JSON-JavaScript Object Notation.

JSON

- JSON stands for JavaScript Object Notation.
- It is an open standard data-interchange format.
- It is lightweight and self describing.
- It is originated from JavaScript.
- JSON is text, written with JavaScript object notation.
- It is easy to read and write than XML. For AJAX applications, JSON is faster and easier than XML.
- It is language independent (interoperability).
- It supports array, object, string, number and values.
- The format was specified by Douglas Crockford.
- The JSON file must be save with .json extension.
- The MIME type for JSON text is "application/json"

JSON and XML

- JSON and XML are human readable formats and are language independent. They both have support for creation, reading and decoding in real world situations.
- **JSON**

```
{"car":{"company": “Volkswagen”, "name": "Vento", "price": 800000 }}
```

- **XML**

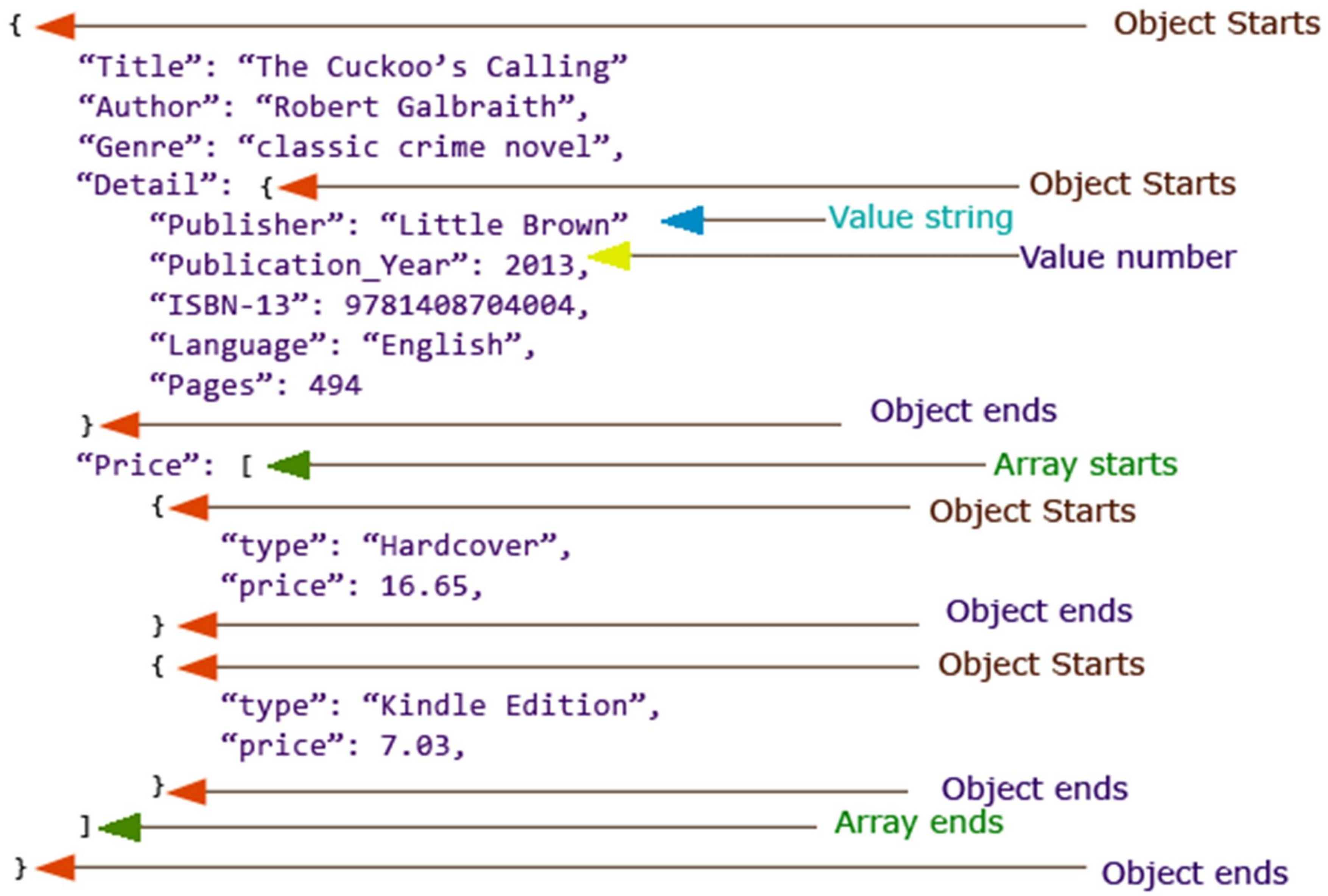
```
<car>
<company>Volkswagen</company>
 <name>Vento</name>
<price>800000</price>
 </car>
```

A syntax for storing and exchanging data

- When exchanging data between a browser and a server, the data can only be text.
- JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server.
- We can also convert any JSON received from the server into JavaScript objects.
- This way we can work with the data as JavaScript objects, with no complicated parsing and translations.
- If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server using JSON.stringify().
- If you receive data in JSON format, you can convert it into a JavaScript object using JSON.parse().

