FRESNO STATE
Discovery. Diversity. Distinction.

Web Programming (CSci 130)

Department of Computer Science
College of Science and Mathematics
California State University Fresno
H. Cecotti

Learning outcomes

- JSON
  - What is it?
  - What is it for?
  - How to use it?
- Common points and differences with XML

4 weeks before Halloween

- **JSON**:
  - **J**avaScript **O**bject **N**otation

```
{
    "fruit": "Apple",
    "size": "Large",
    "color": "Red"
}
```

The rationale

- In the early 2000s
  - Need of a **data** format for real time server to browser communication
    - Without using plugins
- Javascript → object based messaging format
- JSON.org = 2002
- JSON
  - Thought as a **subset** of Javascript
    - Can work with other formats
  - Popularized by Douglas Crockford
- Topic:
  - **Data**, not Document !

Why to use JSON

- Text format only
  - Easily sent **to** and **from** a server
    - Functions to parse JSON are in Javascript **and** PHP
  - Data format for any programming language
- Transformations
  - JSON → Object
  - Object → JSON
  - String → Object
- Information
  - From a table in a database (row = object with properties)
  - Row to JSON to Object → access/modification of properties

Syntax and structure

- JSON (Javascript Object Notation)
  - Lightweight data-interchange format
    - Open-standard file format
    - Text format → easy for humans to read and write
      → easy for the computer to parse and generate
  - Based on a subset of Javascript
  - Language independent
    - Conventions from C/C++/C#/Java…
- JSON structures
  - A collection of name/value pairs
    - Example in other languages: object, record, struct
      - Relationship between entities
  - An ordered list of values
    - Example in other languages: array, list, vector
      - Order between entities

Syntax and structure

- JSON
  - For asynchronous browser-server communication
    - As a replacement for XML in some systems
  - File extension .json
- JSON requires:
  - double quotes to be used around strings and property names.
  - Single quotes are not valid.
- Example:

