Sachin Sirohi

JSON E-BOOK

Chapter-1

JSON - Introduction

JSON

JSON stands for **J**ava**S**cript **O**bject **N**otation

JSON is a **text format** for storing and transporting data

JSON is "self-describing" and easy to understand

JSON Example

This example is a JSON string:

```
'{"name":"John", "age":30, "car":null}'
```

It defines an object with 3 properties:

- name
- age
- car

Each property has a 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;
```

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation

- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent *

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.

The JSON format was originally specified by Douglas Crockford.

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.

Chapter-2

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"};
```

You can access a JavaScript object like this:

Example

```
// returns John
person.name;
```

Example

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

Example

```
person.name = "Gilbert";
```

Example

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

JavaScript Arrays as JSON

The same way JavaScript objects can be written as JSON, JavaScript arrays can also be written as JSON.

You will learn more about objects and arrays later in this tutorial.

JSON Files

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

Chapter-3

JSON vs XML

Both JSON and XML can be used to receive data from a web server.

The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON Example

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

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

JSON is Like XML Because

- Both JSON and XML are "self describing" (human readable)
- Both JSON and XML are hierarchical (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

JSON is 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 is Better Than XML

XML is much more difficult to parse than JSON.
JSON is parsed into a ready-to-use JavaScript object.

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

Chapter-4

JSON Data Types

Valid Data Types

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 values **cannot** be one of the following data types:

- a function
- a date
- *undefined*

JSON Strings

Strings in JSON must be written in double quotes.

Example

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

JSON Numbers

Numbers in JSON must be an integer or a floating point.

Example

```
{"age":30}
```

JSON Objects

Values in JSON can be objects.**Example**

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

Objects as values in JSON must follow the JSON syntax.

JSON Arrays

Values in JSON can be arrays.

Example

```
{
"employees":["John", "Anna", "Peter"]
}
```

JSON Booleans

Values in JSON can be true/false.

Example

```
{"sale":true}
```

JSON null

Values in JSON can be null.

Example

```
{"middlename":null}
```

Chapter-5

JSON.parse()

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.

Example - Parsing JSON

Imagine we received this text from a web server:

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

Use the JavaScript function `JSON.parse()` to convert text into a JavaScript object:

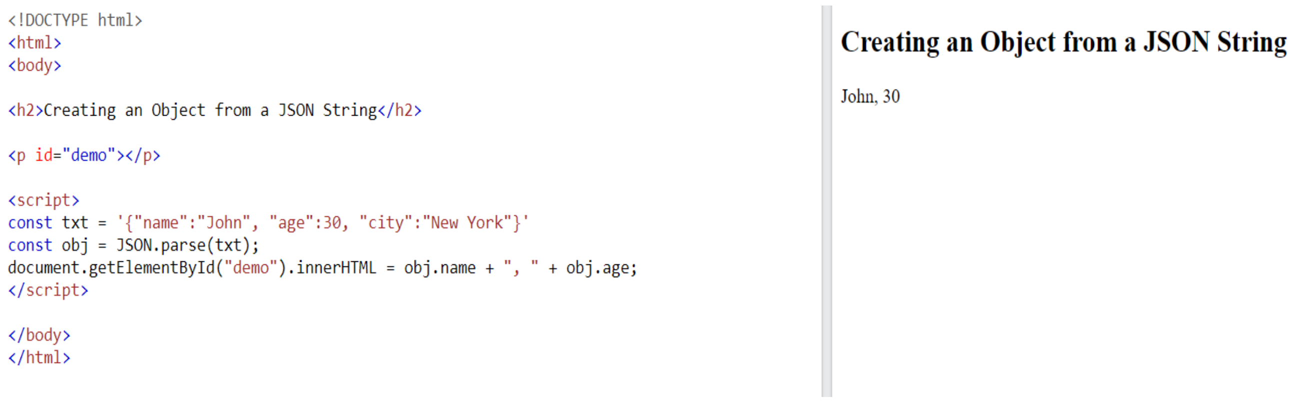
```
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

Make sure the text is in JSON format, or else you will get a syntax error.

Use the JavaScript object in your page:

Example

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = obj.name;
</script>
```



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

<h2>Creating an Object from a JSON String</h2>

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

<script>
const txt = '{"name":"John", "age":30, "city":"New York"}'
const obj = JSON.parse(txt);
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age;
</script>

</body>
</html>
```

```
Creating an Object from a JSON String

John, 30
```

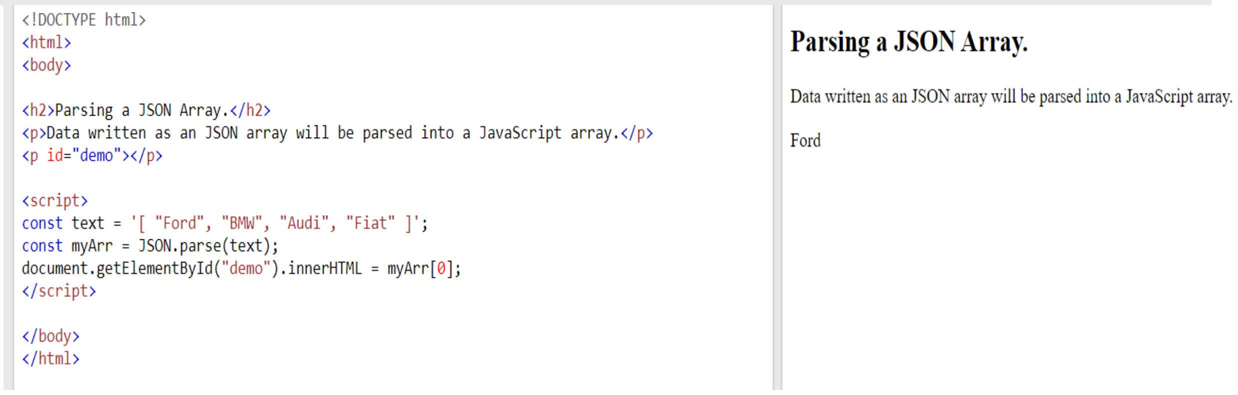
Array as JSON

When using the `JSON.parse()` on a JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

Example

```
const text = '["Ford", "BMW", "Audi", "Fiat"]';
const myArr = JSON.parse(text);
```

Try it Yourself »



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

<h2>Parsing a JSON Array.</h2>
<p>Data written as an JSON array will be parsed into a JavaScript array.</p>
<p id="demo"></p>

<script>
const text = '[ "Ford", "BMW", "Audi", "Fiat" ]';
const myArr = JSON.parse(text);
document.getElementById("demo").innerHTML = myArr[0];
</script>

</body>
</html>
```

Parsing a JSON Array.

Data written as an JSON array will be parsed into a JavaScript array.

Ford

Exceptions

Parsing Dates

Date objects are not allowed in JSON.

If you need to include a date, write it as a string.

