**JSON: JavaScript Object Notation**

**JSON** is a syntax for **storing** and **exchanging data** between **Client** and **Server**.

**JSON** is **lightweight data-interchange format,** **language independent** and it is read for human to read and write.

**JSON** is **text**, written with **JavaScript object notation**. We can **convert** **any JavaScript object** into **JSON** and **send it to server**.

We can also **convert** any **JSON** **received** from **Server** into **JavaScript objects**.

This way we can work with the data as JavaScript objects, with no complicated parsing and translating.

**JSON syntax** is a **subset** of **JavaScript syntax**.

* Data is in name/value pairs
* Data is separated by commas
* Curly braces hold objects
* Square brackets hold arrays

**JSON Data** is written as **name/value pairs** followed by a colon, followed by value in double codes.

“name”: “ John” -----🡪 Keys must be strings, numbers or identifiers names:

JSON – evaluates to JavaScript Objects.

{ “name” : ”John” }

**In JSON, values** **must** be **one of** the following **data types**

* A String
* A Number
* An Object (JSON object)
* An Array
* A Boolean
* Null

**JSON Strings:**  must be in double quotes

{ “name” : “ John” }

**JSON number :**

{ “age” : 32 }

**JSON Objects:**

{ “employee” : { “name” : “John” , “age “ : 32 , city : “ London” }

**JSON Arrays:**

{ “employee” : [ “John “ , ” Peter” ,” Ram”] }

**JSON Booleans: ( true/false)**

{ “ sale” : true }

**JSON Null:**

{ “middlename” : null }

.

1. **JSON.parse() :** A **common use of JSON** is **to exchange data to/from Web Server**. When **receiving data** from a **Web Server**, the **data is always a string**.

We **parse** the **data** with **JSON.parse()**, and the **data becomes** a **JavaScript Object**.

Eg1:

If we receive this text from a Web Server

‘{ “name” : ”John” , “ age” : 32 , “city” : “London”}’

We use JavaScript function JSON.parse() to convert text into JavaScript Object.

Var obj=JSON.parse(‘{ “name” : ”John” , “ age” : 32 , “city” : “London”}’)

Then obj string is ready.

Eg2:

<!DOCTYPE html>

<html>

<body>

<h2>Create Object from JSON String</h2>

<p id="demo"></p>

<script>

var txt = '{"name":"John", "age":30, "city":"New York"}'

var obj = JSON.parse(txt);

document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;

</script>

</body>

</html>

**JSON parsing date:**

var text = '{ "name":"John", "birth":"1986-12-14", "city":"New York"}';  
var obj = JSON.parse(text);  
obj.birth = new Date(obj.birth);  
  
document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;

1. **JSON.stringify():** A **common use** of **JSON** is to **exchange date to/from Web Server**. **when** **sending data** to a **Web Server**, the **data** **has** to be a **string.**

We **convert** a **JavaScript object** into a **string** with **JSON.stringify().**

Eg1:

We have the JavaScript Object

var obj = '{"name":"John", "age":30, "city":"New York"}'

we use JavaScript function JSON.stringyify() to convert object into string

var myJson = JSON.stringify(obj);

then myJson is string ready to send JSON to the Server

**We can also Stringify a JavaScript Array.**

Var arr=[“john” , “peter” , “-Julie”];

Var myJson=JSON.stringify(arr);

1. **JSON Objects:**

**JSON objects** are surrounded by **curly braces {}**

**JSON objects** are **written** in **key/value pairs.**

**Keys** must be **Strings** and **values** must be **valid JSON data type**.

**Keys** and **values** are **separated** by a **colon.**

**Each key/value pair** is **separated** by a **comma.**

**Accessing JSON Object values by dot (.) notation**

<p id="demo"></p>

<script>

var myObj, x;

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

x = myObj.name;

document.getElementById("demo").innerHTML = x;

</script>

Or with [ ] notation

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

**Accessing JSON Object values using loop**

<p id="demo"></p>

<script>

var myObj, x;

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

for (x in myObj) {

document.getElementById("demo").innerHTML += x + "<br>";

}

</script>

**Nested JSON Objects:**

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

we can access the nested JSON Objects by using dot notation or bracket notation

x=myobj.cars.car2;

or

x=myobj.cars[“car2”];

**Modify Values:**

myobj.cars.car2=”Mercedes”;

**Delete Object properties:**

Delete myobj.cars.car2;

1. **JSON Arrays:**

Arrays in JSON are almost the same as arrays in JavaScript.

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

**Accessing Array values**

X=myobj.cars[0];

**Looping through an Arrays**

var i,x;

for ( i in myobj.cars)

{

x +=myonj.cars[i] + “<br>”;

}

Document.getElementById(“demo”).innerHTML=X;

Nested Arrays in JSON Objects

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

accessing values from nested arrays

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];  
  }  
}