

2.6 JSON INTRODUCTION

- JSON stands for **JavaScript Object Notation**
- JSON is a **text format** for storing and transporting data
- It is a lightweight human readable collection of data that can be accessed in a logical manner.

JSON Example

1. While working with .json file, the syntax will be like:

```
{  
  
  "First_Name" : "value";  
  
    "Last_Name": "value "  
  
  }
```

2. While working with JSON object in .js or .html file, the syntax will be like:

```
var varName = {  
  
  "First_Name" : "value";  
  
  "Last_Name": "value "  
  
  }
```

If you parse the JSON string with a JavaScript program, you can access the data as an object:

```
let personName = obj.name;  
let personAge = obj.age;
```

The JSON syntax is derived from JavaScript object notation, but the JSON format is text only.

Code for reading and generating JSON exists in many programming languages.

Why Use JSON?

The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

Since the format is text only, JSON data can easily be sent between computers, and used by any programming language. JavaScript has a built in function for converting JSON strings into JavaScript objects:

```
JSON.parse()
```

JavaScript also has a built in function for converting an object into a JSON string:

```
JSON.stringify()
```

You can receive pure text from a server and use it as a JavaScript object.

You can send a JavaScript object to a server in pure text format.

You can work with data as JavaScript objects, with no complicated parsing and translations.

Storing Data

When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats. JSON makes it possible to store JavaScript objects as text.

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 (aka key/value pairs).

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

Example

```
"name":"John"
```

JSON names require double quotes.

JSON - Evaluates to JavaScript Objects

The JSON format is almost identical to JavaScript objects.

In JSON, *keys* must be strings, written with double quotes:

JSON

```
{"name":"John"}
```

In JavaScript, keys can be strings, numbers, or identifier names:

JavaScript

```
{name:"John"}
```

JSON Values

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

- a string
- a number
- an object
- an array
- a boolean
- null

In **JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:

- a function
- a date
- undefined

In JSON, *string values* must be written with double quotes:

JSON

```
{"name":"John"}
```

In JavaScript, you can write string values with double *or* single quotes:

JavaScript

```
{name:'John'}
```

JavaScript Objects

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 object and assign data to it, like this:

Example

```
person = {name:"John", age:31, city:"New York"};
```

JavaScript JSON Methods

Let's see the list of JavaScript **JSON** method with their description.

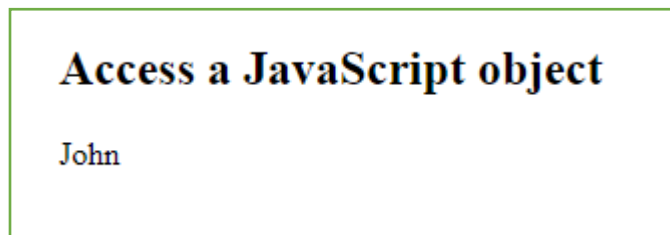
Methods	Description
<u>JSON.parse()</u>	This method takes a JSON string and transforms it into a JavaScript object.
<u>JSON.stringify()</u>	This method converts a JavaScript value (JSON object) to a JSON string representation.

You can access a JavaScript object like this:

Example

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Access a JavaScript object</h2>
    <p id="demo"></p>
    <script>
      const myObj = { name:"John", age:30, city:"New York" };
      document.getElementById("demo").innerHTML = myObj.name;
    </script>
  </body>
</html>
```

Output



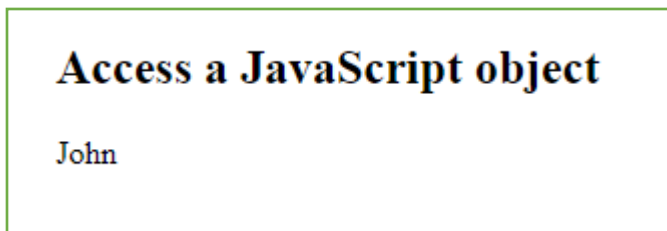
The screenshot shows a browser window with a green border. Inside, the text "Access a JavaScript object" is displayed in a large, bold, black serif font. Below it, the name "John" is displayed in a smaller, black serif font.

(i) It can also be accessed like this:

Example