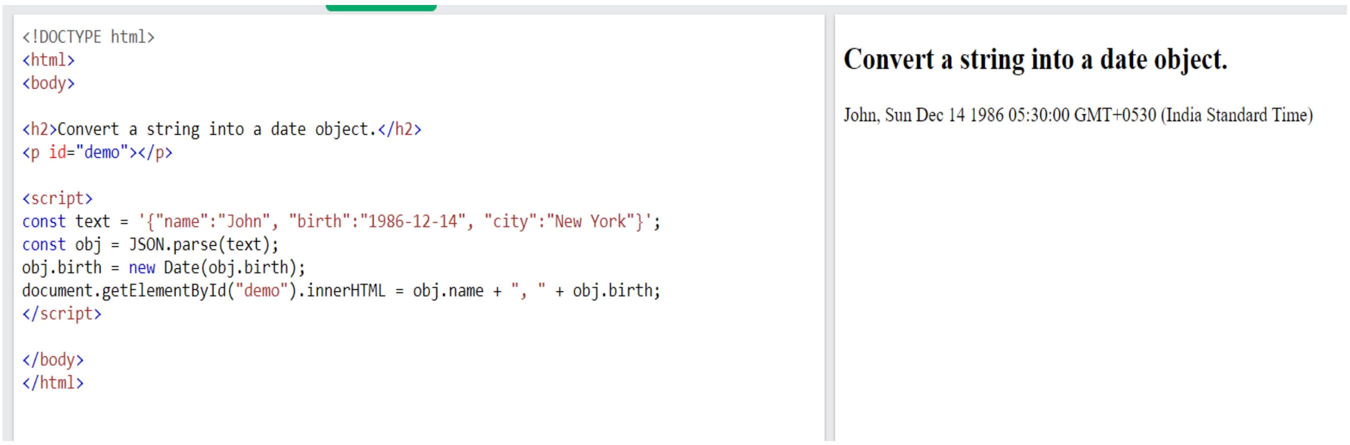
You can convert it back into a date object later:

Example

Convert a string into a date:

```
const text = '{"name":"John", "birth":"1986-12-14", "city":"New
York"}';
const obj = JSON.parse(text);
obj.birth = new Date(obj.birth);

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

Try it Yourself »



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

<h2>Convert a string into a date object.</h2>
<p id="demo"></p>

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

</body>
</html>
```

Convert a string into a date object.

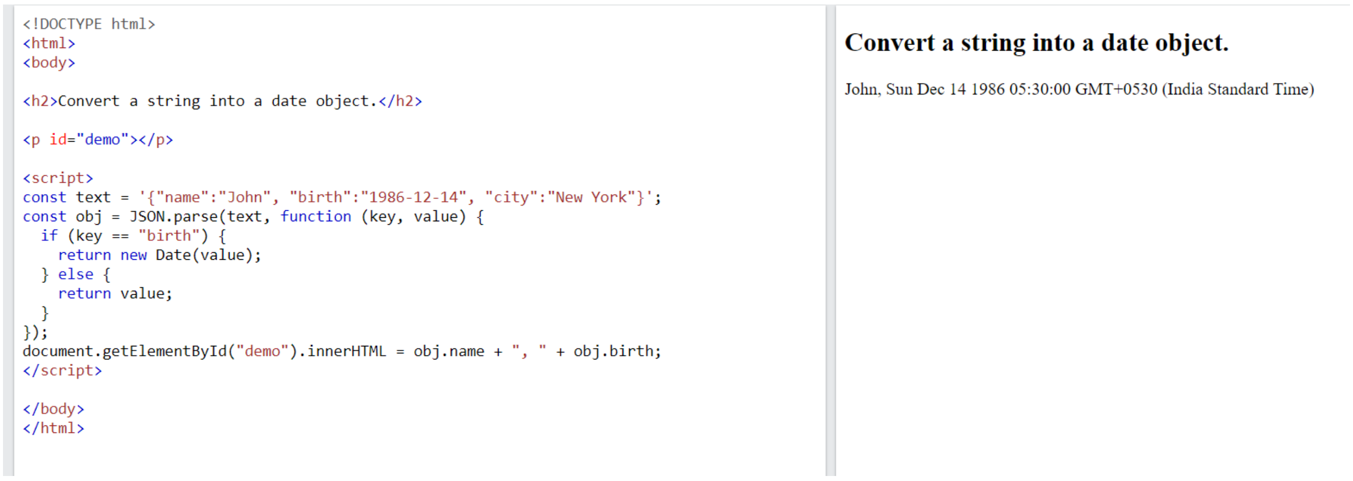
John, Sun Dec 14 1986 05:30:00 GMT+0530 (India Standard Time)

Or, you can use the second parameter, of the `JSON.parse()` function, called *reviver*.

The *reviver* parameter is a function that checks each property, before returning the value.

ExampleConvert a string into a date, using the *reviver* function:

```
const text = '{"name":"John", "birth":"1986-12-14", "city":"New
York"}';
const 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;
```



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

<h2>Convert a string into a date object.</h2>

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

<script>
const text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';
const 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)

Parsing Functions

Functions are not allowed in JSON.

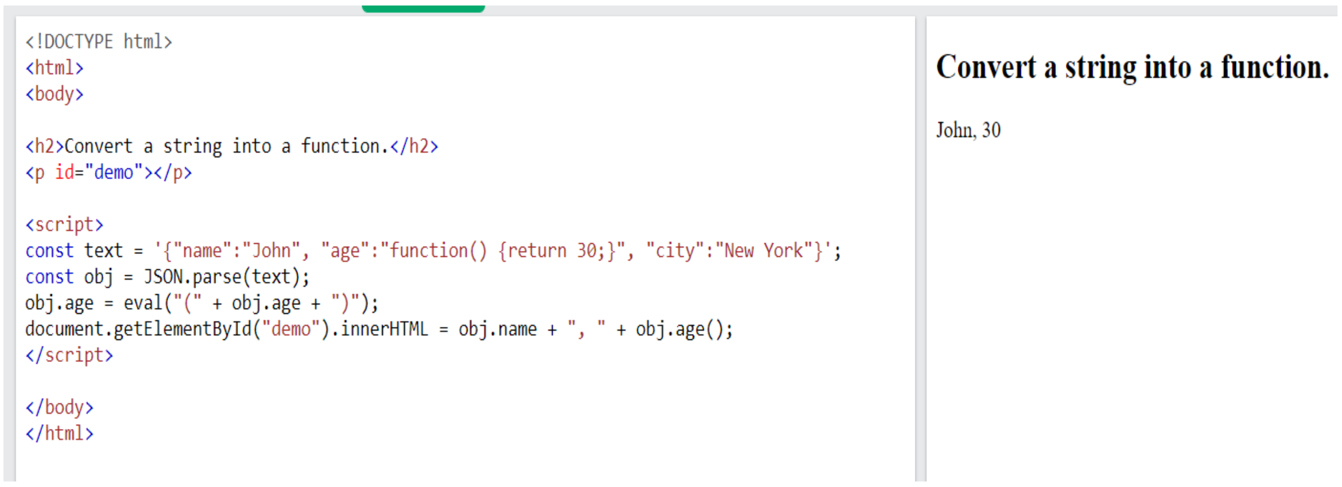
If you need to include a function, write it as a string.

You can convert it back into a function later:

Example

Convert a string into a function:

```
const text = '{"name":"John", "age":"function () {return 30;}",
"city":"New York"}';
const obj = JSON.parse(text);
obj.age = eval("(" + obj.age + ")");

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



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

<h2>Convert a string into a function.</h2>
<p id="demo"></p>

<script>
const text = '{"name":"John", "age":"function() {return 30;}", "city":"New York"}';
const obj = JSON.parse(text);
obj.age = eval("(" + obj.age + ")");
document.getElementById("demo").innerHTML = obj.name + ", " + obj.age();
</script>

</body>
</html>
```

Convert a string into a function.

John, 30

Chapter-6

JSON.stringify()

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()`.

