Advanced JavaScript

JSON

ITI – Assiut Branch
Eng. Hany Saad

What is JSON?

- Stands for (**J**ava**S**cript **O**bject **N**otation).
- It is a lightweight data-interchange format.
- It is not a markup language, not scripting or programming language.
- JSON uses JavaScript syntax, but the JSON format is text only, just like XML.Text can be read and used as a data format by any programming language.
- Text-based.
- It can be used instead of XML.
- It is a special object notational construct, that is a subset of JavaScript.
- It is a text format that is completely <u>language independent</u> but uses conventions that are familiar to programmers of the C-family of languages.

JSON Features

- ❑ lightweight data-interchange format.
- ❑ Easy for humans to read and write.
- ❑ JSON is "self-describing" and easy to understand.
- ❑ Easy for machines to parse and generate.
- ❑ JSON Supports a lot of programming languages such as: ActionScript, C, C#, ColdFusion, E, Java, JavaScript, ML, Objective CAML, Perl, PHP, Python, Rebol, Ruby, and Lua.
- ❑ It has support for Unicode, allowing almost any information in any human language to be communicated.

JSON Structure

- JSON is built on two structures:

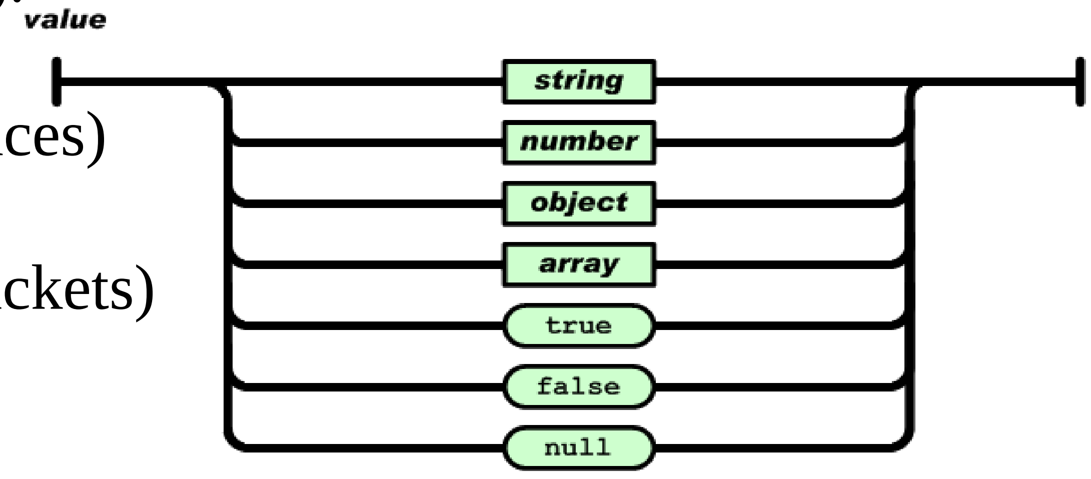
  1- A collection of name:value pairs, separated by (:). Name is string, value is one of JSON values. (Ex.: “Name” : “Ahmed”).

  2- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

JSON Values Formats

❑JSON supports these basic data types:

