PlayerID	TeamID	PositionID	First Name	Last Name	Seasons in Le	Height	Weight	Age	
2	1	20	7	Micheal	Richardson	6	77	189	29	
3	2	14	9	Derek	Smith	2	78	205	23	
4	3	9	2	Calvin	Natt	5	79	220	28	
5	4	15	1	Kareem	Abdul-Jabbar	15	80	225	37	
6	5	2	8	Larry	Bird	5	81	220	28	
7	6	32	9	Darrell	Griffith	4	82	190	26	
8	7	11	7	Sleepy	Floyd	2	83	170	24	
9	8	8	8	Mark	Aguirre	3	84	232	25	
10	9	15	7	Magic	Johnson	5	85	255	25	
11	10	1	8	Dominique	Wilkins	2	86	200	25	
12	11	33	6	Tom	McMillen	9	87	215	32	
13	12	6	9	Michael	Jordan	0	88	215	22	
14	13	7	4	World	Free	9	89	185	31	
15	14	10	7	Isiah	Thomas	3	90	180	23	
16	15	18	6	Terry	Cummings	2	92	220	23	
17	16	6	6	Orlando	Woolridge	3	94	215	25	
18	17	30	1	Jack	Sikma	7	95	230	29	
19	18	22	8	Bernard	King	7	96	205	28	
20	19	25	1	Moses	Malone	8	97	215	29	
21	20	9	8	Alex	English	8	98	190	31	
22	21	26	6	Larry	Nance	3	99	205	26	
23	22	13	1	Herb	Williams	4	101	242	28	
24	23	25	6	Charles	Barkley	1	102	252	23	
25	24	32	8	Adrian	Dantley	9	85	208	30	
26	25	18	9	Sidney	Moncrief	6	89	180	28	

NBA_player_of_the_week.csv viewed as text

```
PlayerID,TeamID,PositionID,First Name,Last Name,Seasons in League,Height ,Weight,Age
1,20,7,Micheal,Richardson,6,77,189,29
2,14,9,Derek,Smith,2,78,205,23
3,9,2,Calvin,Natt,5,79,220,28
4,15,1,Kareem,Abdul-Jabbar,15,80,225,37
5,2,8,Larry,Bird,5,81,220,28
6,32,9,Darrell,Griffith,4,82,190,26
7,11,7,Sleepy,Floyd,2,83,170,24
8,8,8,Mark,Aguirre,3,84,232,25
9,15,7,Magic,Johnson,5,85,255,25
10,1,8,Dominique,Wilkins,2,86,200,25
11,33,6,Tom,McMillen,9,87,215,32
12,6,9,Michael,Jordan,0,88,215,22
13,7,4,World,Free,9,89,185,31
14,10,7,Isiah,Thomas,3,90,180,23
15,18,6,Terry,Cummings,2,92,220,23
16,6,6,Orlando,Woolridge,3,94,215,25
17,30,1,Jack,Sikma,7,95,230,29
18,22,8,Bernard,King,7,96,205,28
19,25,1,Moses,Malone,8,97,215,29
20,9,8,Alex,English,8,98,190,31
21,26,6,Larry,Nance,3,99,205,26
```

Tree-like Structures

- Data is organized in a tree-like/hierarchical way, where any item can have more details below it.
  - E.g., JSON and XML
  - Not limited to two dimensions like CSV files
- However, unlike a relational database, there is no clear pre-defined structure or schema for the data.
- **The data defines its own structure.**

- Compared to CSV, it’s more difficult to read and is more prone to errors because data elements can be missing.

JavaScript Object Notation (JSON)

- Is:
  - A lightweight text based data-interchange format
  - Completely language independent
  - Based on a subset of the JavaScript Programming Language
  - Easy to understand, manipulate and generate
- Is Not:
  - Overly Complex
  - A “document” format
  - A markup language
  - A programming language

JSON Syntax

Basic components are:

- [ ] for ordered lists
  - Items are separated by commas
  - Items can be any JSON
- { } for unordered dictionaries/objects
  - Key: value pairs are separated by commas
  - Keys must be strings (text)
  - Values can be any JSON

Data Types:

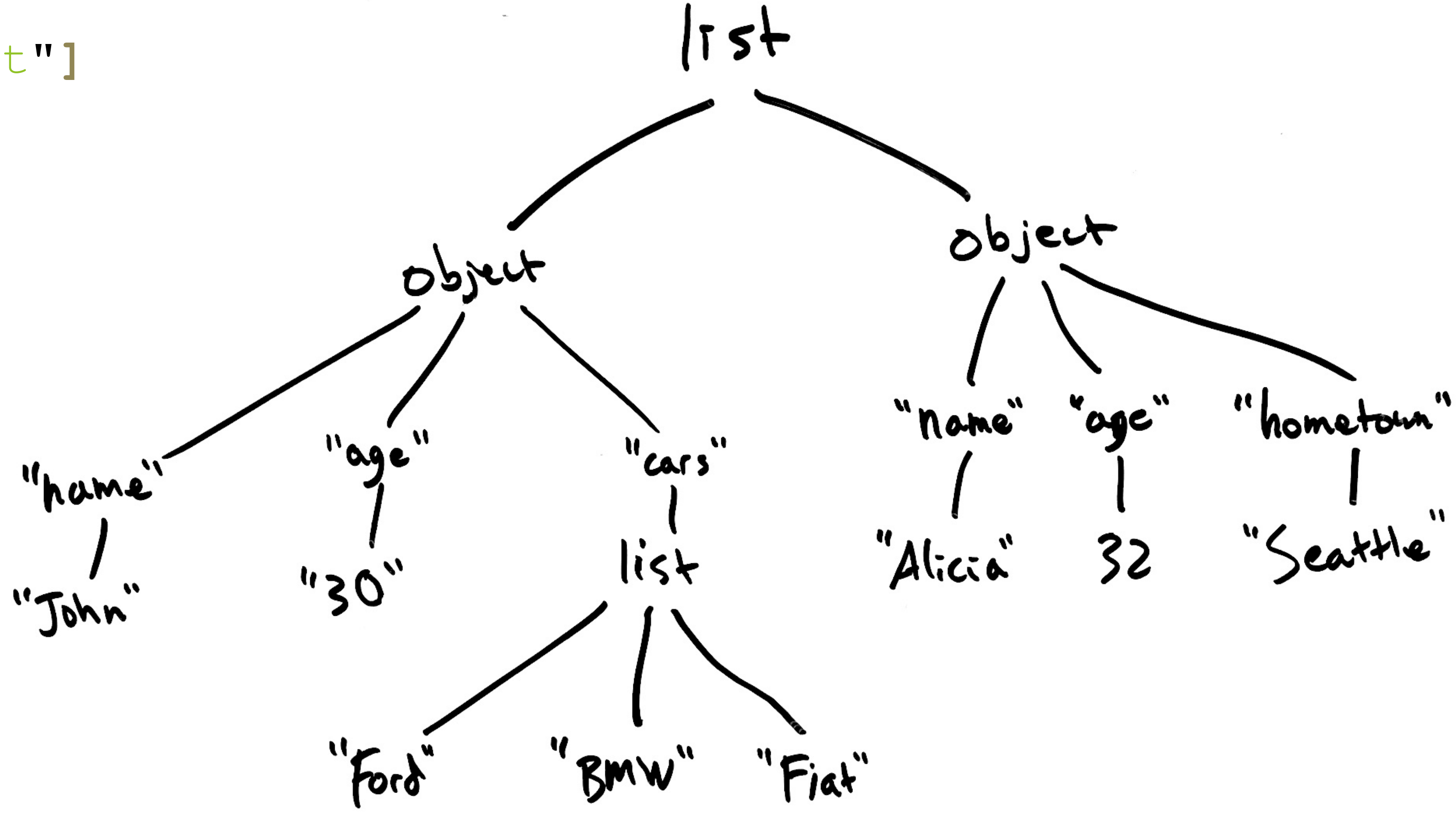
- Strings
  - Sequence of 0 or more Unicode characters
  - Wrapped in “double quotes”
  - Backslash escapement
- Numbers
  - Integer, float
  - No NaN, use null
- Booleans
  - true, false

JSON Example

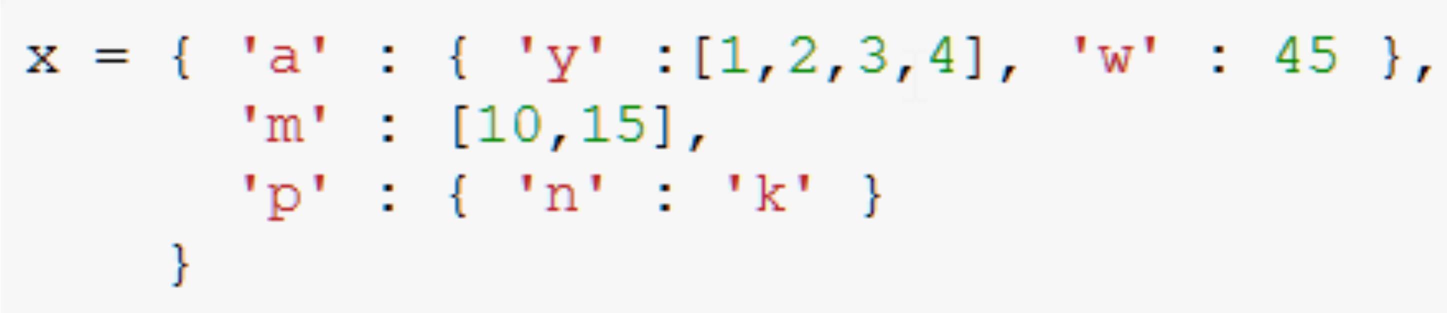
```
var employeeData = {
    "employee_id": 1234567,
    "name": "Jeff Fox",
    "hire_date": "1/1/2013",
    "location": "Norwalk, CT",
    "consultant": false
};
```