Stringify a JavaScript Object

Imagine we have this object in JavaScript:

```
const obj = {name: "John", age: 30, city: "New York"};
```

Use the JavaScript function `JSON.stringify()` to convert it into a string.

```
const myJSON = JSON.stringify(obj);
```

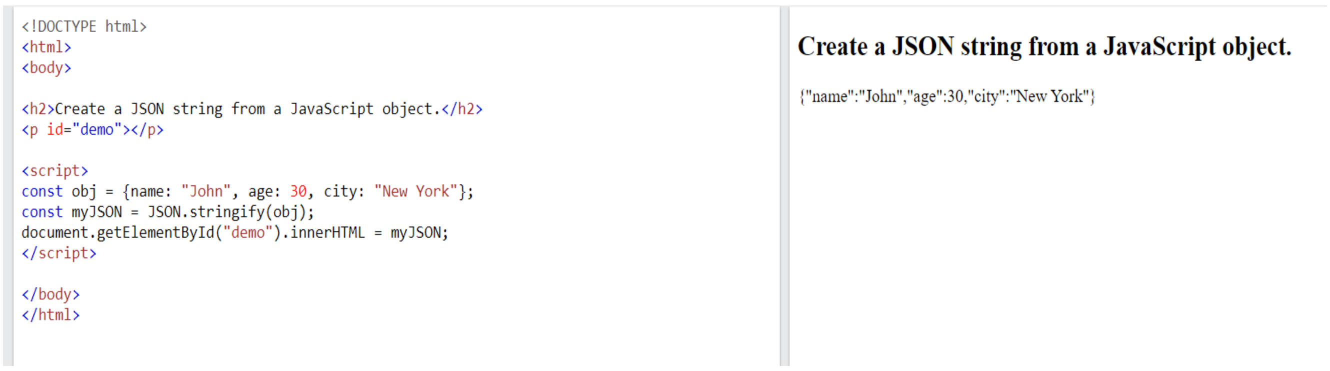
The result will be a string following the JSON notation.

`myJSON` is now a string, and ready to be sent to a server:

Example

```
const obj = {name: "John", age: 30, city: "New York"};
const myJSON = JSON.stringify(obj);
```

Try it Yourself »



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

<h2>Create a JSON string from a JavaScript object.</h2>
<p id="demo"></p>

<script>
const obj = {name: "John", age: 30, city: "New York"};
const myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

Create a JSON string from a JavaScript object.

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

You will learn how to send JSON to a server in the next chapters.

Stringify a JavaScript Array

It is also possible to stringify JavaScript arrays:

Imagine we have this array in JavaScript:

```
const arr = ["John", "Peter", "Sally", "Jane"];
```

Use the JavaScript function `JSON.stringify()` to convert it into a string.

```
const myJSON = JSON.stringify(arr);
```

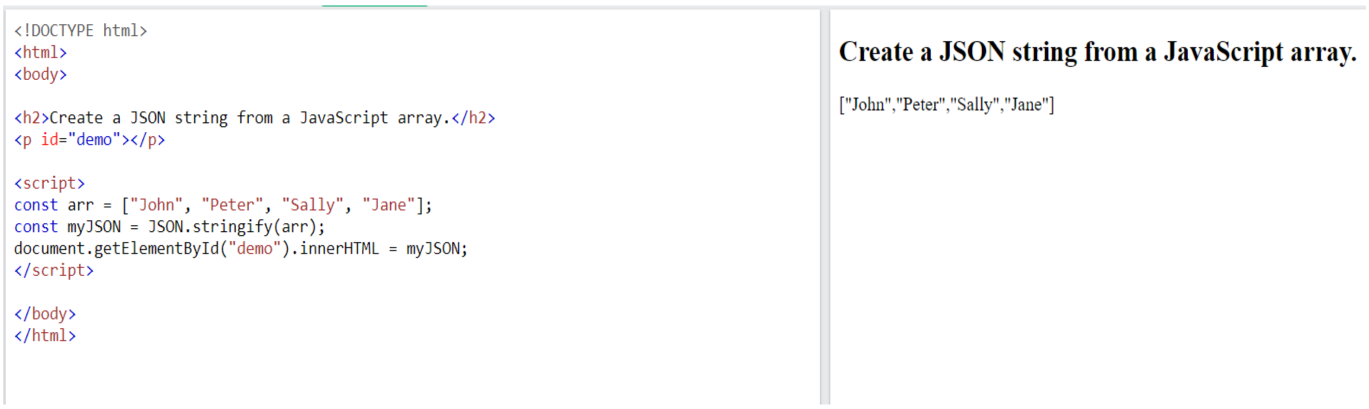
The result will be a string following the JSON notation.

`myJSON` is now a string, and ready to be sent to a server:

Example

```
const arr = ["John", "Peter", "Sally", "Jane"];
const myJSON = JSON.stringify(arr);
```

Try it Yourself »



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

<h2>Create a JSON string from a JavaScript array.</h2>
<p id="demo"></p>

<script>
const arr = ["John", "Peter", "Sally", "Jane"];
const myJSON = JSON.stringify(arr);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

Create a JSON string from a JavaScript array.

```
["John","Peter","Sally","Jane"]
```

You will learn how to send a JSON string to a server in the next chapters.

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.

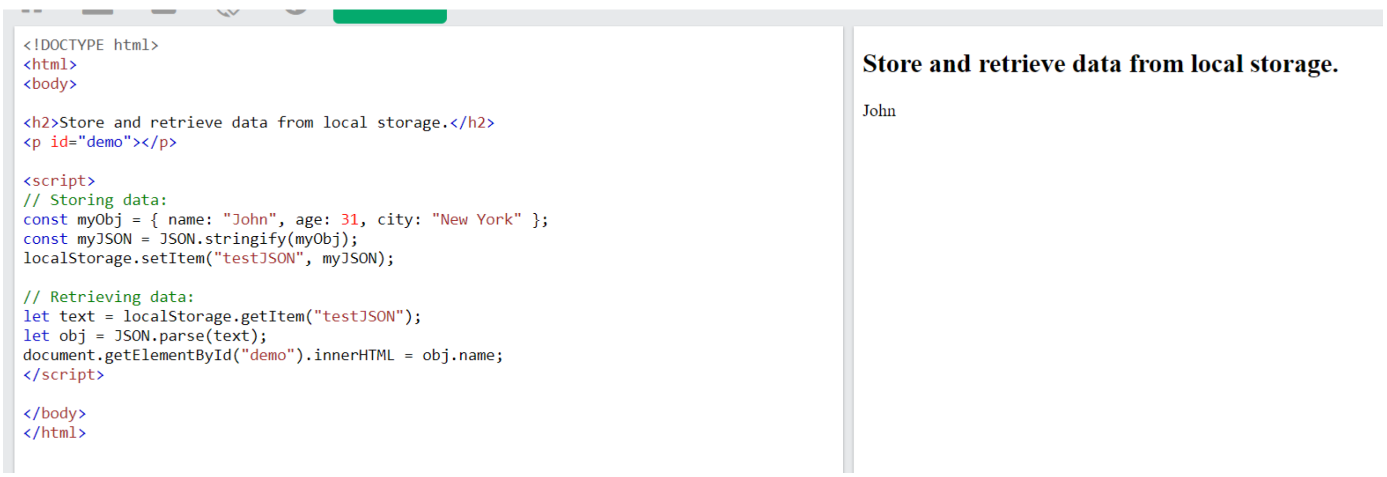
Example

Storing data in local storage

```
// Storing data:
const myObj = {name: "John", age: 31, city: "New York"};
const myJSON = JSON.stringify(myObj);
```

```
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
let text = localStorage.getItem("testJSON");
let obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```

Try it Yourself »



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

<h2>Store and retrieve data from local storage.</h2>
<p id="demo"></p>

<script>
// Storing data:
const myObj = { name: "John", age: 31, city: "New York" };
const myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
let text = localStorage.getItem("testJSON");
let obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
</script>

</body>
</html>
```

Store and retrieve data from local storage.