Uses of JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- JSON uses JavaScript syntax, but the JSON format is text only. Text can be read and used as a data format by any programming language.



```
{
   "Title": "The Cuckoo's Calling"
   "Author": "Robert Galbraith",
   "Genre": "classic crime novel",
   "Detail": {
       "Publisher": "Little Brown"
       "Publication_Year": 2013,
       "ISBN-13": 9781408704004,
       "Language": "English",
       "Pages": 494
   }
   "Price": [
       {
           "type": "Hardcover",
           "price": 16.65,
       }
       {
           "type": "Kindle Edition",
           "price": 7.03,
       }
   ]
}
```

Example 1 – to store employee data - first.json

```
{"employees":[
  {     "name":"Sonoo",
        "email":"sonoojaiswal1987@gmail.com"},
  {     "name":"Rahul",
        "email":"rahul32@gmail.com"},
  {     "name":"John",
        "email":"john32bob@gmail.com"}
]}
```

Example 2 – to store book data – book.json

```
{ "book": [
        {       "id":"01",
                "language": "Java",
                "edition": "third",
                "author": "Herbert Schildt"    },
        {       "id":"07",
                "language": "C++",
                "edition": "second",
                "author": "E.Balagurusamy"    }
] }
```

JSON syntax

- Data is represented in name/value pairs. JSON names require double quotes. JavaScript names don't.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).

```
{
   "employee": {"name": "sonoo", "salary": 56000,  "married":  true}
}
```

- Square brackets hold arrays and values are separated by ,(comma).

1. ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"] → Values in an Array
2. [{"name":"Ram", "email":"Ram@gmail.com"}, → Objects in an Array
   {"name":"Bob", "email":"bob32@gmail.com"}
   ]

JSON Datatypes

- Number
  - Integer (0-9 ,+ve,-ve), Fraction, Exponent.
  - Octal and hexadecimal formats are not used.
  - No NaN or Infinity is used in Number.

```
{
"integer": 34,
"fraction": .2145,
"exponent": 6.61789e+0
}
```

- String → It is a sequence of zero or more double quoted Unicode characters with backslash escaping.

Eg: “aaa ”,’aaa’, \n, \t

JSON Datatypes

- Boolean → true, false
- Array → an ordered collection of values
- Value – String, number, true or false, null etc
- Object – unordered collection of key:value pairs
- Whitespace – can be used between any pair of tokens. It can be added to make a code more readable.

```
var obj1 = {"name":    "Sachin Tendulkar"}
var obj2 = {"name":    "Saurav Ganguly"}
```

- null → empty type

```
var i = null;
```

- JSON values **cannot** be one of the following data types: a function, a date, *undefined*

Accessing Object Values

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

- x = myObj.name; → **Output:** John
- x = myObj["name"]; → **Output:** John
- for (x in myObj) {
      document.getElementById("demo").innerHTML += x;
  }

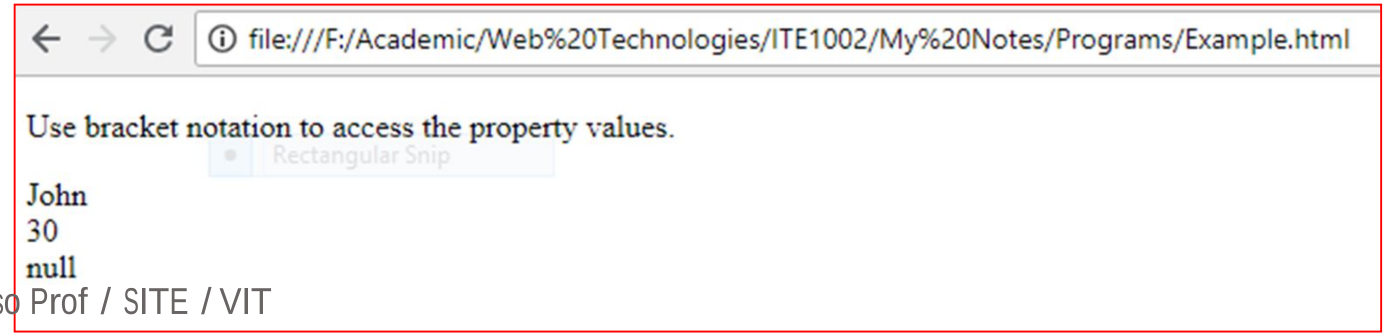
Output:
```
name
 age
car
```

- for (x in myObj) {
      document.getElementById("demo").innerHTML += myObj[x];
  }

