Session 8

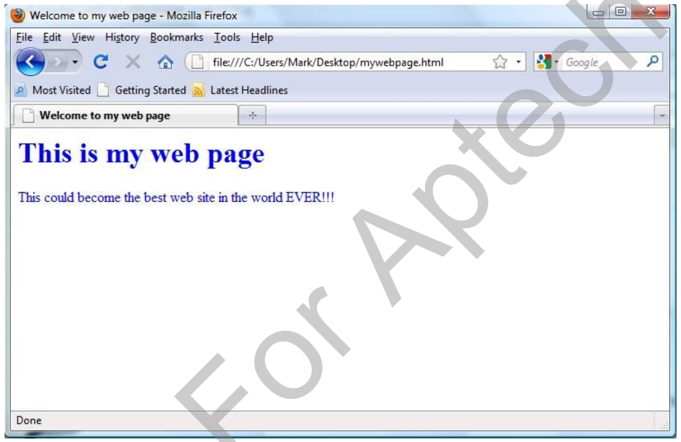
Introduction to JSON

Objectives

- Define and describe JavaScript Object Notation (JSON)
- Explain the uses of JSON
- List and describe the features of JSON
- Compare JSON with XML
- Describe the advantages of JSON
- Explain syntax and rules
- Explain literal notation in JavaScript and conversion to JSON
- Explain how to create a simple JSON file

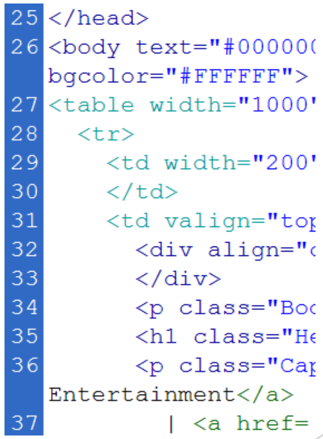
Introduction: The Need of JSON

- XML parses using Document Object Model (DOM) with complex processes and no cross-browser compatibility.
- JavaScript engine incorporates a markup language into an HTML Web page.
- The need of more natural format native to this engine and overcome DOM limitations was felt.


Welcome to my web page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///C:/Users/Mark/Desktop/mywebpage.html
Most Visited Getting Started Latest Headlines
Welcome to my web page
Done

This is my web page

This could become the best web site in the world EVER!!!



```
</head>
<body text="#00000(
bgcolor="#FFFFFF">
<table width="1000'
  <tr>
    <td width="200'
    </td>
    <td valign="toı
      <div align="c
      </div>
      <p class="Boc
      <h1 class="He
      <p class="Caı
Entertainment</a>
          | <a href=
```

What is JSON

- A lenient text format derived from JavaScript language and ECMAScript.
- Platform-as well as language-independent.
- Implements conventions of the C-based languages.
- More concise than XML.

JSON is a lightweight, text-based, and open standard data-interchange format.

Uses of JSON

- For serializing data to be sent over a network
- For facilitating communication between incompatible platforms
- For developing JavaScript based applications
- For sending public data through Web services and Application Programming Interfaces (APIs)
- For fetching data without hard coding in HTML

History of JSON

Coining the Term
- Originally stated by Douglas Crockford
- Coined initially at the State Software Company

The Co-founders' Decision
- To provide an abstraction layer for devising stateful Web applications without plugins

Participation by Other Companies
- Implementing frames to send information to the browsers without refreshing the page

Crockford's New Discovery
- Implementing JavaScript as an object-based format for messaging
- JSON.org in 2002

Features of JSON 1-2

- Standard structure
- Simplicity
- Open-source
- Self-describing
- Lightweight
- Scalable/Reusable

Features of JSON 2-2

- Clean data
- Efficiency
- Interoperability
- Extensibility
- Internationalization

JSON versus XML 1-3

Characteristic	JSON	XML
Data Types	Offers scalar data types.	Does not offer any idea of data types.
Array Support	Provides native support.	Expresses arrays by conventions.
Object Support	Provides native support.	Expresses objects by conventions.
Null Support	Recognizes the value natively.	Mandates the use of `xsi:nil` on elements.
Comments	Does not support.	Provides native support (via APIs).

JSON versus XML 2-3

Characteristic	JSON	XML
Namespaces	Does not support namespaces.	Accepts namespaces to prevent name collisions.
Formatting	Is simple and offers more direct data mapping.	Is complex and needs more effort for mapping application types to elements.
Size	Has very short syntax.	Has lengthy documents.
Parsing in JavaScript	Needs no additional application for parsing.	Implements XML DOM and requires extra code for mapping text to JavaScript objects.
Parsing	Has JSONPath.	Has XPath specification.
Learning Curve	Is not steep at all.	Is steep.

JSON versus XML 3-3

Characteristic	JSON	XML
Complexity	Is complex.	Is more complex.
Schema (Metadata)	Has a schema but is not widely used.	Uses many specifications for defining a schema.
Styling	Has no special specification.	Has XSLT specification for styling an XML document.
Querying	Has specifications such as JSON Query Language (JAQL) and JSONiq but are not widely used.	Has XQuery specification that is widely used.
Security	Is less secure, as the browser has no JSON parser.	Is more secure.