John

Exceptions

Stringify Dates

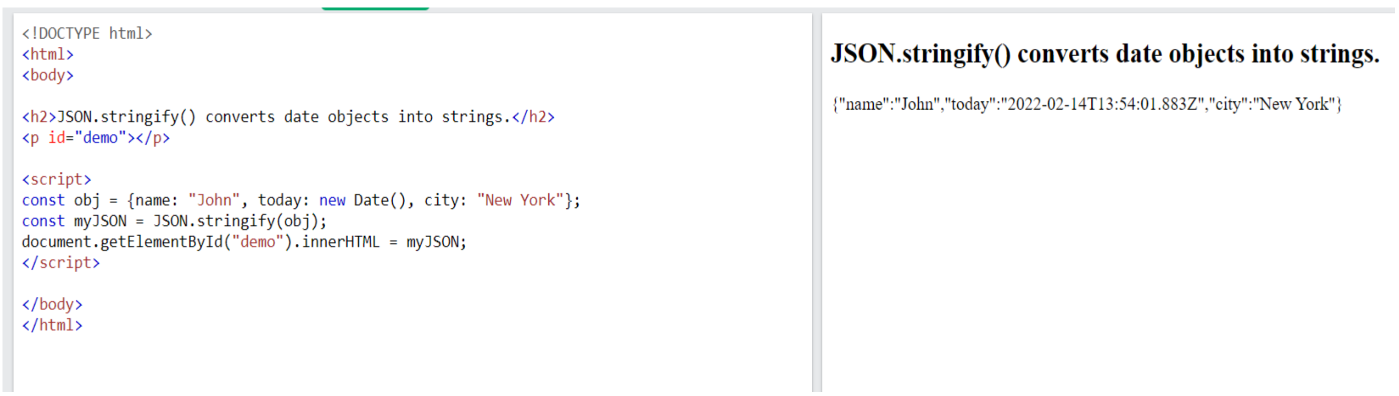
In JSON, date objects are not allowed. The `JSON.stringify()` function will convert any dates into strings.

Example

```
const obj = {name: "John", today: new Date(), city : "New York"};
const myJSON = JSON.stringify(obj);
```

Try it Yourself »

You can convert the string back into a date object at the receiver.



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

<h2>JSON.stringify() converts date objects into strings.</h2>
<p id="demo"></p>

<script>
const obj = {name: "John", today: new Date(), city: "New York"};
const myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

JSON.stringify() converts date objects into strings.

```
{"name":"John","today":"2022-02-14T13:54:01.883Z","city":"New York"}
```

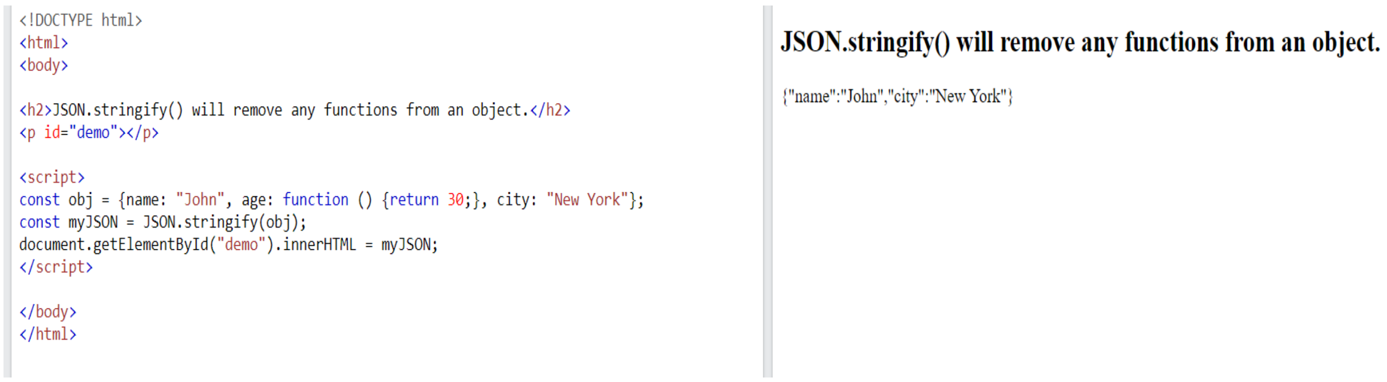
Stringify Functions

In JSON, functions are not allowed as object values.The `JSON.stringify()` function will remove any functions from a JavaScript object, both the key and the value:

Example

```
const obj = {name: "John", age: function () {return 30;}, city: "New
York"};
const myJSON = JSON.stringify(obj);
```

Try it Yourself »



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

<h2>JSON.stringify() will remove any functions from an object.</h2>
<p id="demo"></p>

<script>
const obj = {name: "John", age: function () {return 30;}, city: "New York"};
const myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

JSON.stringify() will remove any functions from an object.

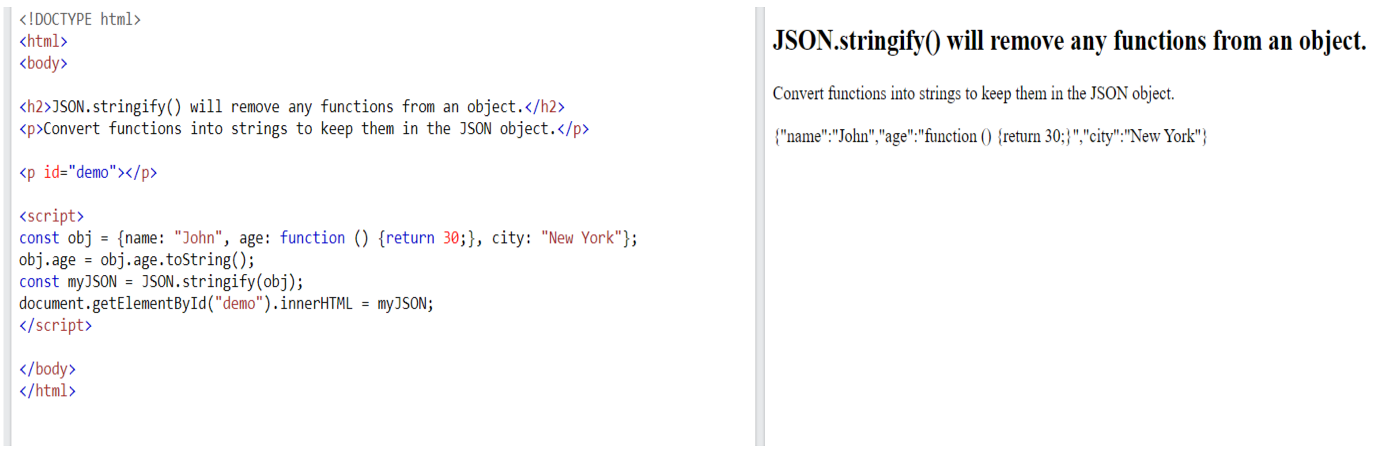
```
{"name":"John","city":"New York"}
```

This can be omitted if you convert your functions into strings before running the `JSON.stringify()` function.

Example

```
const obj = {name: "John", age: function () {return 30;}, city: "New
York"};
obj.age = obj.age.toString();
const myJSON = JSON.stringify(obj);
```

Try it Yourself »



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

<h2>JSON.stringify() will remove any functions from an object.</h2>
<p>Convert functions into strings to keep them in the JSON object.</p>

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

<script>
const obj = {name: "John", age: function () {return 30;}, city: "New York"};
obj.age = obj.age.toString();
const myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
</script>

</body>
</html>
```

JSON.stringify() will remove any functions from an object.

Convert functions into strings to keep them in the JSON object.