JSON data graph example

```
[
  {
    "name": "John",
    "age": 30,
    "cars":
       ["Ford", "BMW", "Fiat"]
  },
  {
    "name": "Alicia",
    "age": 32,
    "hometown": "Seattle"
  }
]
```



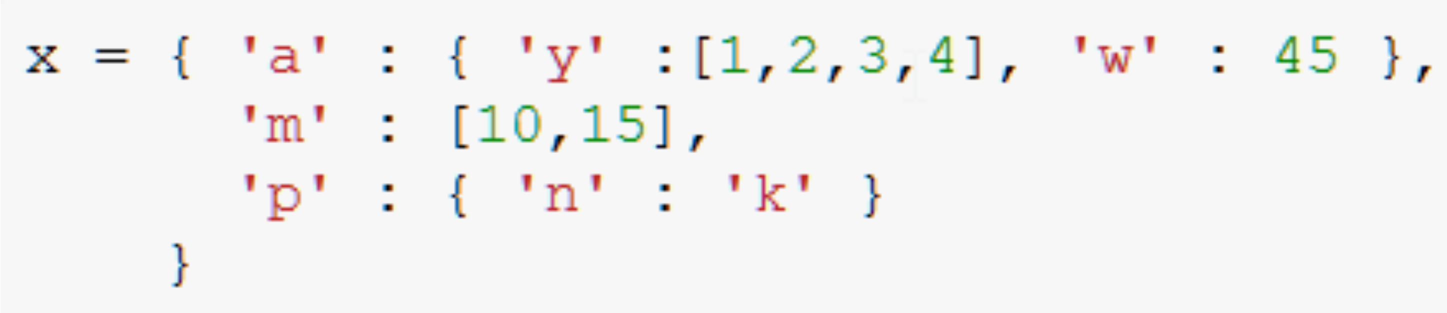
Quiz



```
x = { 'a' : { 'y' :[1,2,3,4], 'w' : 45 },
      'm' : [10,15],
      'p' : { 'n' : 'k' }
    }
```

1. x[‘a’] ?
2. x[‘a’][‘m’] ?
3. x[‘m’][1] ?
4. x[‘p’][‘n’] ?
5. x[‘a’][‘w’] ?
6. x[‘a’][‘y’][-1] ?

Quiz



```
x = { 'a' : { 'y' :[1,2,3,4], 'w' : 45 },
      'm' : [10,15],
      'p' : { 'n' : 'k' }
    }
```

1. x[‘a’] = {‘y’:[1,2,3,4], ‘w’:45}
2. x[‘a’][‘m’] -- error
3. x[‘m’][1] = 15
4. x[‘p’][‘n’] = ‘k’
5. x[‘a’][‘w’] = 45
6. x[‘a’][‘y’][-1] = 4

Where is JSON used today?

- Everywhere!

And many, many more!

XML

- e**X**tensible **M**arkup **L**anguage
- Older than JSON, and now is less common than JSON because many people think XML is unnecessarily complicated.
- HTML is an XML document that defines a web page.