Output:
```
John
30
null
```

Example.html

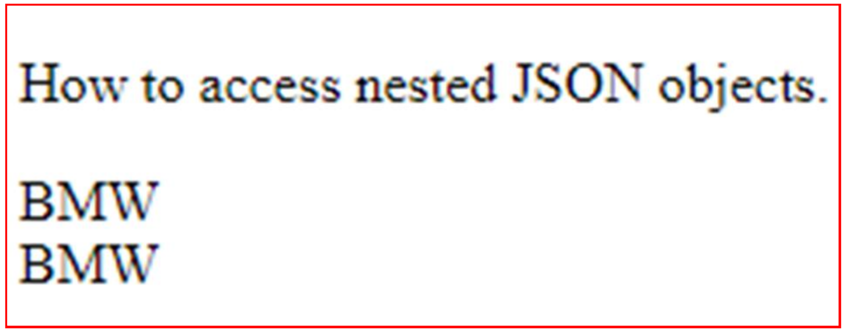
```
<html><body>
<p>Use bracket notation to access the property values.</p>
<p id="demo"></p>
<script>
var myObj = {"name":"John", "age":30, "car":null};
for (x in myObj) {
   document.getElementById("demo").innerHTML += myObj[x] +
"<br>";}
</script></body></html>
```


file:///F:/Academic/Web%20Technologies/ITE1002/My%20Notes/Programs/Example.html

```
Use bracket notation to access the property values.

John
30
null
```

Nested Objects

```
<html> <body> <p>How to access nested JSON objects.</p>
<p id="demo"></p><script>
var myObj = {
        "name":"John",
        "age":30,
        "cars": {
                "car1":"Ford",
                "car2":"BMW",
                "car3":"Fiat"   } }
document.getElementById("demo").innerHTML += myObj.cars.car2 +
"<br>"; //or:
document.getElementById("demo").innerHTML += myObj.cars["car2"];
</script> </body></html>
```



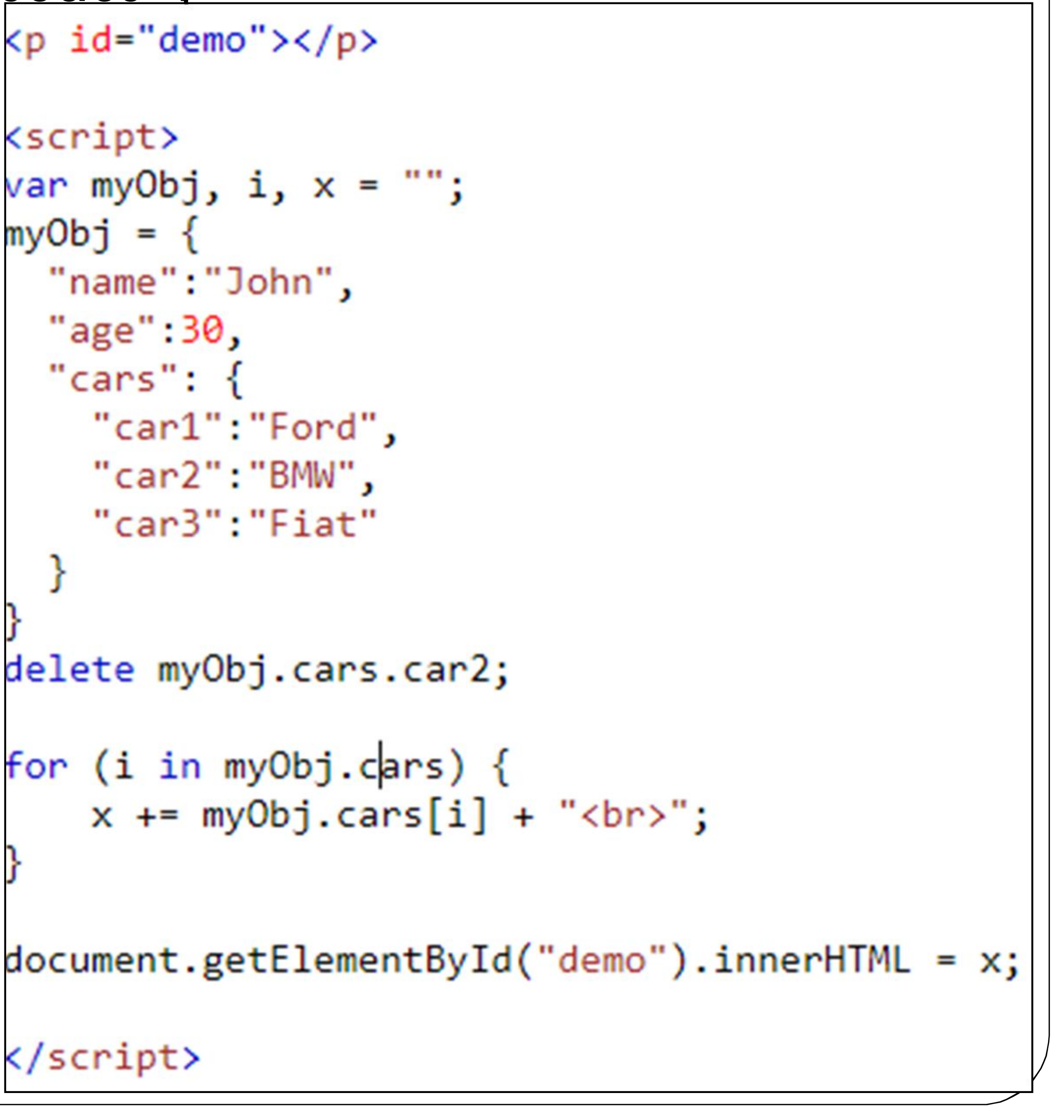
```
How to access nested JSON objects.

BMW
BMW
```

