JSON
(JavaScript Object Notation)

CMT220
Databases & Modelling

Cardiff School of **Computer Science & Informatics**

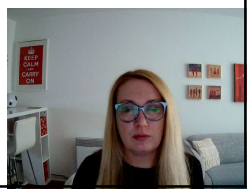
http://www.cs.cf.ac.uk

1

1

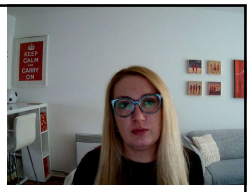
Lecture

- in the previous two lectures we learnt about XML, a markup language used to structure data
- we pointed that XML format is "fat" and briefly mention JSON as a "slim" format that does the same job
- in this lecture we will learn more about JSON



2

JSON

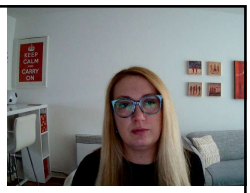


- JSON = **J**ava**S**cript **O**bject **N**otation
- pronounced like the name Jason
- JSON is a syntax for storing and exchanging data
  - text–based
  - light–weight
  - human readable
  - language independent
- JSON is an open standard specified on RFC4627
  - https://www.ietf.org/rfc/rfc4627.txt

3

3

History



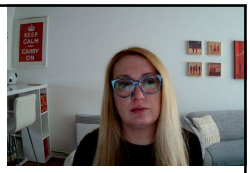
- JavaScript is a high–level, dynamic, untyped and interpreted programming language
- alongside HTML and CSS, JavaScript is one of the three essential technologies of the Web
- programmers need an easy way to transfer data on the Web
- JSON format is syntactically identical to the code for creating JavaScript objects
- instead of using a parser (like XML does), JavaScript can use standard functions to convert JSON data into native objects, e.g. `var json = JSON.parse(text);`

object string

4

4

JSON string

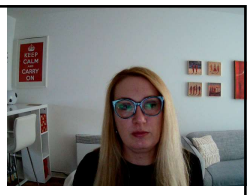


```
            key        value
{ <------------------------------------------------------------ object starts
    "name": "David Jones", <---------------------- value: string
    "age": 23, <------------------------------------ value: number
    "address": { <---------------------------------- object starts
        "streetAddress": "5 The Parade",
        "city": "Cardiff"
    }, <-------------------------------------------- object ends
    "phoneNumber": [ <----------------------------- array starts
        { <------------------------------------------ object starts
            "type": "home",
            "number": "029 1234 5678"
        }, <---------------------------------------- object ends
        { <------------------------------------------ object starts
            "type": "mobile",
            "number": "077 8765 4321"
        } <----------------------------------------- object ends
    ] <--------------------------------------------- array ends
} <---------------------------------------------------- object ends
```

5

5

JSON object in JavaScript



```
<script>
    var text = '{"name": "David Jones", "age": 23,
    "address": {"street": "5 The Parade", "city":
    "Cardiff"}, "phoneNumber": [{"type": "home",
    "number": "029 1234 5678"}, {"type": "mobile",
    "number": "077 8765 4321"}]}';

     var json = JSON.parse(text);

     alert(json.name);                  // David Jones
     alert(json.address.street);        // 5 The Parade
     alert(json.address.city);          // Cardiff
     alert(json.phoneNumber[0].number); // 029 1234 5678
     alert(json.phoneNumber[1].type);   // mobile
</script>
```

- try it online here

6

6

JSON and JavaScript

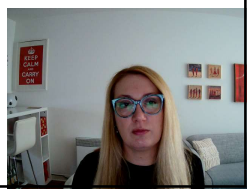
- JSON is considered as a subset of JavaScript
  - ... but that does not mean that JSON cannot be used with other languages
- JSON uses JavaScript syntax, but the JSON format is text only... just like XML
- JSON is language independent
  - it works well with most of the modern programming languages
  - e.g. PHP, Perl, Python, Ruby, Java and many more