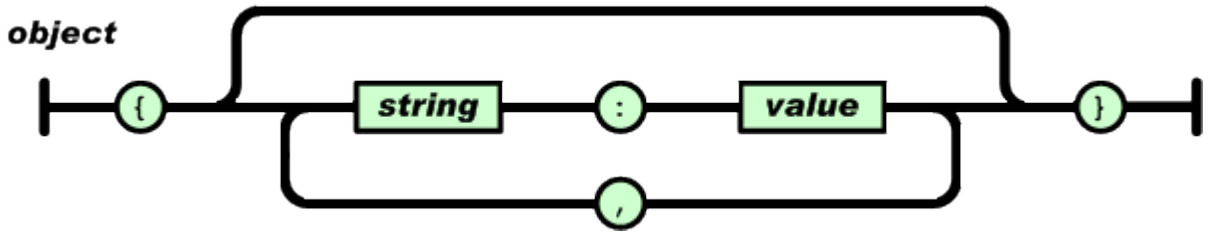
```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Syntax and structure

- A single misplaced comma or colon
  - ➢JSON file go wrong and not work ☹
  - ➢Be careful to validate any data you are attempting to use
    - Computer-generated JSON is less likely to include errors
    - Validate JSON using an application (e.g. JSONLint).
- JSON can take the form of any data type
  - ➢that is valid for inclusion inside JSON (not just arrays or objects).
    - A single string or number would be a valid JSON object.
- JavaScript:
  - ➢Object properties may be unquoted
- JSON:
  - ➢Only quoted strings may be used as properties.

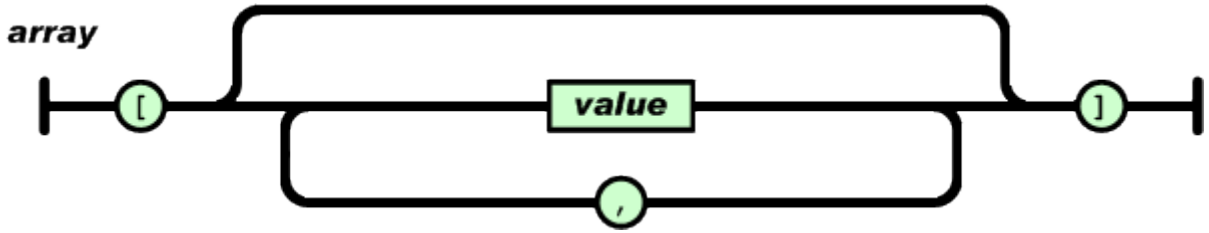
JSON object

- Object
  - Unordered set of name/values pairs
  - Syntax
    - Starts with {
    - Ends with }
    - Name : value
    - Separation of each pair with a comma



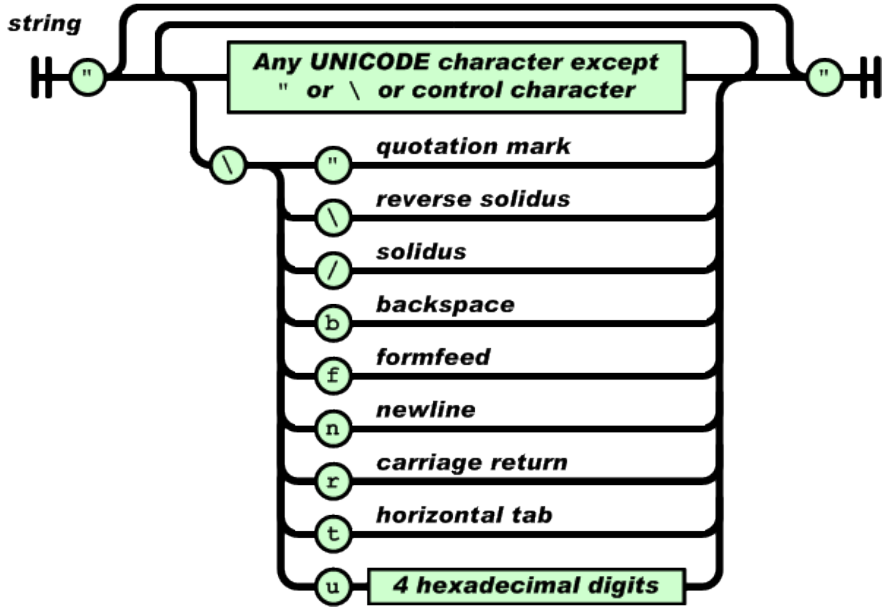
JSON array

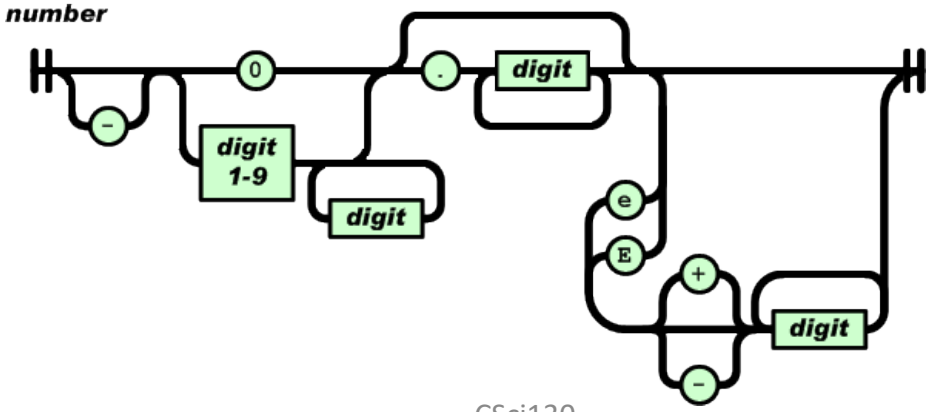
- Array
  - Ordered collection of values
  - Syntax
    - Starts with [
    - Ends with]
    - Values separated with a comma ,



JSON value

- Values
  - String (“ ”)
    - Special characters
  - Number
  - Object
  - Array
  - Boolean
    - True/False
  - null





About formats …

- JSON vs XML
  - Not comparable
  - JSON
    - More compact and readable (less data is transferred , no tags)
  - Yet, they are both
    - self describing
    - Hierarchical
    - Used/parsed in many programming languages
    - Fetched to an XMLhttprequest (objects to interact with servers)
- XML → document (language to describe documents)
  - Benefits of the definition of the structure (XML Schema)
  - Special features: Xpath, XSL,…
- JSON → structured data (data format)

About formats…

- JSON
  - Simple syntax
    - Less markup overhead compared to XML
  - Easy to use with Javascript
    - Markup subset of JS object literal notation
    - Same basic data types as Javascript
  - JSON schema for description and datatype and structure validation (like in XML)
  - Not all the data types are supported

Examples

- JSON

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

- XML

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Examples

- File parse into a var superHeroes
- To access the 3rd superpower of the second hero

listed in the members list:

  - ➢ superHeroes['members'][1]['powers'][2]
    - (from the Mozilla dev page)

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

Access the JSON file

- At this stage:
  - Get a file locally
  - In the next weeks:
    - Server + request to the files in the server (POST/GET)
      - E.g. XMLHttpRequest();
- How to read file in Javascript with HTML5
  - File API specification
    - https://www.w3.org/TR/file-upload/
    - Some example
      - https://www.html5rocks.com/en/tutorials/file/dndfiles/
- See example on Canvas:
  - JSON file → Object → Presentation in the web page
  - File: load_jsonfile.html

Example

- Canvas:
  - **Example 1**: Simple jSON test
    - class_javascript_json_example01.js and class_javascript_json_01.html
  - **Example 2**: Read a jSON file locally and display its content in HTML (DOM) with CSS.
    - The jSON file: superheroes.json and the HTML file: load_jsonfile.html
- Need of callbacks

```
<!DOCTYPE html>
<html lang="en">
<head>
<title> CSci130 - Callback function example</title>
<head>
<body>
<script>
function someEvent(x,y,callbackFunction) {
    return callbackFunction(x, y);
}
function calcProduct(x,y) {
    return x * y;
}
function calcSum(x,y) {
    return x + y;
}
// 5*4=20
alert(someEvent(5,4,calcProduct));
// 5+5=10
alert(someEvent(5,5,calcSum));
</script>
</body>
</html>
```

Javascript functions

- Parse example: **JSON.parse**
  - var myp = document.createElement('P');
  - document.body.appendChild(myp);
  - myp.innerHTML=' ';
  - **var obj1 = JSON.parse('{ "name":"Alfredo", "age":21, "city":"Madera"}');**
  - document.getElementById("demo1").innerHTML = obj1.name + ", " + obj1.age + " living in " + obj1.city;
- Stringify example: **JSON.stringify**
  - var myp = document.createElement('P');
  - document.body.appendChild(myp);
  - myp.innerHTML=' ';
  - var obj2 = { "name":"Rodrigo", "age":24, "city":"Hanford"};
  - **var myJSON = JSON.stringify(obj2);**
  - document.getElementById("demo2").innerHTML = myJSON;

Conclusion

- JSON
  - A lightweight alternative to XML
  - Useful when you want to transmit data across a network
- Just a **data** format
  - Only properties, no methods,
  - But very useful
    - We will use it to move data
      - From the client to the server
      - From the server to the client
- For more examples
  - http://www.json.org/
- Next class:
  - jQuery