Modify Values

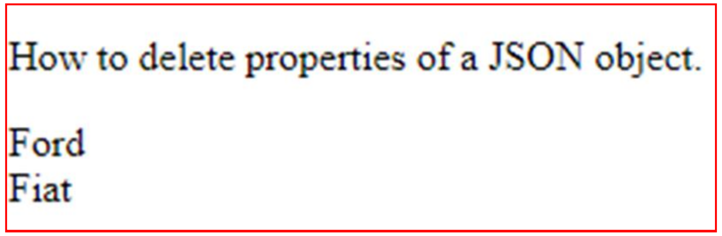
- myObj.cars.car2 = "Mercedes"; / / or
- myObj.cars["car2"] = "Mercedes";

Delete Object Properties

delete myObj.cars.car2;



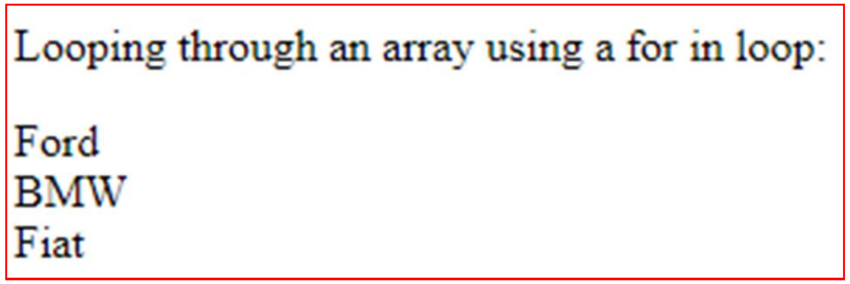
```
<p id="demo"></p>

<script>
var myObj, i, x = "";
myObj = {
  "name":"John",
  "age":30,
  "cars": {
    "car1":"Ford",
    "car2":"BMW",
    "car3":"Fiat"
  }
}
delete myObj.cars.car2;

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

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

</script>
```



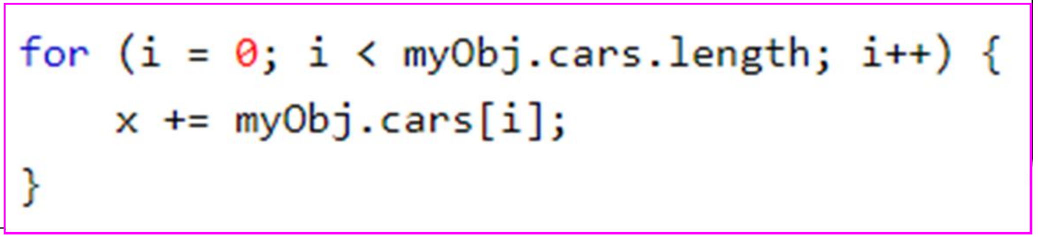
```
How to delete properties of a JSON object.

Ford
Fiat
```

Arrays in JSON Objects

```
<body> <p>Looping through an array using a for in loop:</p>
<p id="demo"></p> <script>
var myObj, i, x = "";
myObj = {
        "name":"John",
        "age":30,
        "cars":[ "Ford", "BMW", "Fiat" ]    };
for (i in myObj.cars) {
      x += myObj.cars[i] + "<br>";}
document.getElementById("demo").innerHTML = x;
</script></body></html>
```



```
Looping through an array using a for in loop:

Ford
BMW
Fiat
```



```
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" ] }
    ]
 }
```

- **Accessing:**

```
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 using index:**

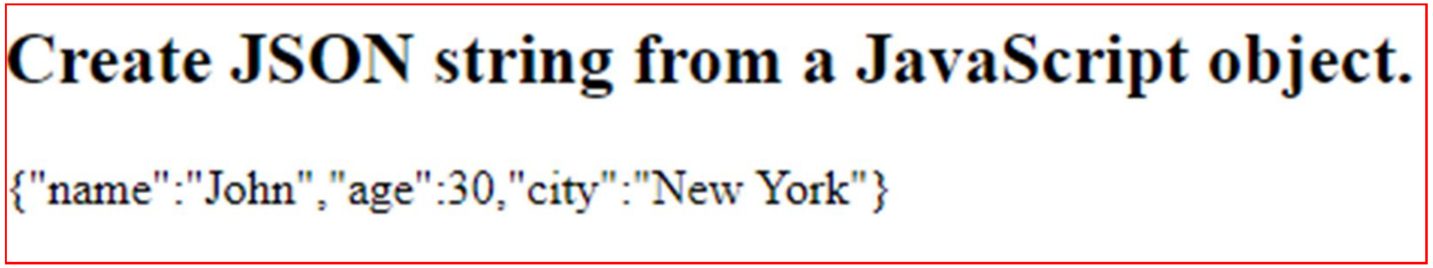
```
myObj.cars[1].name= "Mercedes";
```

- **Delete using index:**