```
{"name":"John","age":"function () {return 30;}","city":"New York"}
```

Chapter-7

JSON Object Literals

This is a JSON string:

```
'{"name":"John", "age":30, "car":null}'
```

Inside the JSON string there is a JSON object literal:

```
{"name":"John", "age":30, "car":null}
```

JSON object literals are surrounded by curly braces {}.

JSON object literals contains key/value pairs.

Keys and values are separated by a colon.

Keys must be strings, and values must be a valid JSON data type:

- string
- number
- object
- array
- boolean
- null

Each key/value pair is separated by a comma.

It is a common mistake to call a JSON object literal "a JSON object".

JSON cannot be an object. JSON is a string format.

The data is only JSON when it is in a string format. When it is converted to a JavaScript variable, it becomes a JavaScript object.

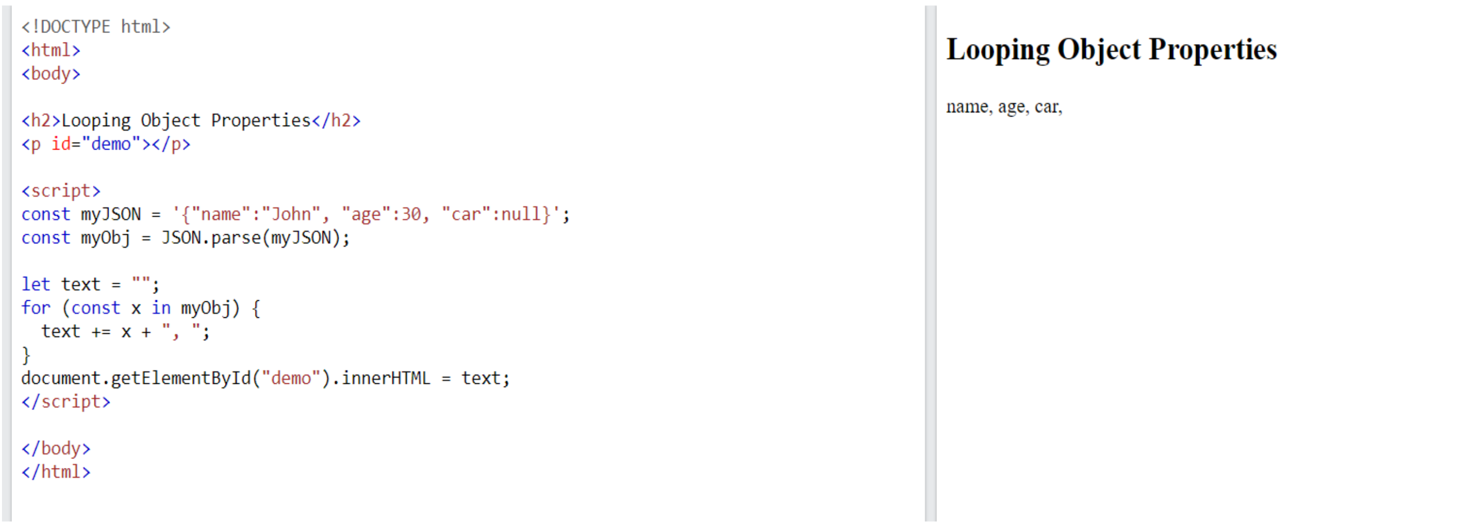
JavaScript Objects

You can create a JavaScript object from a JSON object literal:

Example

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

Try it Yourself »



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

<h2>Looping Object Properties</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += x + ", ";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Looping Object Properties

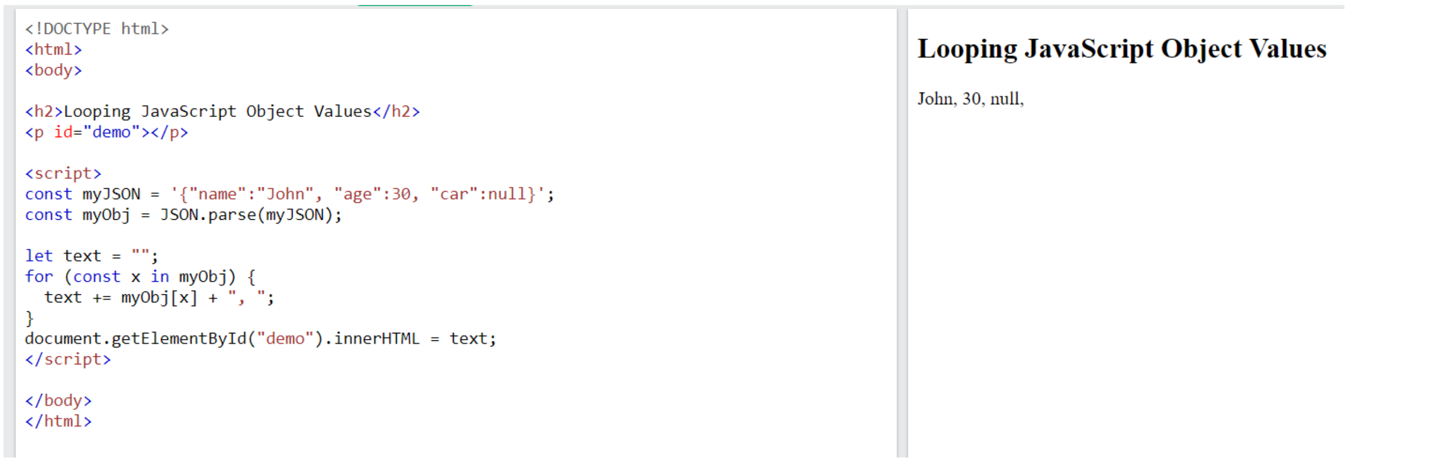
name, age, car,

Normally, you create a JavaScript object by parsing a JSON string:

Example

```
myJSON = '{"name":"John", "age":30, "car":null}';
myObj = JSON.parse(myJSON);
```

Try it Yourself »



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

<h2>Looping JavaScript Object Values</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += myObj[x] + ", ";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Looping JavaScript Object Values

John, 30, null,

Accessing Object Values

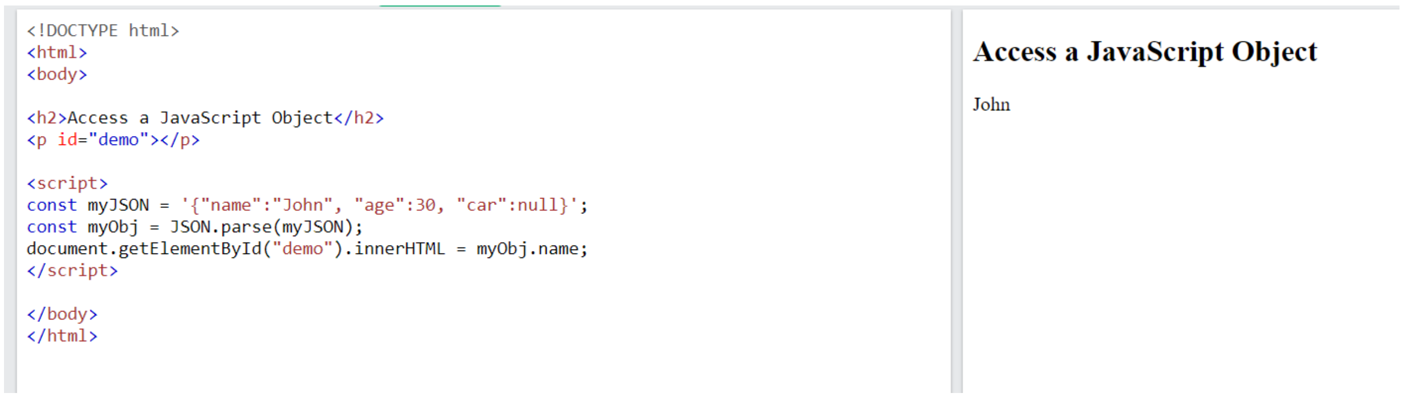
You can access object values by using dot (.) notation:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
x = myObj.name;
```

Try it Yourself »

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



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

<h2>Access a JavaScript Object</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
</script>