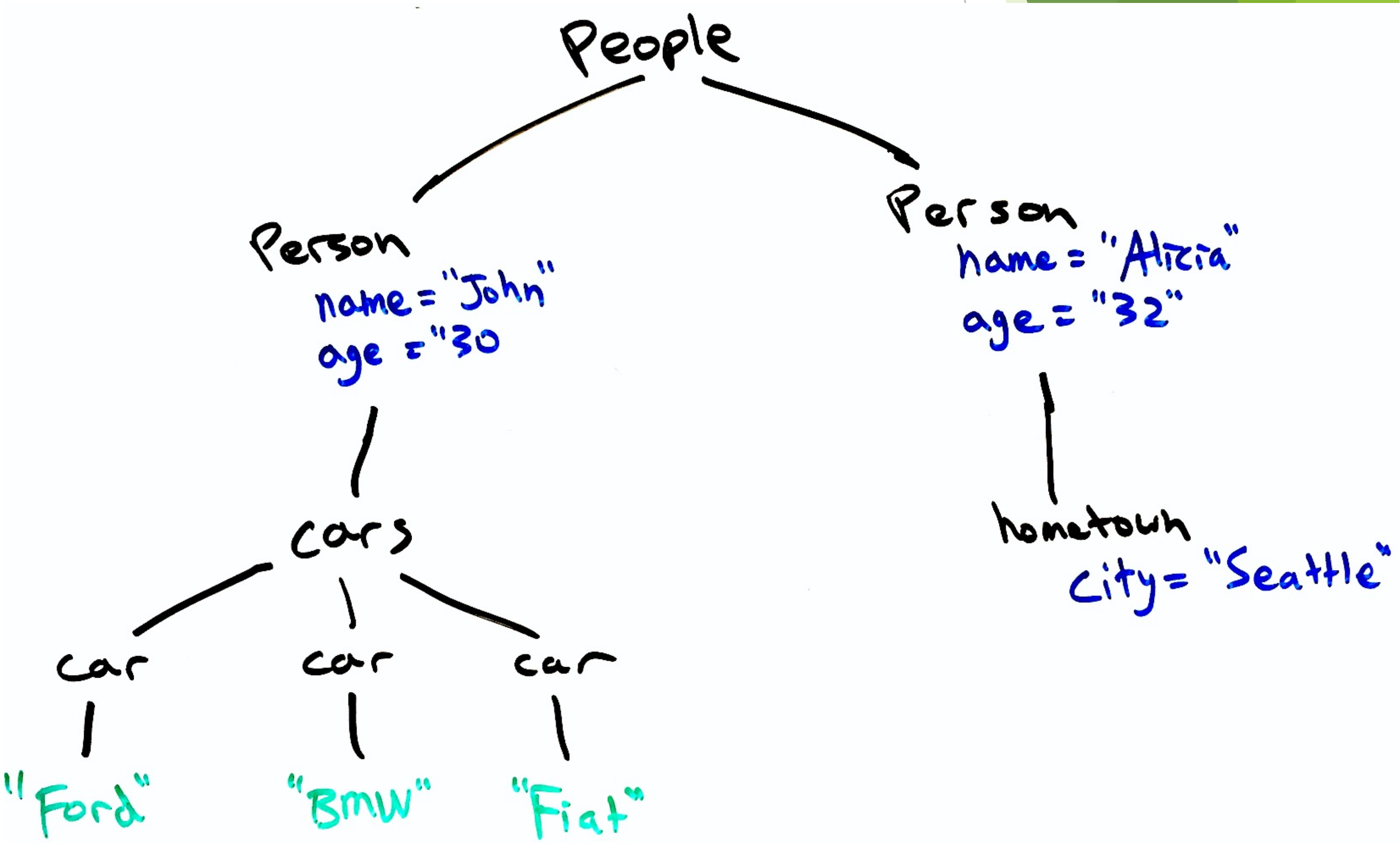
Basic components are:

- Text
- Tags
  - `<tagname>…</tagname>` or just `<tagname>`
  - Have a name, and have XML inside
  - Each start tag has a corresponding end tag, but only if it has data inside.
- Attributes
  - `<tag attr="value" …>`
  - Appear within tags
  - Attribute name and value must be text
  - Tag can have multiple attributes, but each must have a unique name

```
<people>
  <person name="John"
          age="30">
    <cars>
      <car>Ford</car>
      <car>BMW</car>
</cars>
  </person>
  <person name="Alicia"
          age="32">
    <hometown city="Seattle">
  </person>
</people>
```

XML data graph example

```
<people>
  <person name="John"
          age="30">
    <cars>
      <car>Ford</car>
      <car>BMW</car>
      <car>Fiat</car>
    </cars>
  </person>
  <person name="Alicia"
          age="32">
    <hometown
     city="Seattle">
  </person>
</people>
```



Comparison of data formats

	Proprietary	SQL	CSV	JSON	XML
Space efficiency	Compact binary representation	Bloated text with SQL syntax	Text with little extra syntax	Text with little extra syntax	Text with verbose tag names
Compatibility (readable by many)	Must use specific program/DB	Each DBMS has its own SQL dialect	Standardized format	Standardized format	Standardized format
Expressibility (data complexity)	Complex relationships	Complex relationships	Represents a single table	Complex relationships	Complex relationships
Popularity	Rare	Common	Common	Common	Less common
Flexibility/rigidity	SQL DBs are have a clearly defined schema that must be obeyed.		Rows all have same columns.	Data and schema are defined together. Different elements can have different attributes.	

Summary: Data types

- Data is exchanged by data files (arrays of bits, zeros and ones).
- Several file formats are common:
  - CSV, XML, JSON, SQL and proprietary formats.
- Many of these formats are text files with special syntax.
- Text files represent each character with a certain bit sequence.
  - ASCII uses 8 bits (one byte) for each character
  - UTF-8 uses 1-4 bytes for each character, is backward-compatible with ASCII
- CSV files store just one table.
- JSON and XML files represent data with complex, nested relationships
  - However, no schema is defined ahead of time.
  - Data itself gives the structured (hence, we call it **semi-structured** data).
  - Python and R scripts can easily load these files.