```
delete myObj.cars[1];
```

A common use of JSON is to exchange data to/from a web server. When sending data to a web server, the data has to be a string. Convert a JavaScript object into a string with JSON.stringify()

```
<html> <body>
<h2>Create JSON string from a JavaScript object.</h2>
<p id="demo"></p><script>
var obj = {"name":"John", "age":30, "city":"NewYork"};
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script> </body></html>
```

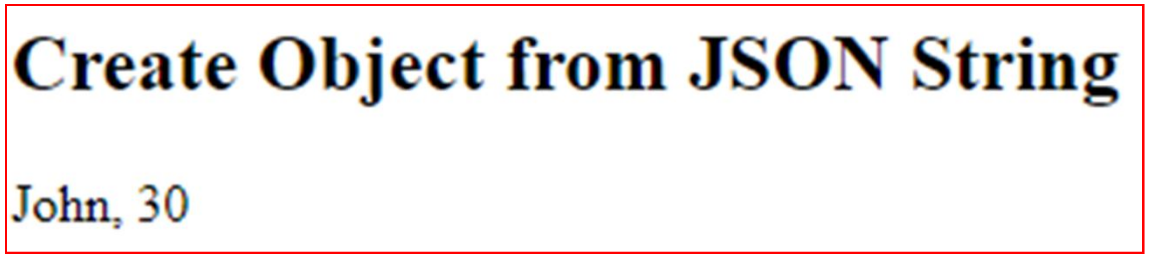


Create JSON string from a JavaScript object.

```
{"name":"John","age":30,"city":"New York"}
```

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.

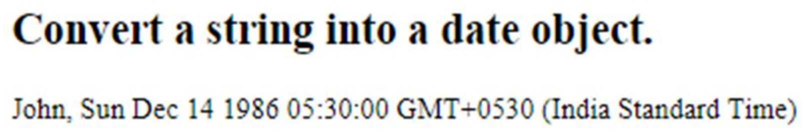
```
<html> <body><h2>Create Object from JSON String</h2>
<p id="demo"></p><script>
var txt = '{"name":"John", "age":30, "city":"NewYork"}'
var obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script></body></html>
```



Create Object from JSON String

John, 30

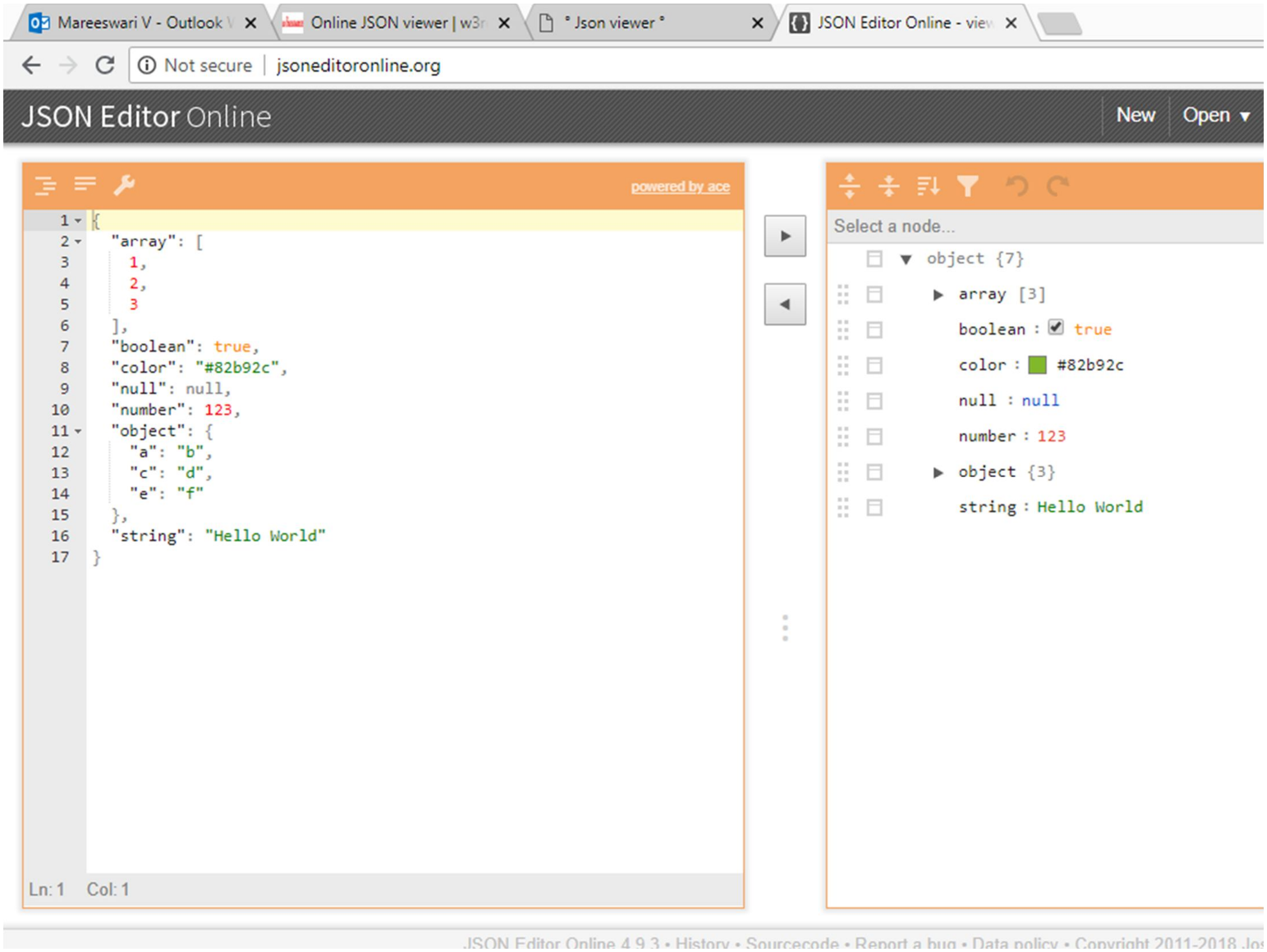
```
<!DOCTYPE html>
<html>
<body>
<h2>Convert a string into a date object.</h2>
<p id="demo"></p>
<script>
var text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';
var obj = JSON.parse(text, function (key, value) {
    if (key == "birth") {
        return new Date(value);
    } else {
        return value;
    }});
document.getElementById("demo").innerHTML = obj.name + ", " + obj.birth;
</script>
</body>
</html>
```