</body>
</html>
```

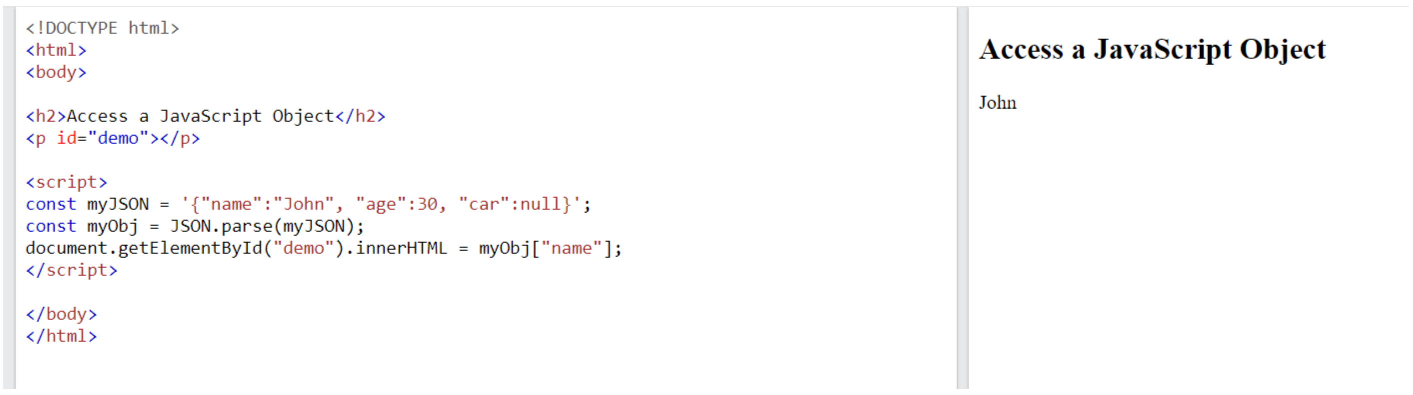
Access a JavaScript Object

John

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
x = myObj["name"];
```

Try it Yourself »



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

<h2>Access a JavaScript Object</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj["name"];
</script>

</body>
</html>
```

Access a JavaScript Object

John

Looping an Object

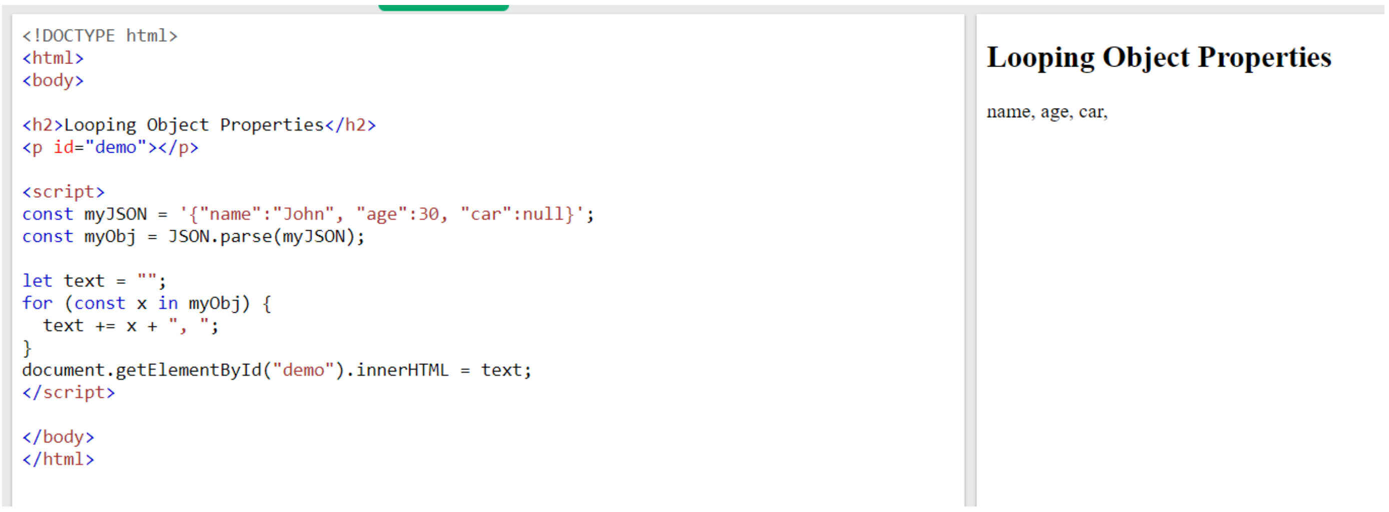
You can loop through object properties with a for-in loop:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += x + ", ";
}
```

Try it Yourself »

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

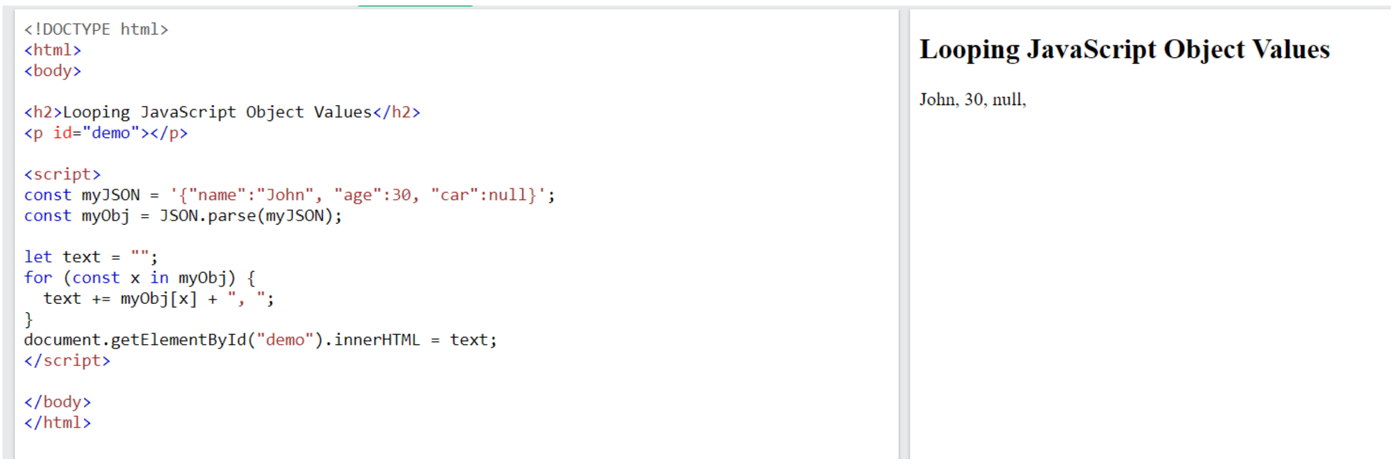


```
<!DOCTYPE html>
<html>
<body>

<h2>Looping Object Properties</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += x + ", ";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

```
Looping Object Properties

name, age, car,
```

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += myObj[x] + ", ";
}
```



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

<h2>Looping JavaScript Object Values</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += myObj[x] + ", ";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

```
Looping JavaScript Object Values

John, 30, null,
```

Chapter-8

JSON Array Literals

This is a JSON string:

```
'["Ford", "BMW", "Fiat"]'
```

Inside the JSON string there is a JSON array literal:

```
["Ford", "BMW", "Fiat"]
```

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

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

In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and *undefined.*

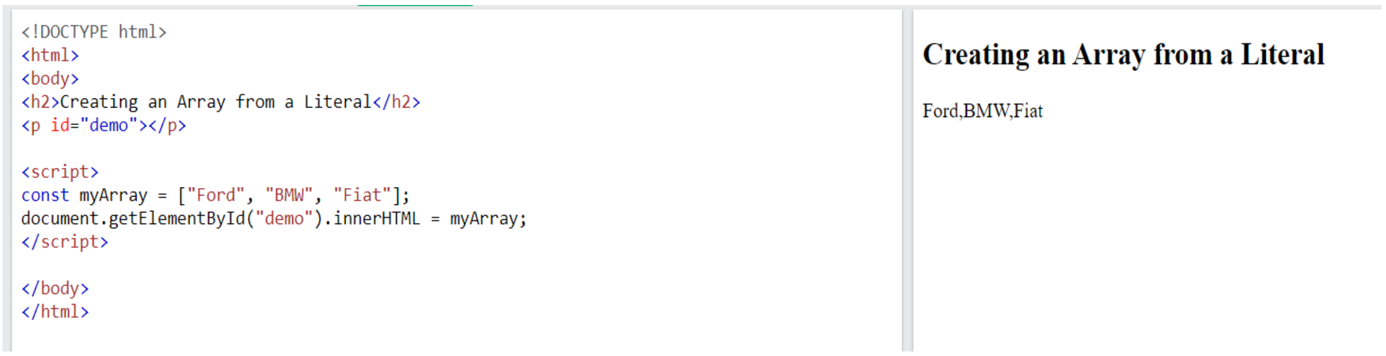
JavaScript Arrays

You can create a JavaScript array from a literal:

Example

```
myArray = ["Ford", "BMW", "Fiat"];
```

Try it Yourself »



```
<!DOCTYPE html>
<html>
<body>
<h2>Creating an Array from a Literal</h2>
<p id="demo"></p>

<script>
const myArray = ["Ford", "BMW", "Fiat"];
document.getElementById("demo").innerHTML = myArray;
</script>

</body>
</html>
```

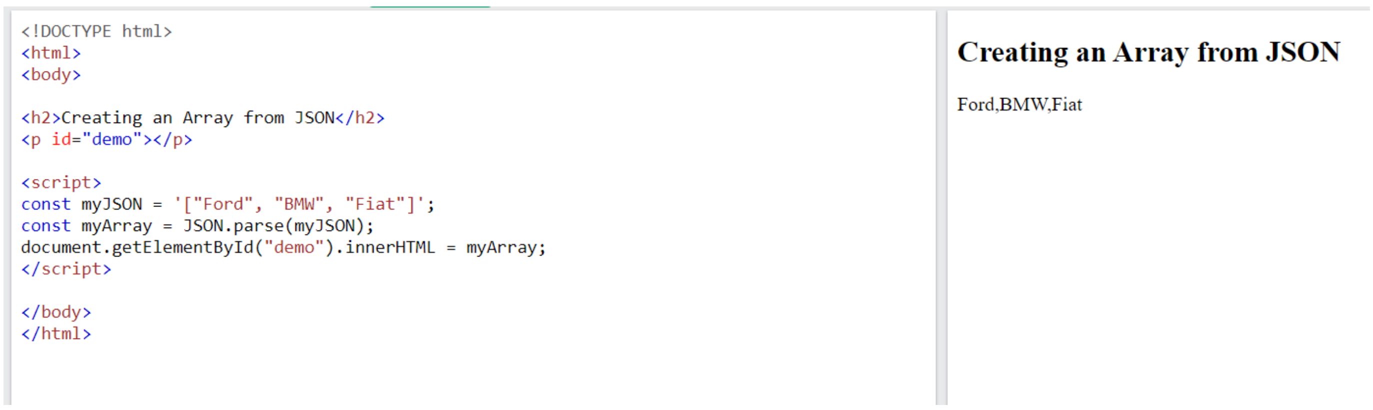
```
Creating an Array from a Literal

Ford,BMW,Fiat
```

You can create a JavaScript array by parsing a JSON string:

Example

```
myJSON = '["Ford", "BMW", "Fiat"]';
myArray = JSON.Parse(myJSON);
```

Try it Yourself »



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

<h2>Creating an Array from JSON</h2>
<p id="demo"></p>

<script>
const myJSON = '["Ford", "BMW", "Fiat"]';
const myArray = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myArray;
</script>

</body>
</html>
```

Creating an Array from JSON

Ford,BMW,Fiat

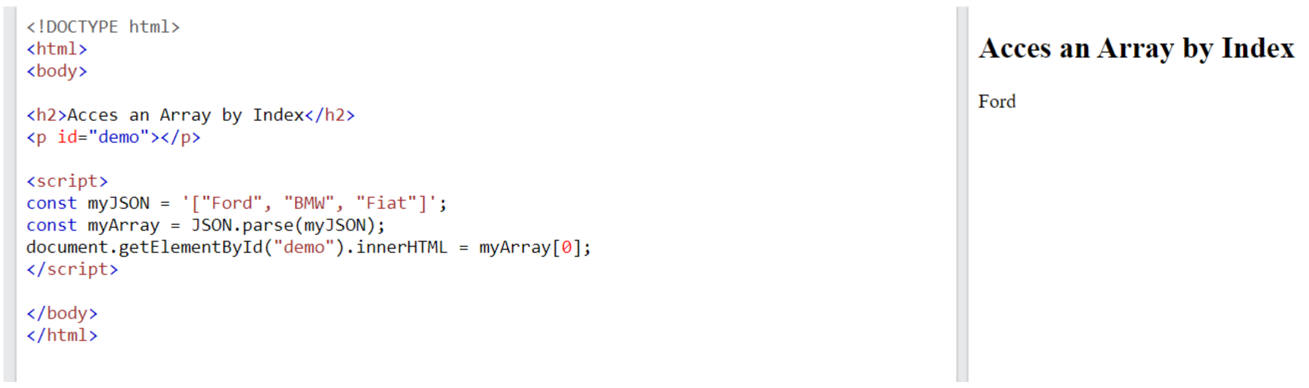
Accessing Array Values

You access array values by index:

Example

```
myArray[0];
```

Try it Yourself »



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

<h2>Acces an Array by Index</h2>
<p id="demo"></p>

<script>
const myJSON = '["Ford", "BMW", "Fiat"]';
const myArray = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myArray[0];
</script>

</body>
</html>
```

Acces an Array by Index

Ford

Arrays in Objects

Objects can contain arrays:

Example

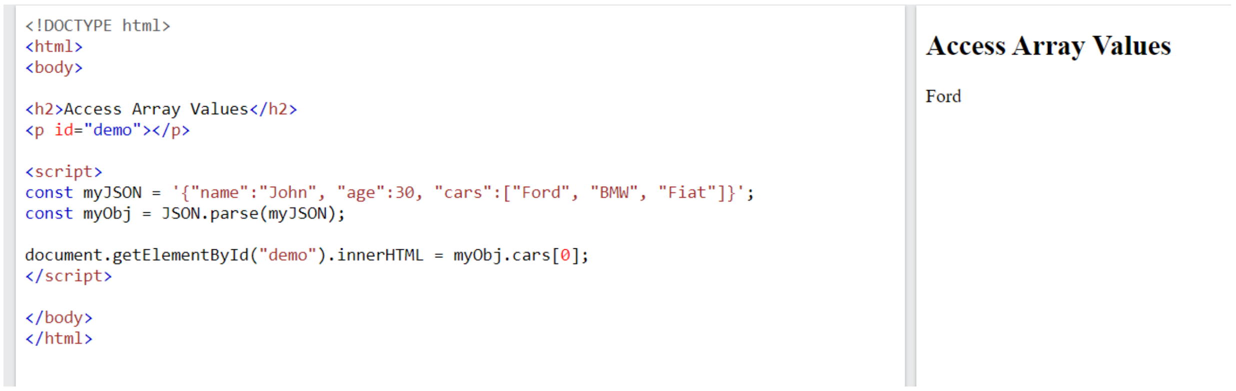
```
{
"name":"John",
"age":30,
"cars":["Ford", "BMW", "Fiat"]
}
```

You access array values by index:

Example

```
myObj.cars[0];
```

Try it Yourself »