7

7

JSON on the Web

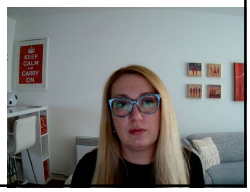
- serialization is the process of converting an object into a format suitable to be stored in a file or memory buffer and/or transmitted
- JSON is often used to serialize and transfer data over a network connection
  - e.g. between web server and a web application
  - note: XML serves the same purpose!
- Web services and APIs use JSON format to provide public data
  - e.g. Flickr and Twitter



8

JSON syntax

- JSON syntax is derived from JavaScript object notation syntax:
  - data is in name/value pairs "name":"value"
  - data is separated by commas ,
  - curly braces hold objects { object }
  - square brackets hold arrays [array]

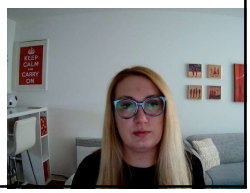


9

JSON data

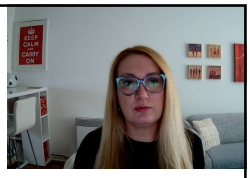
- JSON data are written as name/value pairs
- a name/value pair consists of:
  1. field name (in quotes)
  2. colon
  3. value
- e.g. "firstName":"John"

name colon value



10

JSON data



- JSON values can be of the following data types:

Type	Description	Example
number	double–precision floating–point format in JavaScript	{"marks": 97}
string	double–quoted Unicode with backslash escaping	{"name": "John"}
Boolean	true or false	{name: "John", marks: 97, distinction: true}
object	an unordered collection of key:value pairs	{name: "John", marks: 97, distinction: true}
array	an ordered sequence of values	{ "books": [{"title":"Game" }, {title":"Set"}, {"title":"Match"}] }
null	empty	

11

11

JSON data

- JSON objects are written inside curly braces
  - just like JavaScript, JSON objects can contain multiple name/values pairs, e.g.

```
{"firstName":"John", "lastName":"Doe"}
```

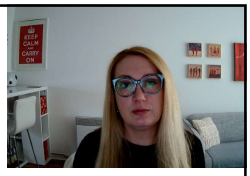
- JSON arrays are written inside square brackets
  - just like JavaScript, a JSON array can contain multiple objects, e.g.

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

12

12

JavaScript



- JSON syntax is derived from JavaScript object notation
- in JavaScript, an array of objects can be created like this:

```
var employees = [
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
];
```

- an element of the JavaScript object array can be accessed like this:

```
employees[0].firstName + " " +
employees[0].lastName;
```

Try it yourself »

- or

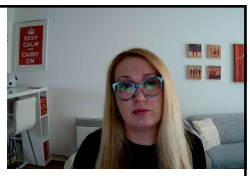
```
employees[0]["firstName"] + " " +
employees[0]["lastName"];
```

Try it yourself »

13

13

JavaScript



- an element of the JavaScript object array can be modified like this:

```
employees[0].firstName = "Gilbert";
```

Try it yourself »

- or

```
employees[0]["firstName"] = "Gilbert";
```

Try it yourself »

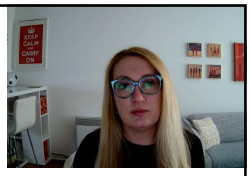
- result:

```
var employees = [
    {"firstName":"Gilbert", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
];
```

14

14

JSON within JavaScript



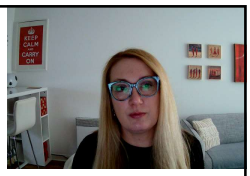
- JSON syntax is derived from JavaScript object notation
- very little extra software is needed to work with JSON within JavaScript
- JSON is commonly used to read data from a web server, and display the data in a web page
- for simplicity, we will demonstrate such use with a JSON string as input (instead of a file):

```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter", "lastName":"Jones" } ]}';
```

15

15

JSON within JavaScript



- the JavaScript function `JSON.parse()` can be used to convert a JSON string into a JavaScript object:

```
var obj = JSON.parse(text);
```

- the new JavaScript object can now be used in the web page, e.g.