```
Convert a string into a date object.

John, Sun Dec 14 1986 05:30:00 GMT+0530 (India Standard Time)
```

JSON Tools

- JSON Editors
- JSON Parser
- JSON Viewer
- JSON Tree Viewer Tool

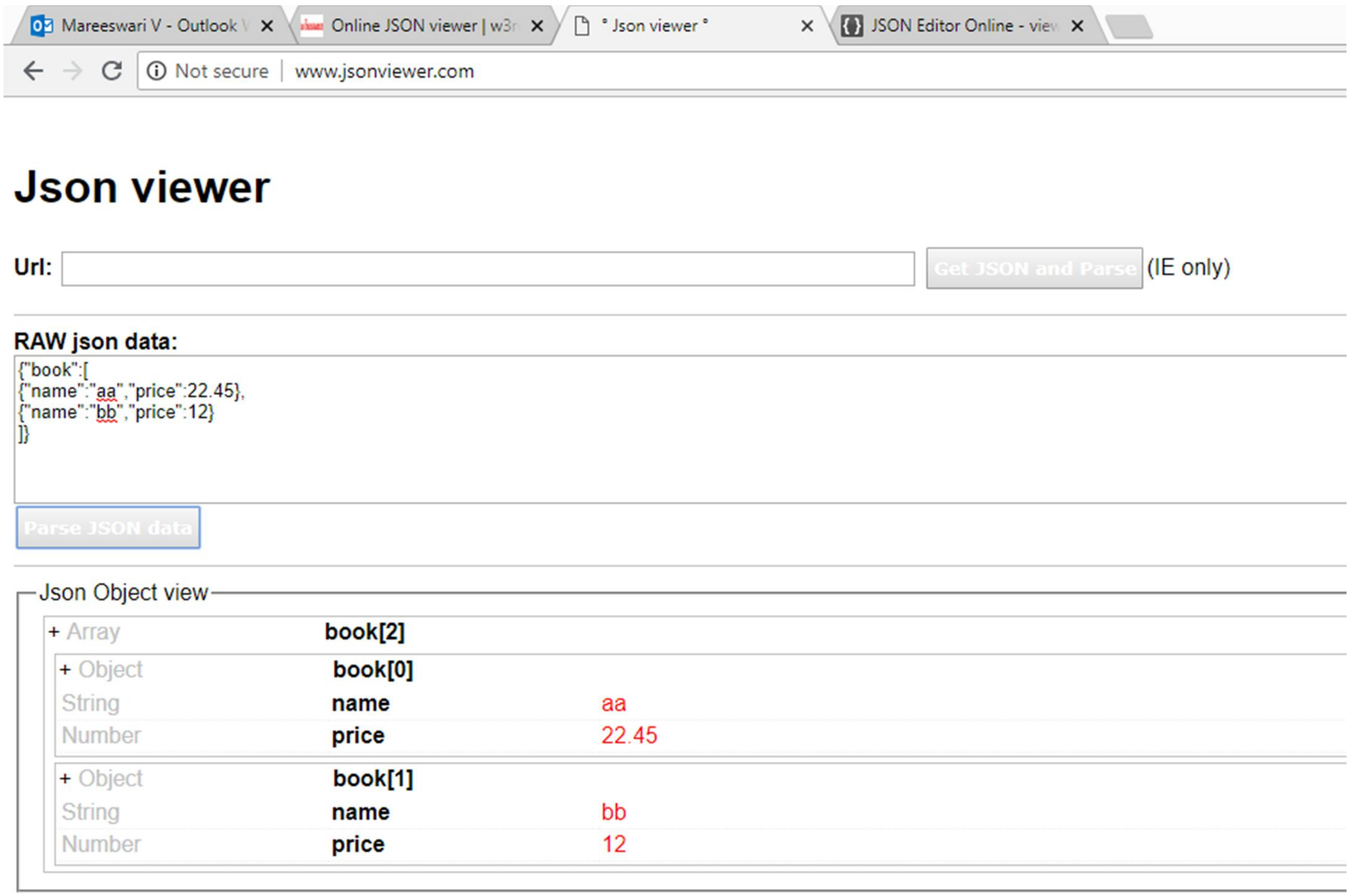


