**Working with JSON (5)**  
JSON – Introduction,Need of JSON,JSON Syntax Rules,JSON Data - a Name and a Value,JSON Objects,JSON Arrays,JSON Uses JavaScript Syntax,JSON Files,JSON & Security Concerns, Cross Site Request Forgery (CSRF), Injection Attacks,JS XMLHttpRequest functions,JavaScript XMLHttpRequest & Web APIs,JSON & Client Side Frameworks,JSON & Server Side Frameworks,Replacing XML with JSON,JSON parsing,AJAX using JSON and jQuery

----------------------------------------------------------------------------------------------------------------------------------

**Introduction to JSON**

* JSON is a text-based data exchange format derived from JavaScript that is used in web services and other connected applications.
* JSON stands for Javascript Object Notation. JSON is a text-based data format that is used to store and transfer data.
* In the early 2000s, JSON was initially specified by Douglas Crockford. In 2013, JSON was standardized as ECMA-404, and RCF 8259 was published in 2017.

## JSON Data

JSON data consists of **key/value** pairs similar to JavaScript object properties. The key and values are written in double quotes separated by a colon **:**. For example,

// JSON data

"name": "John"

## JSON Syntax

* // JSON syntax

{

"name": "John",

*"age": 22,*

"gender": "male",

}

* JSON defines only two data structures: **objects and arrays**.
* An object is a set of name-value pairs, and an array is a list of values. JSON defines seven value types: string, number, object, array, true, false, and null.
* **Objects** are enclosed in braces ({}), their name-value pairs are separated by a comma (,), and the name and value in a pair are separated by a colon (:). Names in an object are strings, whereas values may be of any of the seven value types, including another object or an array. For Example:
* // JSON object

{ "name": "John", "age": 22 }

* **Arrays** are enclosed in brackets ([]), and their values are separated by a comma (,). Each value in an array may be of a different type, including another array or an object. When objects and arrays contain other objects or arrays, the data has a tree-like structure.

// JSON array

[ "apple", "mango", "banana"]

// JSON array containing objects

[

{ "name": "John", "age": 22 },

{ "name": "Peter", "age": 20 }.

{ "name": "Mark", "age": 23 }

]

**Accessing JSON Data**

We use the **.**notation to access JSON data. Its syntax is: **variableName.key.** For example,

// JSON object

const data = {

"name": "John",

"age": 22,

"hobby": {

"reading" : true,

"gaming" : false,

"sport" : "football"

},

"class" : ["JavaScript", "HTML", "CSS"]

}

// accessing JSON object

console.log(data.name); // John

console.log(data.hobby); // { gaming: false, reading: true, sport: "football"}

console.log(data.hobby.sport); // football

console.log(data.class[1]); // HTML

**You can also use square bracket syntax [] to access JSON data. For example,**

// JSON object

const data = {

"name": "John",

"age": 22

}

// accessing JSON object

console.log(data["name"]); // John

**Example JSON File**

#user.json

{

"name": "Steve",

"age": 43,

"isProgrammer" true,

"hobbies": ["Reading Java books", "cooking", "classic music"],

"friends": [{

"name": "joey",

"age": 39,

"isProgrammer": false,

"friends": [...]

}]

}

In the above example, we have a file called user.json. Inside the file we have different data types.

**Converting JSON to JavaScript Object**

You can convert JSON data to a JavaScript object using the built-in **JSON.parse()** function. For example,

// json object

const jsonData = ‘{ "name": "John", "age": 22 }';

// converting to JavaScript object

const obj = JSON.parse(jsonData);

// accessing the data

console.log(obj.name); // John

**Converting JavaScript Object to JSON**

You can also convert JavaScript objects to JSON format using the JavaScript built-in **JSON.stringify()** function. For example,

// JavaScript object

const jsonData = { "name": "John", "age": 22 };

// converting to JSON

const obj = JSON.stringify(jsonData);

// accessing the data

console.log(obj); // "{"name":"John","age":22}"

Example:

<!DOCTYPE html>

<html>

<head>

<title>JSON Example</title>

</head>

<body>

<script type="text/javascript">

let companies =

`[

{

"name": "Big corporate",

"numberOfEmployees": 1000,

"ceo": "Neil",

"rating": 3.6

},

{

"name": "Small startup",

"numberOfEmployees": 10,

"ceo": null,

"rating": 4.3

}

]`

console.log(JSON.parse(companies))

</script>

hello

</body>

</html>

**Example of JSON to JS**

<!DOCTYPE html>

<html>

<head>

<title>JSON Example</title>

</head>

<body>

<script type="text/javascript">

var person = {

name: "Brad",

age: 35

}

jsonPerson = JSON.stringify(person); //convert to JSON

console.log(jsonPerson.name); //undefined

jsPerson = JSON.parse(jsonPerson); //convert to JS Object

console.log(jsPerson.name); //Brad

</script>

</body>

</html>

**Uses of JSON**

JSON is often used as a common format to serialize and deserialize data in applications that communicate with each other over the Internet. These applications are created using different programming languages and run in very different environments. JSON is suited to this scenario because it is an open standard, it is easy to read and write, and it is more compact than other representations.

RESTful web services use JSON extensively as the format for the data inside requests and responses. The HTTP header used to indicate that the content of a request or a response is JSON data is **Content-type:application/json**

## Application of JSON

Here are some common applications of JSON:

* Helps you to transfer data from a server
* Sample JSON file format helps in 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.

## JSON vs. XML

Here is the prime difference between JSON vs. XML

|  |  |
| --- | --- |
| 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

Below is a simple JSON example:

{

"student": [

{

"id":"01",

"name": "Tom",

"lastname": "Price"

},

{

"id":"02",

"name": "Nick",

"lastname": "Thameson"

}

]

}

#### XML Example

<?xml version="1.0" encoding="UTF-8" ?>

<root>

<student>

<id>01</id>

<name>Tom</name>

<lastname>Price</lastname>

</student>

<student>

<id>02</id>

<name>Nick</name>

<lastname>Thameson</lastname>

</student>

</root>

**AJAX**