JSON

Introduction

- JSON is a light-weight alternative to XML for data-interchange
- JSON = JavaScript Object Notation
- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.
  - It’s really language independent
  - most programming languages can easily read it and instantiate objects or some other data structure

Why use JSON?

- Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.

- JavaScript has a built in function to convert a string, written in JSON format, into native JavaScript objects:

- **JSON.parse()**

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Features of JSON

- **Easy to use** - JSON API offers high-level facade, which helps you to simplify commonly used use-cases.
- **Performance** - JSON is quite fast as it consumes very less memory space, which is especially suitable for large object graphs or systems.
- **Free tool** - JSON library is open source and free to use.
- **Doesn't require to create mapping** - Jackson API provides default mapping for many objects to be serialized.
- **Clean JSON** - Creates clean, and compatible JSON result that is easy to read.
- **Dependency** - JSON library does not require any other library for processing.

Exchanging Data

- When exchanging data between a **browser and a server**, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

Sending Data

- If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

```
<!DOCTYPE html>
<html>
<body>
<h2>Convert a JavaScript object into a JSON string, and send it to the server</h2>
<script>
        var myObj = { name: "John", age: 31, city: "New York" };
        var myJSON = JSON.stringify(myObj);
        window.location = "demo_json.php?x=" + myJSON;
</script>
</body>
</html>
```

Receiving Data

- If you receive data in JSON format, you can convert it into a JavaScript object:

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Storing Data

- When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats.
- JSON makes it possible to store JavaScript objects as text.

```
// Storing data:
myObj = {name: "John", age: 31, city: "New York"};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

JSON Syntax

- JSON syntax is derived from JavaScript object notation syntax:
  - Data is in name/value pairs
  - Data is separated by commas
  - Curly braces hold objects
  - Square brackets hold array

JSON Data - A Name and a Value

- JSON Data - A Name and a Value
- JSON data is written as name/value pairs.
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:
  - "name":"John“

JSON - Evaluates to JavaScript Objects

- The JSON format is almost identical to JavaScript objects.
- In JSON, *keys* must be strings, written with double quotes:
  - { "name":"John" }

JSON Values

- In **JSON**, *values* must be one of the following data types:
  - a string
  - a number
  - an object (JSON object)
  - an array
  - a boolean
  - null
- In **JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:
  - a function
  - a date
  - undefined
  - In JSON, *string values* must be written with double quotes:

JSON Values Examples

JSON

- { "name":"John" }

- In JavaScript, you can write string values with double *or* single quotes:

JavaScript

- { name:'John' }

JSON vs XML

- Both JSON and XML can be used to receive data from a web server.
- The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON	XML
JSON object has a type	XML data is typeless
JSON types: string, number, array, Boolean	All XML data should be string
Data is readily accessible as JSON objects	XML data needs to be parsed.
JSON files are more human-readable.	XML files are less human-readable.
JSON is supported by most browsers.	Cross-browser XML parsing can be tricky
JSON has no display capabilities.	XML provides a capability to display data because it is a markup language.
Retrieving value is easy	Retrieving value is difficult
Supported by many Ajax toolkit	Not fully supported by Ajax toolkit
A fully automated way of deserializing/serializing JavaScript.	Developers have to write JavaScript code to serialize/de-serialize from XML
Native support for object.	The object has to be express by conventions - mostly missed use of attributes and elements.

JSON Example

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

XML Example

-
  ```
  <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>
  ```

Why JSON is Better Than XML?

- XML is much more difficult to parse than JSON.
- JSON is parsed into a ready-to-use JavaScript object.

Application of JSON

- Here are some common applications of JSON:
  - ➢ Helps you to transfer data from a server
  - ➢ JSON format helps transmit and serialize all types of structured data.
  - ➢ Allows you to perform asynchronous data calls without the need to do a page refresh
  - ➢ Helps you to transmit data between a server and web applications.
  - ➢ It is widely used for JavaScript-based application, which includes browser extension and websites.
  - ➢ You can transmit data between the server and web application using JSON.
  - ➢ We can use JSON with modern programming languages.
  - ➢ It is used for writing JavaScript-based applications that include browser add-ons.
  - ➢ Web services and Restful APIs use the JSON format to get public data.

Disadvantages of JSON

- Here are few advantages of JSON:
- No namespace support, hence poor extensibility
- Limited development tools support
- No support for formal grammar definition