**JSON – JavaScript Object Notation**

**What is it?**

* 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

**JSON Syntax Rules**

* Data is in name/value pairs – names require double quotes
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

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

**JSON Files:**

* File type: .json
* MIME 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

\*\* XML is more difficult to parse than JSON; JSON is 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 – in double quotes
* number – 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 🡪 strings, values 🡪 valid data type
* accessing: dot notation or bracket notation

Looping An Object: loop through properties using the for-in-loop

Ex. // for (x in myObj) { document.getElementById(“demo”).innerHTML += x OR myObj[x] ; }

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:** almost same as JS arrays

* can be values of an object property. // { “cars”: [ “Ford”, “BMW”, “Fiat”] }
* access array values: use index number. // myObj.car[0];
* OR by looping through an 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: 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 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: evolved to rid Relational Database bottlenecks; use JSON format to store data

-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

Online JSON Viewer: <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

JSON & JS:

* Serialization:
* Deserialization:
* JavaScript 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()