JSON
(**J**ava**S**cript **O**bject **N**otation)

infobizzs.com

What is JSON?

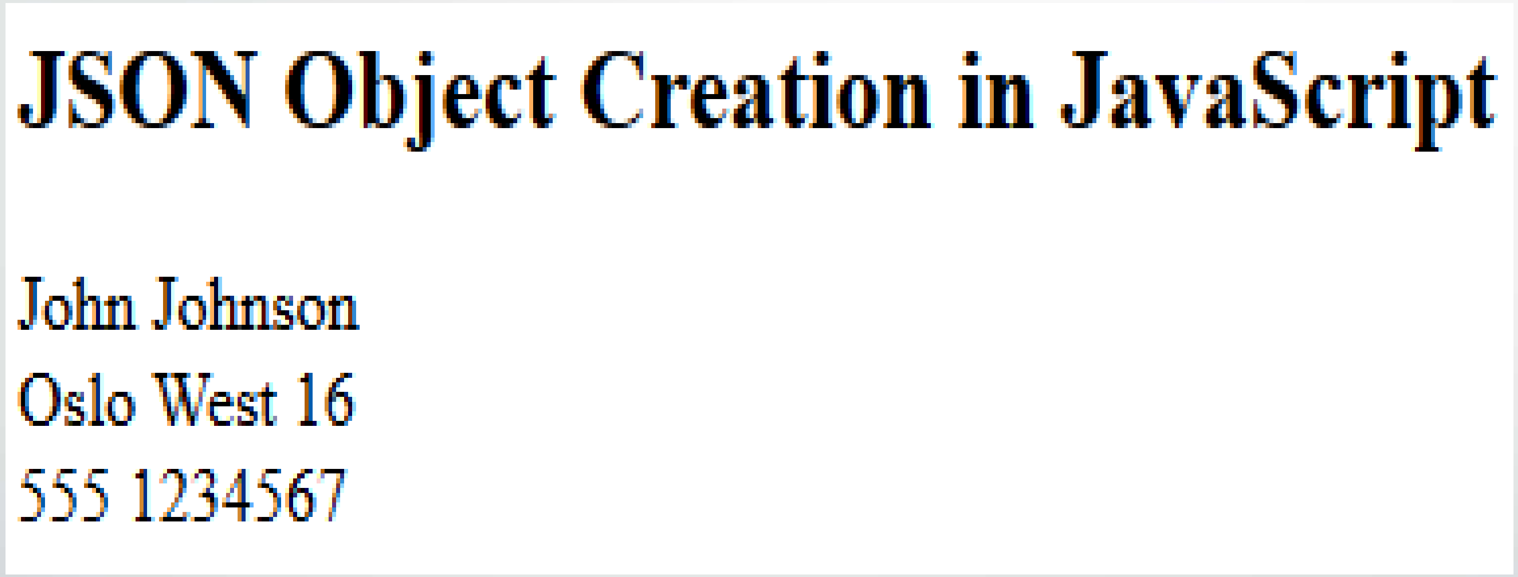
- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand
- JSON is a syntax for storing and exchanging data.
- JSON is an easier-to-use alternative to XML.

JSON Example

```
<!DOCTYPE html>
<html>
<body>

	<h2>JSON Object Creation in JavaScript</h2>

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

	<script>
		var text = '{"name":"John Johnson","street":"Oslo West 16","phone":"555 1234567"}';

		var obj = JSON.parse(text);

		document.getElementById("demo").innerHTML =
		obj.name + "<br>" +
		obj.street + "<br>" +
		obj.phone;
	</script>

</body>
</html>
```

infobizzs.com

Output



```
JSON Object Creation in JavaScript

John Johnson
Oslo West 16
555 1234567
```

infobizzs.com

Much Like XML Because

- Both JSON and XML is "self describing" (human readable)
- Both JSON and XML is hierarchichal (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

Much Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays
- The biggest difference is:

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

Why JSON?

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

JSON Syntax

The JSON syntax is a subset of the JavaScript syntax.

- **JSON 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

JSON Data - A Name and a Value

- JSON data is written as name/value pairs.
- A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

```
"firstName":"John"
```

- **JSON Values**
  - JSON values can be:
  - A number (integer or floating point)
  - A string (in double quotes)
  - A Boolean (true or false)
  - An array (in square brackets)
  - An object (in curly braces)
  - null

JSON Objects

- JSON objects are written inside curly braces.
- Just like JavaScript, JSON objects can contain multiple name/values pairs:

```
{"firstName":"John", "lastName":"Doe"}
```

JSON Arrays

- JSON arrays are written inside square brackets.
- Just like JavaScript, a JSON array can contain multiple objects:

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

infobizzs.com

JSON Uses JavaScript Syntax

- Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.
- With JavaScript you can create an array of objects and assign data to it, like this:

```
var employees = [
        {"firstName":"John", "lastName":"Doe"},
        {"firstName":"Anna", "lastName":"Smith"},
        {"firstName":"Peter","lastName": "Jones"}
    ];
```

JSON How To

- A common use of JSON is to read data from a web server, and display the data in a web page.
- For simplicity, this can be demonstrated by using a string as input (instead of a file).
- ***JSON Example - Object From String***
    - Create a JavaScript string containing JSON syntax:

```
var text = '{ "employees" : [' +
           '{ "firstName":"John" , "lastName":"Doe" },' +
           '{ "firstName":"Anna" , "lastName":"Smith" },' +
           '{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

infobizzs.com

- JSON syntax is a subset of JavaScript syntax.
- The JavaScript function JSON.parse(*text*) can be used to convert a JSON text into a JavaScript object:

```
var obj = JSON.parse(text);
```

Use the new JavaScript object in your page:

- **Example**

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

<script>
    document.getElementById("demo").innerHTML =
    obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

JSON Http Request

- 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, in 4 easy steps, how to read JSON data, using XMLHttp.
- This example reads a menu from **myTutorials.txt**, and displays the menu in a web page:

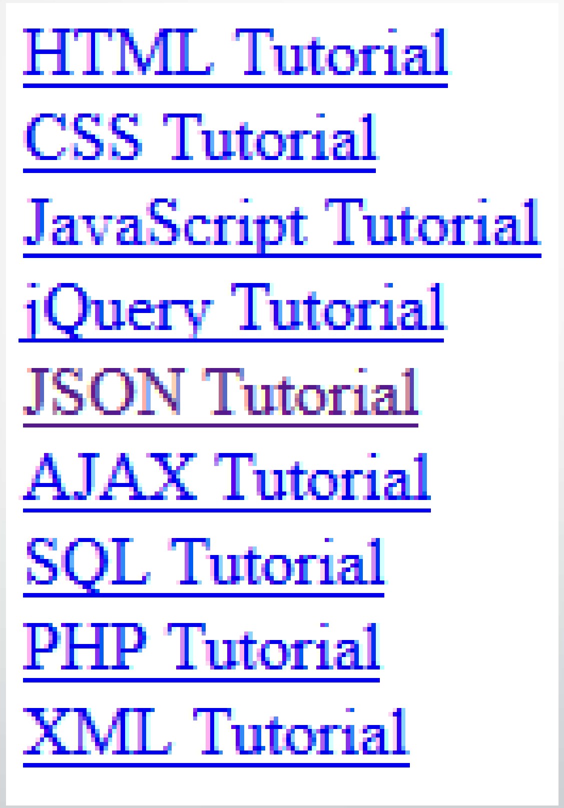
```
<div id="id01"></div>

    <script>
        var xmlhttp = new XMLHttpRequest();
        var url = "myTutorials.txt";

        xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        var myArr = JSON.parse(xmlhttp.responseText);
        myFunction(myArr);
        }
    }
```

infobizzs.com

```
xmlhttp.open("GET", url, true);
xmlhttp.send();

function myFunction(arr) {
        var out = "";
        var i;
        for(i = 0; i < arr.length; i++) {
        out += '<a href="' + arr[i].url + '">' +
        arr[i].display + '</a><br>';
        }
        document.getElementById("id01").innerHTML = out;
    }
</script>
```

infobizzs.com

Output


```
HTML Tutorial
CSS Tutorial
JavaScript Tutorial
jQuery Tutorial
JSON Tutorial
AJAX Tutorial
SQL Tutorial
PHP Tutorial
XML Tutorial
```

infobizzs.com

Example Explained

1: Create an array of objects.

- Use an **array literal** to declare an **array** of **objects**.
- Give each object two properties: **display** and **url**.
- Name the array **myArray**:

myArray

```
var myArray = [
        {
            "display": "JavaScript Tutorial",
            "url": "http://www.w3schools.com/js/default.asp"
        },
        {
            "display": "HTML Tutorial",
            "url": "http://www.w3schools.com/html/default.asp"
        },
        {
            "display": "CSS Tutorial",
            "url": "http://www.w3schools.com/css/default.asp"
        }
        ]
```

2: Create a JavaScript function to display the array.

- Create a function **myFunction()** that loops the array objects, and display the content as HTML links:

myFunction()

```
function myFunction(arr) {
        var out = "";
        var i;
        for(i = 0; i < arr.length; i++) {
        out += '<a href="' + arr[i].url + '">' + arr[i].display + '</a><br>';
        }
        document.getElementById("id01").innerHTML = out;
}
```

infobizzs.com

3: Create a text file

- Put the **array literal** in a file named **myTutorials.txt**:

myTutorials.txt

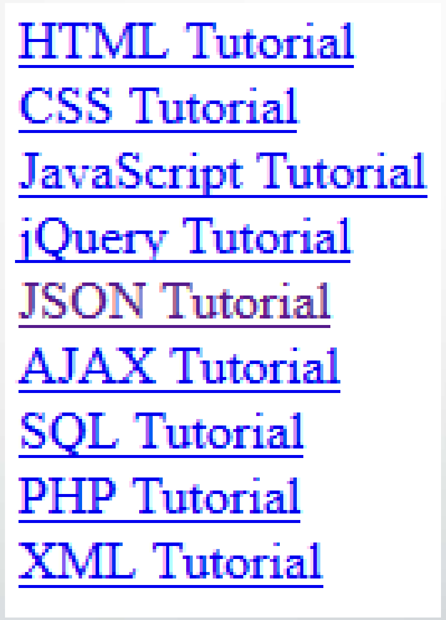
```
[
    {
        "display": "JavaScript Tutorial",
        "url": "http://www.w3schools.com/js/default.asp"
    },
    {
        "display": "HTML Tutorial",
        "url": "http://www.w3schools.com/html/default.asp"
    },
    {
        "display": "CSS Tutorial",
        "url": "http://www.w3schools.com/css/default.asp"
    }
]
```

4: Read the text file with an XMLHttpRequest

- Write an **XMLHttpRequest** to read the text file, and use **myFunction()** to display the array:

```
var xmlhttp = new XMLHttpRequest();
var url = "myTutorials.txt";

    xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
    var myArr = JSON.parse(xmlhttp.responseText);
    myFunction(myArr);
     }
}

xmlhttp.open("GET", url, true);
xmlhttp.send();
```

Output



```
HTML Tutorial
CSS Tutorial
JavaScript Tutorial
jQuery Tutorial
JSON Tutorial
AJAX Tutorial
SQL Tutorial
PHP Tutorial
XML Tutorial
```

infobizzs.com