```
// returns John
person["name"];
```

Output:



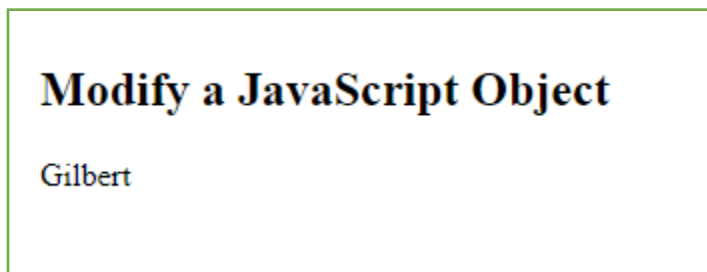
The screenshot shows a browser window with a green border. Inside, the text "Access a JavaScript object" is displayed in a large, bold, black serif font. Below it, the name "John" is displayed in a smaller, black serif font.

(ii) Data can be modified like this:

Example

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Modify a JavaScript Object</h2>
    <p id="demo"></p>
    <script>
      const myObj = { name:"John", age:30, city:"New York" };
      myObj.name = "Gilbert";
      document.getElementById("demo").innerHTML = myObj.name;
    </script>
  </body>
</html>
```

Output:



(iii) It can also be modified like this:

Example

```
person["name"] = "Gilbert";
```

Modify a JavaScript Object

Gilbert

JSON Files

- The file type for JSON files is ".json"
- The MIME type for JSON text is "application/json"

JSON FUNCTION FILES

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 function files.

JSON Example

This example reads a menu from [myTutorials.js](#), and displays the menu in a web page:

JSON Example

```
<div id="id01"></div>
<script>
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>
<script src="myTutorials.js"></script>
```

Output:

[HTML Tutorial](#)
[CSS Tutorial](#)
[JavaScript Tutorial](#)
[SQL Tutorial](#)
[PHP Tutorial](#)
[XML Tutorial](#)

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": "https://www.w3schools.com/js/default.asp"  
  },  
  {  
    "display": "HTML Tutorial",  
    "url": "https://www.w3schools.com/html/default.asp"  
  },  
  {  
    "display": "CSS Tutorial",  
    "url": "https://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()

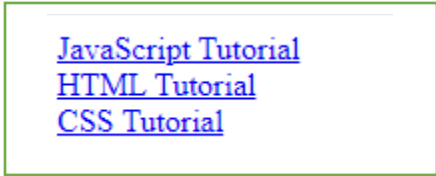
```
function myFunction(arr) {
  var out = "";
  var i;
  for(i = 0; i < arr.length; i++) {
    out += '<a href="' + arr[i].url + "'>' + arr[i].display + '</a><br>';
  }
  document.getElementById("id01").innerHTML = out;
}
```

Call myFunction() with myArray as argument:

Example

```
myFunction(myArray);
```

Output:



```

JavaScript Tutorial
HTML Tutorial
CSS Tutorial

```

3: Use an array literal as the argument (instead of the array variable):

Call **myFunction()** with an array **literal** as argument:

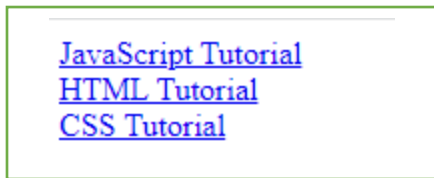
Calling myFunction()

```
myFunction([
  {
    "display": "JavaScript Tutorial",
    "url": "https://www.w3schools.com/js/default.asp"
  },
  {
    "display": "HTML Tutorial",
    "url": "https://www.w3schools.com/html/default.asp"
  }
])
```

```

},
{
"display": "CSS Tutorial",
"url": "https://www.w3schools.com/css/default.asp"
}
]);

```

Output:**4: Put the function in an external js file**

Put the function in a file named **myTutorials.js**:

```

myTutorials.js
myFunction([
{
"display": "JavaScript Tutorial",
"url": "https://www.w3schools.com/js/default.asp"
},
{
"display": "HTML Tutorial",
"url": "https://www.w3schools.com/html/default.asp"
},
{
"display": "CSS Tutorial",
"url": "https://www.w3schools.com/css/default.asp"
}
]);

```

Add the external script to your page (instead of the function call):

Add External Script

```
<script src="myTutorials.js"></script>
```

Output:

[HTML Tutorial](#)
[CSS Tutorial](#)
[JavaScript Tutorial](#)
[SQL Tutorial](#)
[PHP Tutorial](#)
[XML Tutorial](#)