**JSON – JavaScript Object Notation**

**JSON:**

* Introduction:
* A syntax for storing & exchanging data
* Text written with JS object notation
* \*Often used when data is sent from a server to a web page
* Lightweight data-interchange format
* ‘Self-Describing’ & easy to understand
* Language independent – JSON uses JS syntax but the JSON format is text only

\*\*Text can be read & used as a data format by any programming language

* Exchanging Data:
* Data exchanged between a Browser & a Server can only be text
* Since JSON is text – can convert any JS Object into JSON and send JSON to the Server
* Can also convert any JSON received from the Server into JS Objects
* \*\*Allows for working w/ data as JS Objects w/out complicated parsing & translations
* Sending Data:
* var myObj = { "name":"John", "age":31, "city":"New York" };  
  var myJSON = JSON.stringify(myObj);  
  window.location = "demo\_json.php?x=" + myJSON;
* Receiving Data:
* var myJSON = '{ "name":"John", "age":31, "city":"New York" }';  
  var myObj = JSON.parse(myJSON);  
  document.getElementById("demo").innerHTML = myObj.name;

\***JSON.parse()** : built in JS function to convert a string, written in JSON format, into native JS Objects

* Storing Data:
* When storing, data has to be a certain format – text is always one of the legal formats
* JSON makes it possible to store JS Objects as text:

**//Storing data:**  
myObj = { "name":"John", "age":31, "city":"New York" };  
myJSON = JSON.stringify(myObj);  
localStorage.setItem("testJSON", myJSON); 🡪 Stores in local storage  
  
**//Retrieving data:**  
text = localStorage.getItem("testJSON");  
obj = JSON.parse(text);  
document.getElementById("demo").innerHTML = obj.name;

**JSON SYNTAX:**

* Rules: derived from JS
* Data is written in name/value pairs
* Data is separated by commas
* Curly Braces hold Objects & Square Brackets hold Arrays

\*\*JS Objects & Arrays can be used as JSON

* Name/Value pairs:
* “field name”: value 🡪 JSON names require double quotes
* Objects: { “name”: “John” } 🡪 JSON keys must be strings – in double quotes

JS: { name: “John” } 🡪 JS keys can be strings, numbers, or identifier names

* JSON Values:
* Must be one of the following data types
* string, number, Boolean, object (JSON object), array, null

\*\*strings must be – “John”

* JS: values can be all of the above + any other valid JS expression –

function, date, undefined

\*\*strings can be either – “John” or ‘John’

* JSON Files:
* File type: Files are saved w/ 🡪 “.json”
* MIME type: Internet Media type 🡪 “application/json”

**JSON VS XML:**

* Similarities:
* Both JSON and XML are "self-describing" (human readable)
* Both JSON and XML are hierarchical (values within values)
* Both JSON and XML can be parsed and used by lots of programming languages
* Both JSON and XML can be fetched with an XMLHttpRequest
* Differences:
* JSON doesn't use end tag
* JSON is shorter
* JSON is quicker to read and write
* JSON can use arrays

\*\*XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function

* WHY JSON IS BETTER THAN XML:

\*\*XML 🡪 more difficult to parse than JSON

\*\*JSON 🡪 parsed into a ready-to-use JavaScript object

\*\*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

**DATA TYPES:**

* Valid Data Types: 4 basic built in & 2 structured
* string 🡪 must be in double quotes
* number 🡪 must be an integer or floating point
* boolean
* null
* Object (JSON object) 🡪 list of name : value pairs
* Array 🡪 list of values
* Invalid Data Types:
* Function, Date, or Undefined

**OBJECTS:**

Ex. { “name” : “John”, “age” : 30, “car” : null };

* Keys 🡪 must be strings, Values 🡪 must be a valid data type
* Keys & Values separated by colon
* Each Key/Value pair separated by comma
* Accessing:
* Dot Notation or Bracket Notation
* Looping an Object:
* Loop through properties using the for-in-loop

for (x in myObj) {

document.getElementById(“demo”).innerHTML += x; OR 🡪 += myObj[x];

}

\*\*Use bracket notation to access property values in for-in loop

* Nested Objects:
* Values in an object can be another object

Ex. myObj = {“cars”:

{“car1” : “Ford”, “car2” : “BMW”, “car3” : “Fiat”} }

Access nested object using dot or bracket notation

* myObj.cars.car2 / myObj.cars[“car2”]
* Modify Values:
* Use dot or bracket notation
* Delete Object Properties:
* Use the ‘delete’ keyword