```
{
  "array": [
    1,
    2,
    3
  ],
  "boolean": true,
  "color": "#82b92c",
  "null": null,
  "number": 123,
  "object": {
    "a": "b",
    "c": "d",
    "e": "f"
  },
  "string": "Hello World"
}
```

Ln: 1   Col: 1

```
object {7}
  ▶ array [3]
    boolean : ✓ true
    color : 🟩 #82b92c
    null : null
    number : 123
  ▶ object {3}
    string : Hello World
```

JSON Editor Online 4.9.3 • History • Sourcecode • Report a bug • Data policy • Copyright 2011-2018 Jos



Json viewer

Url: [Get JSON and Parse] (IE only)

RAW json data:

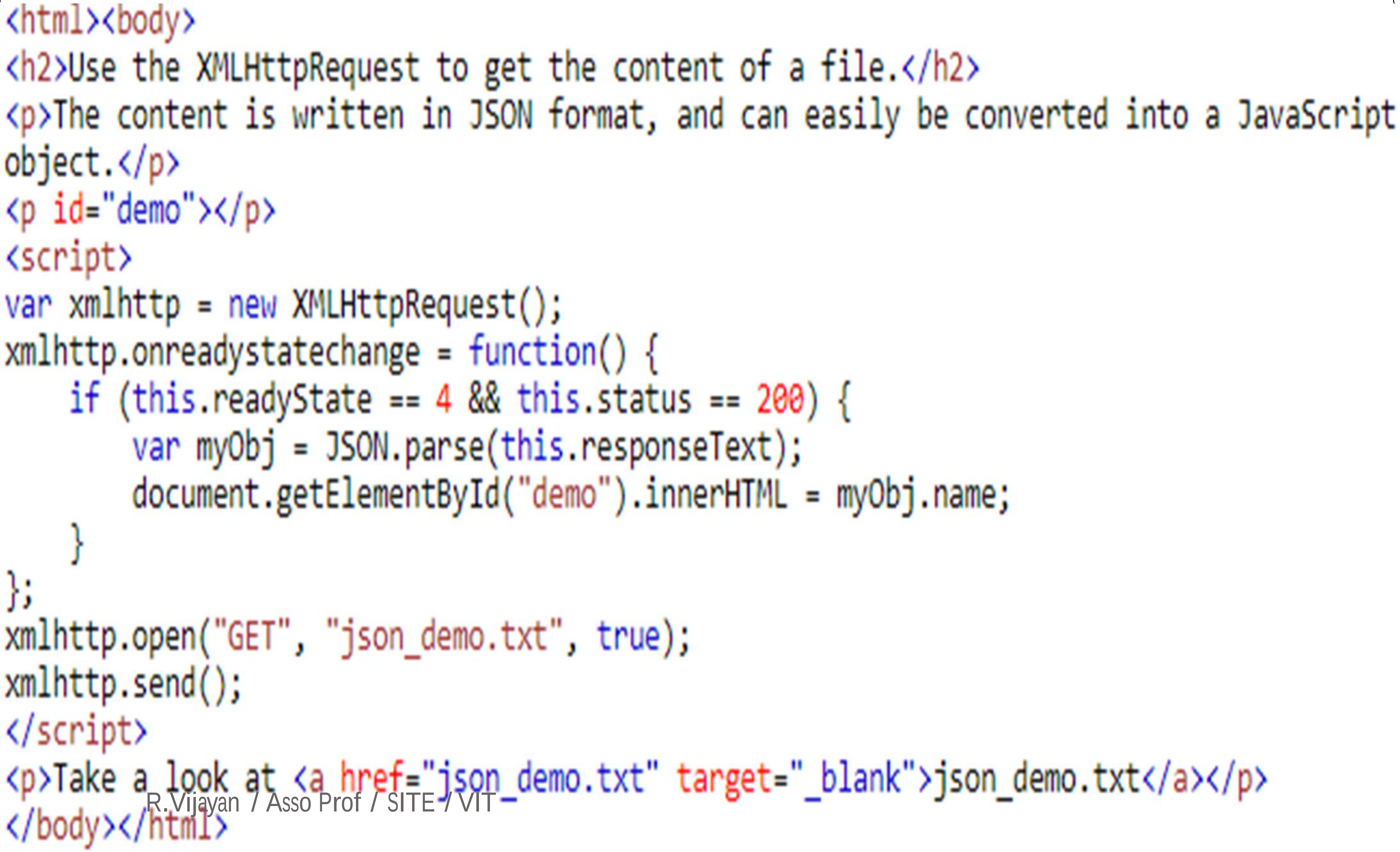
```
{"book":[
{"name":"aa","price":22.45},
{"name":"bb","price":12}
]}
```

[Parse JSON data]

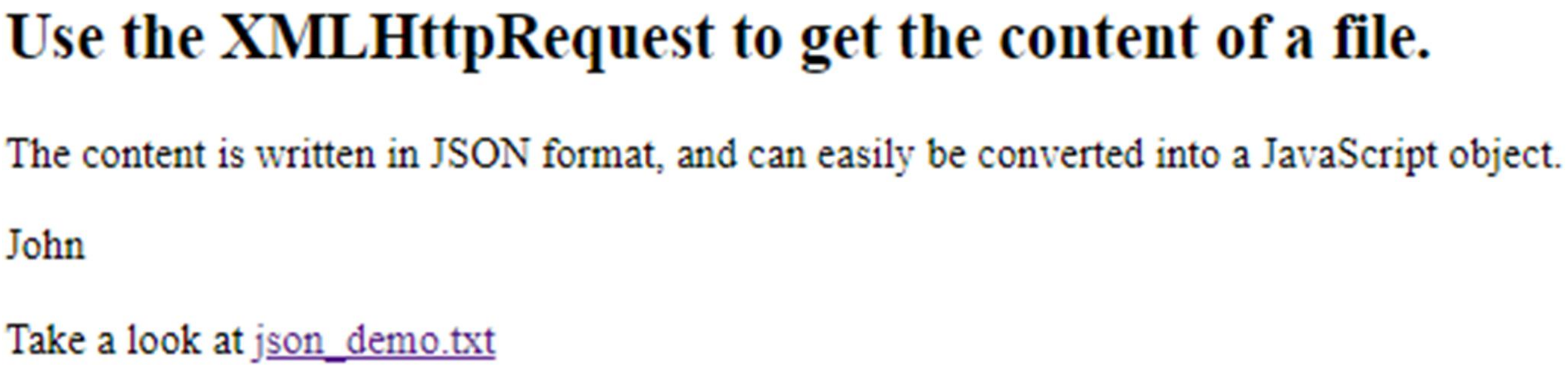
Json Object view