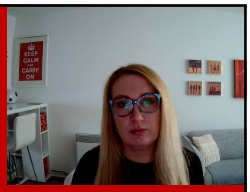
```
<p id="demo"></p>

<script>
  document.getElementById("demo").innerHTML =
  obj.employees[1].firstName + " " +
  obj.employees[1].lastName;
</script>
```

Try it yourself »

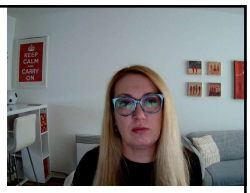
16

16



JSON Schema

17



Example

schema

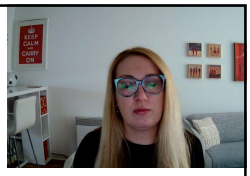
```
{
  "title": "Example Schema",
  "type": "object",
  "properties": {
    "firstName": {"type":"string"},
    "lastName": {"type":"string"},
    "age": {
      "description":"Age in years",
      "type":"integer",
      "minimum":0
    }
  },
  "required": ["firstName", "lastName"]
}
```

JSON

```
{"firstName":"Peter","lastName":"Pan","age":12}
```

18

JSON Schema

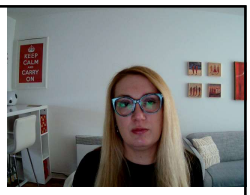


- JSON Schema is a specification for JSON–based format for defining the structure of JSON data
- JSON Schema itself is written in JSON
- schema is data itself, not a computer program
- it is just a declarative format for "describing the structure of other data"
- JSON data can be validated against a schema using a computer program
- for documentation see: http://json-schema.org/

19

19

Hello, World!



- in JSON Schema, an empty object is a valid schema that will accept any valid JSON, e.g.

```
{ }
```

- accepts any valid JSON, e.g.

```
42
```

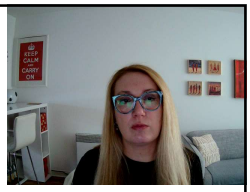
```
"I'm a string"
```

```
{ "an": [ "arbitrarily", "nested" ], "data": "structure" }
```

20

20

The type keyword



- the most common thing to do in a JSON Schema is to restrict to a specific type, e.g.

```
{ "type": "string" }
```

- only strings are accepted, e.g.

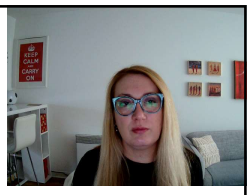
```
"I'm a string"
```

```
42
```

21

21

Declaring a JSON Schema



- JSON Schema is itself JSON
- it is not always easy to tell when something is JSON Schema or just JSON
- the `$schema` keyword is used to declare that something is JSON Schema, e.g.

  { "$schema": "http://json-schema.org/schema#" }

- it is generally good practice to include it, though it is not required

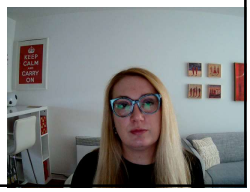
22

22

Declaring a unique identifier

- it is also good practice to include an `id` property as a unique identifier for each schema, e.g.

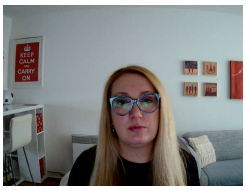
  { "id": "http://yourdomain.com/schemas/myschema.json" }



23

Metadata

```
{
  "title" : "Match anything",
  "description" : "This is a schema that matches anything.",
  "default" : "Default value"
}
```



- JSON Schema includes keywords: `title`, `description` and `default`
- not used for validation
- used to describe parts of a schema
- title will provide a short description
- description will provide a more lengthy explanation about the purpose of the data described by the schema
- neither are required, but they are encouraged as good practice
- default specifies a default value for an item

24

24

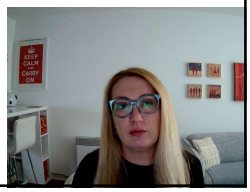
Enumerated values

- the enum keyword is used to restrict a value to a fixed set of values
- it must be an array with at least one element, where each element is unique, e.g.

```
{
  "type": "string",
  "enum": ["red", "amber", "green"]
}
```

```
"red"
```

```
"blue"
```



25

Enumerated values

- enum can be used without a type, to accept values of different types, e.g.