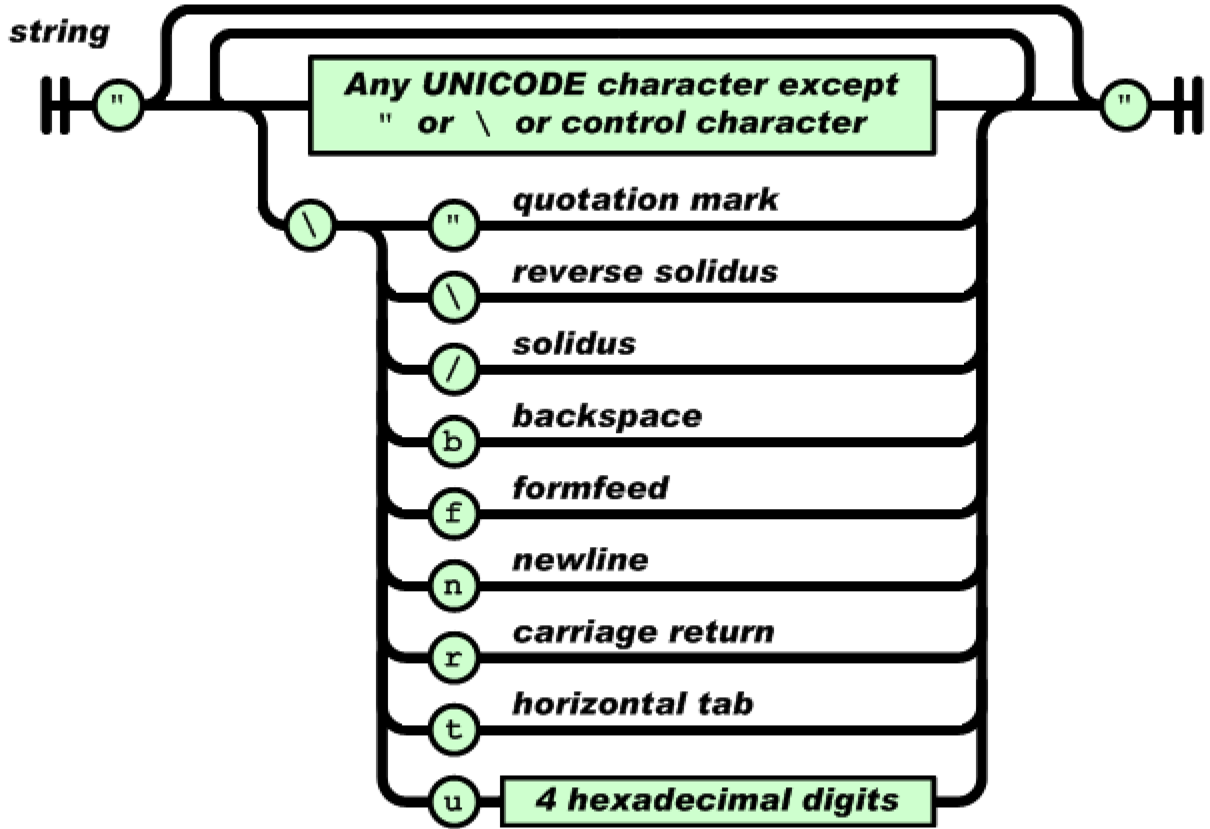
- Strings (in double quotes)
- Numbers (integer or floating point)
- Booleans (true, false).
- Objects (in curly braces)
- Arrays (in square brackets)
- null



JSON Data Values (Cont.)

1- String:

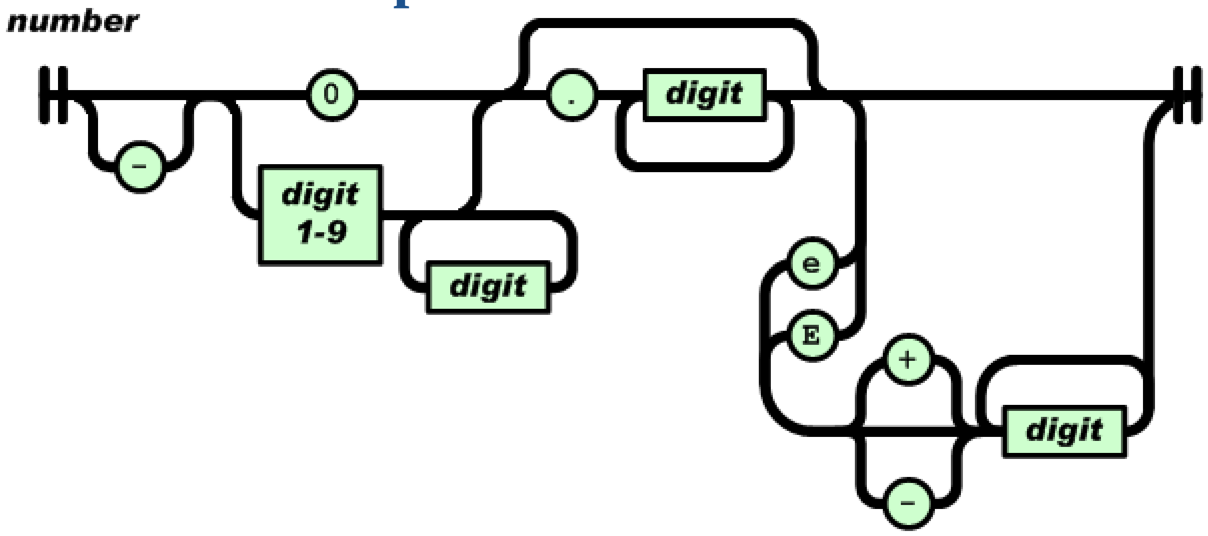
- Sequence of 0 or more Unicode characters.
- Wrapped in "double quotes“
- Example: **"Name": "Ahmed"**