```
+ Array       book[2]
+ Object      book[0]
String        name        aa
Number        price       22.45
+ Object      book[1]
String        name        bb
Number        price       12
```

JSON From the Server : You can request JSON from the server by using an AJAX request. As long as the response from the server is written in JSON format, you can parse the string into a JavaScript object.



```
<html><body>
<h2>Use the XMLHttpRequest to get the content of a file.</h2>
<p>The content is written in JSON format, and can easily be converted into a JavaScript
object.</p>
<p id="demo"></p>
<script>
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        var myObj = JSON.parse(this.responseText);
        document.getElementById("demo").innerHTML = myObj.name;
    }
};
xmlhttp.open("GET", "json_demo.txt", true);
xmlhttp.send();
</script>
<p>Take a look at <a href="json_demo.txt" target="_blank">json_demo.txt</a></p>
</body></html>
```



Use the XMLHttpRequest to get the content of a file.

The content is written in JSON format, and can easily be converted into a JavaScript object.

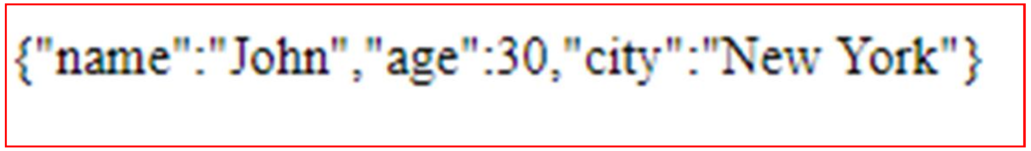
John

Take a look at json_demo.txt

The PHP File

- PHP has some built-in functions to handle JSON.
- Objects in PHP can be converted into JSON by using the PHP function json_encode():

```
<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";
$myJSON = json_encode($myObj);
echo $myJSON;
?>
```



```
{"name":"John","age":30,"city":"New York"}
```

S.NO	JSON	Javascript
1	Strings in JSON must be written in double quotes. Eg: “name”:“John”	JavaScript names don't.
2	*values* must be one of the following data types: a string,a number an object (JSON object),an array,a boolean and null	values can be all of the JSON value types, plus any other valid JavaScript expression, including: a function, a date, undefined

Exercise

- Develop a json program for creating 20 students personal information using the following fields as a inputs and print the outputs in table format,student name, date_of_birth(date,month,year),parents(fathers_name,mothers_name), blood_group,email,phone(landline,mobile),address(door_no ,street_name,place_name,pincode),degree(ug,pg and others),employee(self or organization).

JSON Schema

- JSON Schema is a specification for JSON based format for defining the structure of JSON data. It was written under IETF draft which expired in 2011. JSON Schema –
- Describes your existing data format.
- Clear, human- and machine-readable documentation.
- Complete structural validation, useful for automated testing.
- Complete structural validation, validating client-submitted data.