# COMP1531

# 📚 Full-Stack - Data Interchange

## Lecture 3.1

Author(s): Hayden Smith

(Download as PDF)

# In This Lecture

- **Why? 🤔**
  - We need a common way of communicating between different operating systems, servers, and programming languages

- **What? 📰**
  - Standard Interfaces
  - JSON
  - YAML
  - XML

# Standard Interfaces

What is a standard interface?

# Standard Interfaces

What is a standard interface?

What do you think of when you think of a standard interface?

# Standard Interfaces

What is a standard interface?

What do you think of when you think of a standard interface?

# Standard Interfaces

What is a standard interface?

What do you think of when you think of a standard interface?

# Standard Interfaces

What is a standard interface?

What do you think of when you think of a standard interface?

# 🔌 Standard Interfaces

What is a standard interface?

What do you think of when you think of a standard interface?

# Standard Interfaces

What do all of these systems have in common?

# 🔌 Standard Interfaces

What do all of these systems have in common?

They are **standard interfaces**: A universal method of connecting different systems together

# 📀 Standard Interfaces For Data

In the world of software, we actually have standard interfaces for how we transfer and store data between different applications.

For example, how would you share the results that a C program outputs with a Javascript program?...

# 📀 Standard Interfaces For Data

In the world of software, we actually have standard interfaces for how we transfer and store data between different applications.

For example, how would you share the results that a C program outputs with a Javascript program?...

**Use a data interchange format**

# 📀 Data Interchange Formats

To enable this standard interface for data, we have **data interchange formats**. These are basically extremely simple markup languages that are collections of arrays, objects, numbers, strings, booleans.

- Three main interchange formats we will talk about:
  - JSON
  - YAML
  - XML

#  JSON

JavaScript Object Notation (JSON) - TFC 7159

A format made up of braces for objects, square brackets for arrays, where all non-numeric items must be wrapped in quotations. Very similar to javascript data structures.

# 🏠 JSON

Let's represent a structure that contains a list of locations, where each location has a suburb and postcode:

```
{
  "locations": [
    {
      "suburb" : "Kensington",
      "postcode" : 2033
    },
    {
      "suburb" : "Mascot",
      "postcode" : 2020
    },
    {
      "suburb" : "Sydney CBD",
      "postcode" : 2000
    }
  ]
}
```

It's very similar to normal Javascript, except:

- No trailing commas allowed
- Object keys must be strings and include apostrophes

#  JSON

Most languages have capabilities either built-in or via libraries to write and read json.

Most of the time these libraries are responsible for converting between Javascript-based data structure and a stringified / text-based dump of JSON.

# JSON Code

```
1  import fs from 'fs';
2
3  const DATA_STRUCTURE = {
4    names: [
5      {
6        first: 'Bob',
7        last: 'Carr',
8      },
9      {
10       first: 'Julia',
11       last: 'Gillard',
12     },
13     {
14       first: 'Ken',
15       last: 'Henry',
16     },
17   ],
18 };
19
20 const data = JSON.stringify(DATA_STRUCTURE);
21 fs.writeFileSync('export.json', data, { flag: 'w' });
```

3.1_json_it.js

# JSON Code

```
1 import fs from 'fs';
2
3 const json = fs.readFileSync('export.json', { flag: 'r' });
4 const data = JSON.parse(json);
5 console.log(data);
```

3.1_unjson_it.js

# 👶 YAML

*YAML Ain't Markup Language* (YAML) is a popular modern interchange format due it's ease of editing and concise nature. It's easy to convert between JSON and YAML online.

```yaml
---
locations:
- suburb: Kensington
  postcode: 2033
- suburb: Mascot
  postcode: 2020
- suburb: Sydney CBD
  postcode: 2000
```

- Unlike javascript, indentation matters
- A dash is used to begin a list item
- Very common for configuration(s)

# 👵🏽 XML

eXtensible Markup Language (XML) is more of a legacy interchange format being used less and less

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root>
   <locations>
      <element>
         <postcode>2033</postcode>
         <suburb>Kensington</suburb>
      </element>
      <element>
         <postcode>2020</postcode>
         <suburb>Mascot</suburb>
      </element>
      <element>
         <postcode>2000</postcode>
         <suburb>Sydney CBD</suburb>
      </element>
   </locations>
</root>
```

#  XML

Issues with XML include:

- More verbose (harder to read at a glance)
- More demanding to process/interpret
- More bytes required to store (due to open/closing tags)

While you will find very few modern applications choose to use XML as an interchange format, many legacy systems will still use XML as a means of storing data

# 🦻 Feedback

Or go to the form here.