JSON versus XML Coding

JSON

```
{
 "students":
  [
    {"name":"Steve",
     "age":"20",
     "city":"Denver"},

    {"name":"Bob",
     "age":"21",
     "city":"Sacramento"}
  ]
}
```

XML

```
<students>

 <student>

   <name>Steve</name>

   <age>20</age>

   <city>Denver</city>

 </student>

 <student>

   <name>Bob</name>

   <age>21</age>

   city>Sacramento</city>

 </student>
```

Advantages of JSON

- Quick serialization
- Efficient encoding
- Faster parsing than XML and Yaml Ain't Markup Language (YAML)
- Simpler to work with other languages
- Easy differentiation between data types

Limitations of JSON

- Difficult in reading the format and precision of keeping brackets in its place
- Following snippet illustrates complexity:

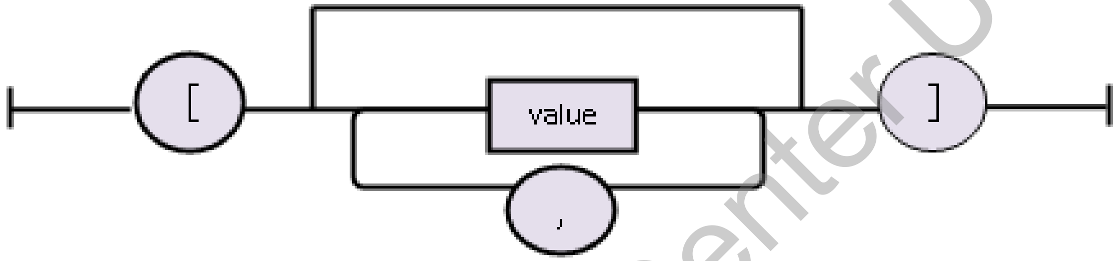
```
{
"problems":  [{
 "Diabetes":[{
 "medications":[{
 "medicationsClasses":[{
 "className":[{
  "associatedDrug":[{
 "name":"Asprin", "dose":"", "strength":"500 mg"
}]
 }]
}]
}]
}]
}]
}
```

JSON Data Structures

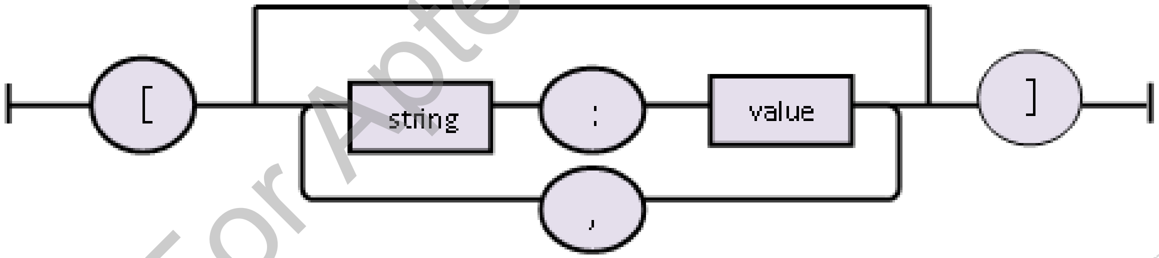
- Support for two universally-supported data structures:
  - Ordered list of values, which can be an array, vector, list, or sequence.
  - Unordered set of name/value pairs, which can be a keyed list, object, record, struct, or a hash table.

JSON Arrays and Objects

- Following figure represents an array (ordered set) in JSON:



- Following figure represents an object (unordered set) in JSON:



JSON Objects

- Following snippet represents a JSON object:

```
{
 "Name":"Janet George",
 "Street":"123 Ashley Street",
 "City":"Chicago",
 "Code":"60604"
}
```

JSON Syntax Rules

Arrays

- Always in square brackets

Objects

- Always in curly braces
- Always in name value pairs separated by commas, with name and value delimited by colon

Literal Notation in JavaScript

- Expresses fixed values literally.
- Is a floating-point number, integer, string, Boolean, array, or object.
- Contains zero or more expressions in a sequence in an array literal.
- Contains a list of members in an object literal.

JavaScript Array Literal Notation 1-2

- Following snippet defines an array using literal notation:

```
var fruits = ["Pineapple", "Orange", "Melon",
"Kiwi", "Guava"];

alert (fruits[4] + " is one of the " +
fruits.length + " fruits.");
```

JavaScript Array Literal Notation 2-2

- Following snippet defines an object using literal notation:

```
var Address = {
       "Street": "123 New Mansion Street.",
       "City": "Denver",
       "PostalCode": 80123  };
alert ("The product will be sent to postal code
   " + Address.PostalCode);
```

From JavaScript Literals to JSON

- JSON has stricter rules:
  - Object member name to be a valid JSON string in quotation marks
  - A member value or an array element in JSON to be confined to a restricted set
- JSON does not support Date/time literals.

Creating a JSON File

- The `JSON.stringify()` function converts a JavaScript value to a serialized JSON string.

- Following is the syntax:

```
JSON.stringify(value [, replacer] [,
space])
```

where,

`value` identifies a JavaScript value.

`replacer` represents a function or array.

`space` adds line break and white space.

Summary 1-2

- JSON is a text-based and open standard format derived from JavaScript and ECMAScript.
- JSON is lighter, more scalable, less complex, quicker to learn, and faster to process than XML.
- JSON works independently of any language or platform.
- JSON is preferred for serializing and de-serializing data to be exchanged on the Web.
- Data in JSON format is represented as name/value pairs.

Summary 2-2

- JSON supports two data structures namely, array as ordered list and objects as unordered set of name/value pairs.
- While syntax for JavaScript's literal values is flexible, JSON follows stricter rules for arrays and objects.
- A JSON file is saved with `.json` extension.
- The `JSON.stringify()` function is used to transform a JavaScript object to a JSON string.