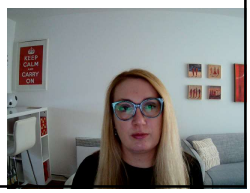
```
{
  "enum": ["red", "amber", "green", null, 42]
}
```

```
"red"
```

```
null
```

```
42
```

```
0
```



26

Combining schemas

- JSON schemas can be combined
- this does not necessarily mean combining schemas from multiple files
- it may be as simple as allowing data to be validated against multiple criteria
- `anyOf` is used to say that the given data may be valid against any of the given sub–schemas
- as long as a value validates against any of the sub–schemas, it is considered valid against the entire combined schema

```
{
  "anyOf": [
    { "type": "string", "maxLength": 5 },
    { "type": "number", "minimum": 0 }
  ]
}
```

"short" ✔

"too long" ✘

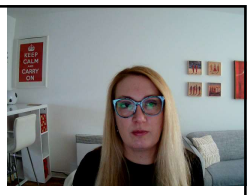
12 ✔

-5 ✘

27

27

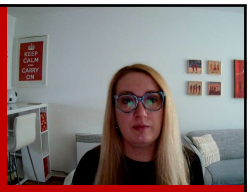
Combining schemas



- keywords used to combine schemas are:
  - `allOf` must be valid against all sub–schemas
  - `anyOf` must be valid against any sub–schema
  - `oneOf` must be valid against exactly one of the sub–schemas
- these keywords must be set to an **array**, where each **item** is a **schema**
- in addition, there is:
  - `not` must not be valid against the given schema

28

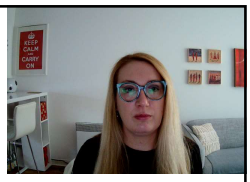
28



JSON vs. XML

29

JSON vs. XML



JSON

```
{"students":[
  {"name":"John", "age":"23", "city":"Cardiff"},
  {"name":"Steve", "age":"28", "city":"Swansea"},
  {"name":"Peter", "age":"32", "city":"Bristol"},
]}
```

notice how the use an array removes the need for the nested "element"

XML

```
<students>
  <student>
    <name>John</name> <age>23</age> <city>Cardiff</city>
  </student>
  <student>
    <name>Steve</name> <age>28</age> <city>Swansea</city>
  </student>
  <student>
    <name>Peter</name> <age>32</age> <city>Bristol</city>
  </student>
</students>
```

30

30

JSON vs. XML

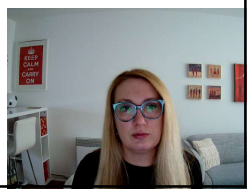
- **similarities**
  - both are self–describing (human–readable)
  - both are hierarchical (values within values)
  - both can be parsed and used by many programming languages
  - both can be fetched with an XMLHttpRequest
- **differences**
  - JSON does not use end tag
  - JSON is shorter
  - JSON is quicker to read and write
  - JSON can use arrays
- **biggest difference**
  - XML has to be parsed with an XML parser
  - JSON can be parsed by a standard JavaScript function

31

31

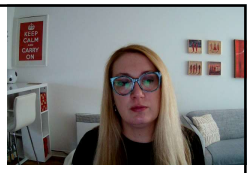
JSON vs. XML

- for AJAX applications, JSON is faster and easier than XML
- using XML
  1. fetch an XML document
  2. use the XML DOM to traverse through the document
  3. extract values and store in variables
- using JSON
  1. fetch a JSON string
  2. JSON.Parse the JSON string



32

JSON vs. XML



XML	JSON
there are several specifications to define schema for XML, e.g. DTD and XML Schema	JSON Schema does the same for JSON, but it is not as widely used
for selecting specific parts of an XML document, there is standard specification called XPath	JSONPath does the same for JSON, but is not as widely used
XML has XQuery specification for querying XML data	JSON has JAQL, JSONiq etc, but they are not as widely used
XML has XSLT specification, which may be used to apply style to an XML document	JSON does not have any such thing

33