JSON Data Values (Cont.)

2- Number:

- Can be: Integer, Real, float, Scientific.
- Can’t be: octal, hex (Use String instead), NaN or Infinity (Use Null instead).
- Example: **"rating": 123**
  "length": 122.234
  "Temp": -5
  "atoms per mole": 6.023e+23



JSON Data Values (Cont.)

3- Boolean:

- Example: **"active":true**
  "active":false

4- null:

- Example: **"email":null**

JSON Data Values (Cont.)

5- Object:

- Objects are unordered containers of key/value pairs
- Objects are wrapped in { }
- (,) separates key/value pairs
- (:) separates keys and values
- Keys are strings
- Values are JSON values

JSON Data Values (Cont.)

5- Object (Cont.):

- Example:

```
Var jObj=
{
        “id”:10,
        "company":"ITI" ,
        "city": “Assiut"
}
```

JSON Data Values (Cont.)

5- Object (Cont.):

- JavaScript code Example:

```
<script>
        var sObj= ‘{“ID”:10,"name":“Ali",“City":“Assiut“, “Job”: null,
“Married”: true}’;
        var jObj= JSON.parse(sObj);

        //OR
         var jobj= {ID:10,name:“Ali",City:“Assiut“, Job: null, Married:
true};
        //when embded in JS code, keys aren’t qouted.
        document.write(jobj.ID);
        document.write(jobj.name);
        document.write(jobj.Job);
</script>
```

JSON Data Values (Cont.)

5- Array:

- Arrays are ordered sequences of values
- Arrays are wrapped in []
- (,) separates values

JSON Data Values (Cont.)

5- Array (Cont.):

- Array of string:

```
var Arr=["Ahmed", "Ali", "Emad"]
```

- Array of Numbers:

```
var Arr =[30, 20, 10]
```

- Array of different Data Types:

```
var Arr =["Ahmed", 10, null, true]
```

- Array of Objects:

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

JSON Data Values (Cont.)

5- Array (Cont.):

- Example:

```
<script>
  //Array of one type
   var stdArr=[“Ahmed”,”Ali”,”Emad”];
   for (i=0; i<3; i++)
   {
       document.write(stdArr[i]);
   }
  //Array with different types
  var Jarr=["red", 1, null, true];
  document.write(Jarr[2]);
  //Nested array
  var Jarr2=["red",1,["a", "b", "c"], false];
  document.write(Jarr2[2][1]);
  //Object in an array
  var Jarr3=[“Ahmed", {Grade:“A", Mark:20}, 2];
  document.write(Jarr 3[1].Grade);
</script>
```

JSON vs. XML

- Much Like XML:
  - Both JSON and XML is plain text
  - Both JSON and XML is "self-describing" (human readable)
  - Both JSON and XML is hierarchical (values within values)
  - Both JSON and XML can be fetched with an HttpRequest

- Much Unlike XML:
  - JSON doesn't use tags
  - JSON is shorter
  - JSON is quicker to read and write
  - JSON can use arrays

- The biggest difference: XML has to be parsed with an XML parser, JSON can be parsed by a standard JavaScript function.

JSON vs. XML (Cont.)

- **JSON supports these basic data types:**
  Strings, Numbers, Booleans (true, false), Objects, Arrays, null

- **JSON Doesn't Have Namespaces:**
  Every object is a namespace. Its set of keys is independent of all other objects, even exclusive of nesting.

- **JSON Has No Validator:**
  Ultimately, every application is responsible for validating its inputs.