**ARRAYS AS JSON OBJECTS:**

**\*\*A**lmost same as JS arrays; Must be one of the 6 valid data types

* Arrays in JSON Objects:
* Can be values of an object property

{ “cars”: [ “Ford”, “BMW”, “Fiat”] }

* Accessing Array Values:
* Use index number 🡪 myObj.car[0];
* Loop through the Array:

**// for-in loop**

for (i in myObj.cars) {  
    x += myObj.cars[i];  
}

**// for loop**

for (i = 0; i < myObj.cars.length; i++) {  
    x += myObj.cars[i];  
}

* Nested Arrays in JSON Objects:
* Values in an Array can be another Array, or even another Object.
* To access arrays inside of arrays, use a for loop for each array:

for (i in myObj.cars) {  
    x += "<h1>" + myObj.cars[i].name + "</h1>";  
    for (j in myObj.cars[i].models) {  
        x += myObj.cars[i].models[j];  
    }  
}

**JSON PARSE:** JSON.parse()

* \*\*A common use of JSON is to exchange data to/from a web server
* \*\*When receiving data from a web server, the data is always a string
* \*\*Parse the data with JSON.parse(), and the data becomes a JavaScript object

**JSON STRINGIFY:** JSON.stringify()

* \*\*When sending data to a web server, the data has to be a string
* \*\*Convert a JavaScript object into a string with JSON.stringify()

JSON Tutorial @ <https://www.w3resource.com/JSON/introduction.php>

JSON:

* JSON is used as a data model
* Has low learning curve compared to other data models like Relational Database or XML

JSON vs RELATIONAL DATABASES (RDBMS)

* Structure:
* RDBMS: tables that store data in rows & columns
* JSON: objects & arrays
* Metadata:
* RDBMS: pre-defined schema for storing data about the structure & what type of data to be stored
* JSON: can also use a schema to define the structure & type of data to be represented- not predefined; mostly self-describing & more flexible than schemas in a RDBMS
* Retrieving Data:
* RDBMS: uses SQL (Structured Query Language)
* JSON: doesn’t have a widely used/accepted language to query the stored data; JAQL & JSONiq
* Sorting:
* RDBMS: SQL does this
* JSON: arrays often used in programs 🡪 arrays can be sorted
* Application:
* RDBMS: many open-source & commercial RDB systems available 🡪 MySQL, SQL Server, Oracle, DB2 etc.
* JSON: is mostly applied to programming languages; but noSQL systems use JSON format to store data 🡪 MongoDB, CoucheDB etc.
* Learning Curve:
* RDBMS: separate field of study 🡪 harder
* JSON: basics similar to other programming languages 🡪 easier

USES OF JSON:

* API:
* Most widely used area where JSON is used for data exchange; especially social web apps – Twitter, Facebook, Flicker etc.
* noSQL:
* Use JSON format to store data; ?? Evolved to rid Relational Database bottlenecks??
* JSON structures can be transformed into JS objects within the browser environment & integrated w/ server-side JS
* AJAX:
* Uses JSON to retrieve data from the server
* Retrieved data is presented in browser environment & using JS it can be manipulated and rendered
* Package Management:
* Developers nowadays use tools to create a package of their application 🡪 easier app development & maintenance
* Most tools available (Bower, Yomen, NPM) use a package.json file where the metadata is written

**WORKING WITH JSON & JS:**

* Serialization: Gets JSON value from JS value; JS 🡪 JSON
* Deserialization: JSON 🡪 JS
* The **JS JSON Object:**
* Comprises methods that convert JS values to JSON format & JSON notation to JS values
* **JSON.stringify**: converts JS values to JSON
* **JSON.parse**: converts JSON notation to JS values

\*\* Not safe to parse JSON using eval()

\*ONLINE JSON VEIWER: <https://www.w3resource.com/JSON/online-JSON-viewer.php>

\*JSONLint: open source project that allows you to validate your JSON data

* Copy into JSONLint online – <http://jsonlint.com/> 🡪 shows errors

The JSON format is syntactically identical to the code for creating JavaScript objects🡪 so, a JS program can easily convert JSON data into native JavaScript objects.

**Converting a JSON text to a JS object:**

* First, create a JavaScript string containing JSON syntax
* Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object
* Finally, use the new JavaScript object in your page