JSON FILE NOTES

Syntax Rules

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 arrays

<https://www.w3schools.com/js/js_json_syntax.asp>

JSON objects are surrounded by curly braces {}.

JSON objects are written in key/value pairs.

Keys must be strings

Values must be a valid JSON data type (string, number, object, array, boolean or null).

Keys and values are separated by a colon.

Each key/value pair is separated by a comma.

## Accessing Object Values

You can access the object values by using dot (.) notation:

myObj = { "name":"John", "age":30, "car":null };  
x = myObj.name;

You can also access the object values by using bracket ([]) notation:

myObj = { "name":"John", "age":30, "car":null };  
x = myObj["name"];

## Looping an Object

You can loop through object properties by using the for-in loop:

myObj = { "name":"John", "age":30, "car":null };  
for (x in myObj) {  
    document.getElementById("demo").innerHTML += x;  
}

In a for-in loop, use the bracket notation to access the property values:

myObj = { "name":"John", "age":30, "car":null };  
for (x in myObj) {  
    document.getElementById("demo").innerHTML += myObj[x];  
}

## Nested JSON Objects

Values in a JSON object can be another JSON object.

myObj = {  
    "name":"John",  
    "age":30,  
    "cars": {  
        "car1":"Ford",  
        "car2":"BMW",  
        "car3":"Fiat"  
    }  
 }

You can access nested JSON objects by using the dot notation or bracket notation:

x = myObj.cars.car2;  
//or:  
x = myObj.cars["car2"];

## Modify Values

You can use the dot notation to modify any value in a JSON object:

myObj.cars.car2 = "Mercedes";

You can also use the bracket notation to modify a value in a JSON object:

myObj.cars["car2"] = "Mercedes";

## Delete Object Properties

Use the delete keyword to delete properties from a JSON object:

delete myObj.cars.car2;

## Arrays as JSON Objects

[ "Ford", "BMW", "Fiat" ]

In JSON, array values must be of type string, number, object, array, boolean or null.

## Arrays in JSON Objects

Arrays can be values of an object property:

{  
"name":"John",  
"age":30,  
"cars":[ "Ford", "BMW", "Fiat" ]  
}

## Accessing Array Values

You access the array values by using the index number:

x = myObj.cars[0];

## Looping Through an Array

You can access array values by using a for-in loop:

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

Or you can use a for loop:

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

## Nested Arrays in JSON Objects

Values in an array can also be another array, or even another JSON object:

myObj = {  
    "name":"John",  
    "age":30,  
    "cars": [  
        { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },  
        { "name":"BMW", "models":[ "320", "X3", "X5" ] },  
        { "name":"Fiat", "models":[ "500", "Panda" ] }  
    ]  
 }

To access arrays inside arrays, use a for-in 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];  
    }  
}

## Modify Array Values

Use the index number to modify an array:

 myObj.cars[1] = "Mercedes";

## Delete Array Items

Use the delete keyword to delete items from an array:

delete myObj.cars[1];

# **JSON PHP**

A common use of JSON is to read data from a web server, and display the data in a web page.

This chapter will teach you how to exchange JSON data between the client and a PHP server.

## JSONP Intro

JSONP stands for JSON with Padding.

Requesting a file from another domain can cause problems, due to cross-domain policy.

Requesting an external script from another domain does not have this problem.

JSONP uses this advantage, and request files using the script tag instead of the XMLHttpRequest object.

https://stackoverflow.com/questions/47612822/how-to-create-pandas-dataframe-from-twitter-search-api