- **JSON Is Not Extensible:**
  It does not need to be. New fields can be added to existing structures without obsoleting existing programs.

JSON vs. XML (Cont.)

- **JSON Is Versionless:**

  JSON is very stable. No revisions to the JSON grammar are anticipated.

- **JSON Is Not XML:**

  JSON(Object, Array, String, number, Null).

  XML(element, Attribute, attribute string, Content, Entities, Declarations, Schema, Version, namespace).

Why JSON over XML?

❑ **Why JSON over XML?**

- **Lighter and faster than XML as on the wire data format.**
- **JSON objects are typed while XML data is typeless.**
  - **JSON types:** string, number, array, boolean, object.
  - **XML data:** are all string
- **Native data form for JavaScript code**

  - Data is readily accessible as JSON objects in your JavaScript Code vs. XML data needed to be parsed and assigned to variables.

  - Retrieving values is as easy as reading from an object property in your JavaScript code

Why JSON over XML? (Cont.)

❑**For AJAX applications, JSON is faster and easier than XML:**

❑ **Using XML:**

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

❑ **Using JSON:**

- Fetch a JSON string
- JSON.Parse the JSON string

JSON Syntax vs. XML Syntax

❑XML Data representation:

```
<wclass>
   <!--My students who took web programming class
   with me-->
   <student id="1">
      <name>Linda Jones</name>
      <legacySkill>Access, VB5.0</legacySkill>
   </student>
   <student id="2">
      <name>Adam Davidson</name>
      <legacySkill>Cobol, MainFrame</legacySkill>
   </student>
</wclass>
```

JSON Syntax vs. XML Syntax (Cont.)

❑JSON Data representation:

```
<SCRIPT LANGUAGE="JavaScript">
    var webclass =
    {wclass:
        [
            {student:
                {
                    id:1,
                    name:"Linda Jones",
                    legacySkill:["Access″, ‶VB 5.0‶]
                }
            },

            {student:
                {
                    id:2,
                    name:"Adam Davidson",
                    legacySkill:["Cobol″, ‶MainFrame‶]
                }
            }
        ]
    };

    alert(webclass.wclass[1].student.legacySkill[0]);
    // Cobol
</SCRIPT>
```

Parsing JSON string

- You can write JSON objects and arrays directly as objects and arrays Directly in JavaScript:

```
<script>
        var obj= {ID:10,name:“Ali",City:“Assiut“, Job: null, Married: true};
        document.write(obj.ID);
        document.write(obj.name);
</script>
```

- You need to parse it to JSON when you read it from text file:

```
<script>
        var jsonStr= ‘{“ID”:10,"name":“Ali",“City":“Assiut“, “Job”: null,
“Married”: true}’;
        var obj=JSON.parse(jsonStr);
        document.write(obj.ID);
        document.write(obj.name);
</script>
```

Parsing JSON string (Cont.)

- For old browsers that don’t support the JavaScript function JSON.parse() can use the eval() function:

```
<script>
        var jsonStr= ‘{“ID”:10,"name":“Ali",“City":“Assiut“, “Job”: null,
“Married”: true}’;
        var obj=eval ("(" + text + ")");
        document.write(obj.ID);
        document.write(obj.name);
</script>
```

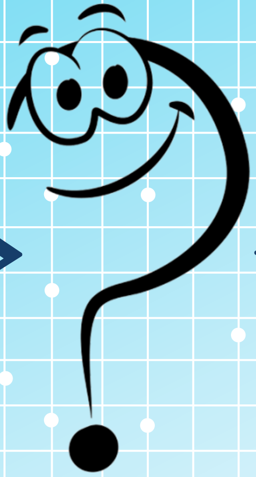
How to read JSON files from the server?

- The file type for JSON files is ".json" (or even .txt files), it's just text files.
- **JSON Http Request**: A common use of JSON is to read data from a web server, and display the data in a web page is using Http Request (XMLHttpRequest Object).
- Http Request is also a technique used in Ajax.

JSON Web resources and References

http://www.json.org

http://www.w3schools.com/json/



<SCRIPT > </SCRIPT>

<script>document.writeln(“Thank You!”)</script>