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

<h2>Access Array Values</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "cars":["Ford", "BMW", "Fiat"]}';
const myObj = JSON.parse(myJSON);

document.getElementById("demo").innerHTML = myObj.cars[0];
</script>

</body>
</html>
```

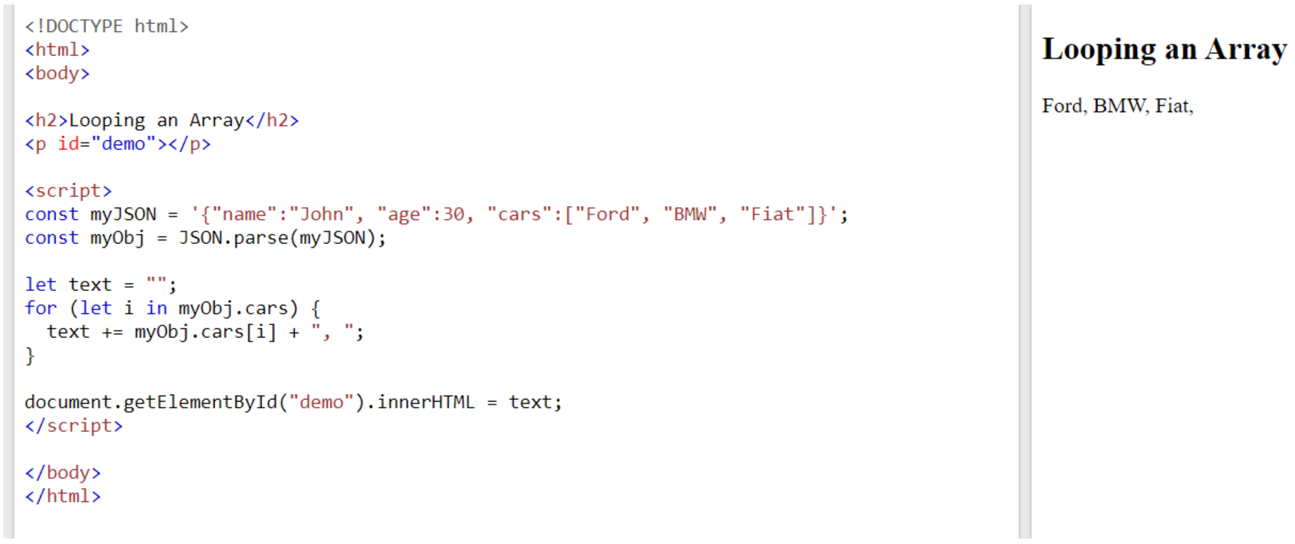
```
Access Array Values

Ford
```

Looping Through an Array

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

Example

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



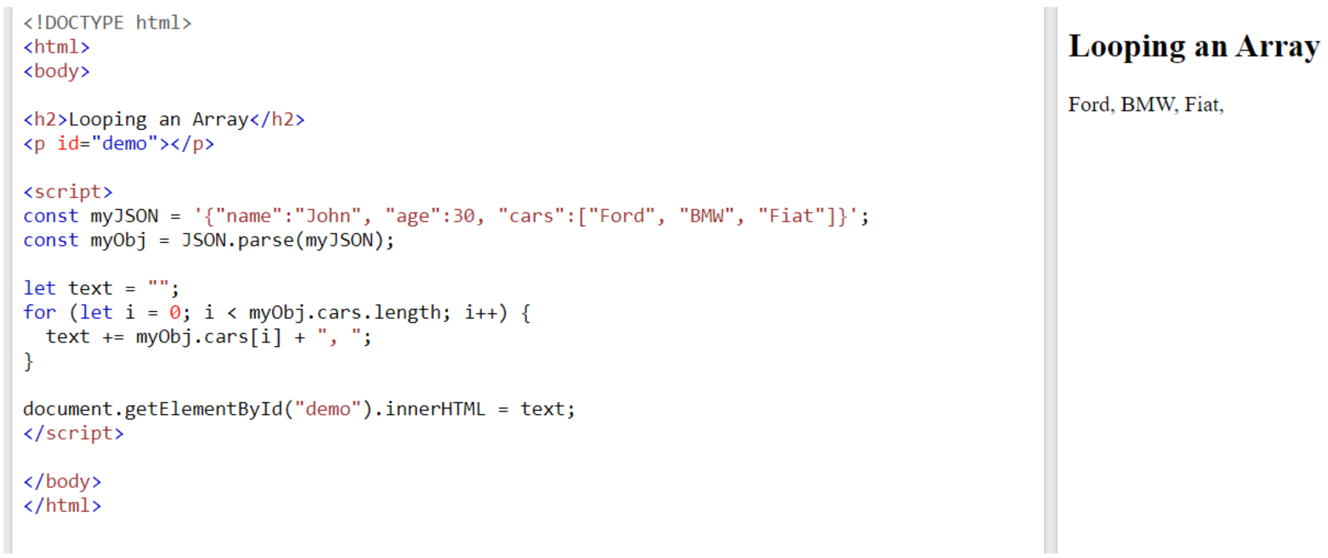
```
<!DOCTYPE html>
<html>
<body>

<h2>Looping an Array</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "cars":["Ford", "BMW", "Fiat"]}';
const myObj = JSON.parse(myJSON);

let text = "";
for (let i in myObj.cars) {
  text += myObj.cars[i] + ", ";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

```
Looping an Array

Ford, BMW, Fiat,
```

Or you can use a `for` loop:

Example

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



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

<h2>Looping an Array</h2>
<p id="demo"></p>

<script>
const myJSON = '{"name":"John", "age":30, "cars":["Ford", "BMW", "Fiat"]}';
const myObj = JSON.parse(myJSON);

let text = "";
for (let i = 0; i < myObj.cars.length; i++) {
  text += myObj.cars[i] + ", ";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

```
Looping an Array

Ford, BMW, Fiat,
```

Chapter-9

JSON Server

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.

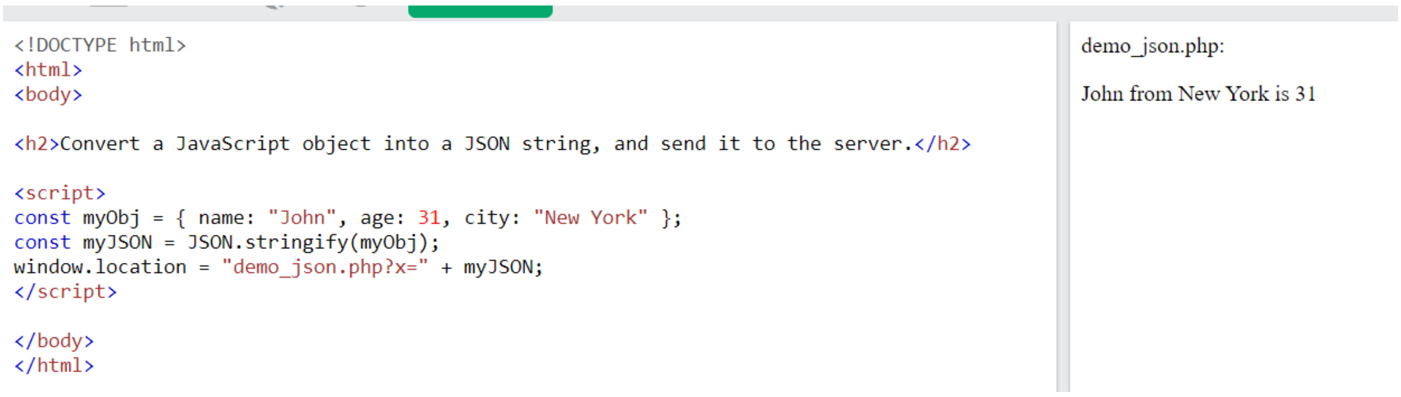
Sending Data

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

Example

```
const myObj = {name: "John", age: 31, city: "New York"};
const myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Try it Yourself »



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

<h2>Convert a JavaScript object into a JSON string, and send it to the server.</h2>

<script>
const myObj = { name: "John", age: 31, city: "New York" };
const myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
</script>

</body>
</html>
```

```
demo_json.php:

John from New York is 31
```

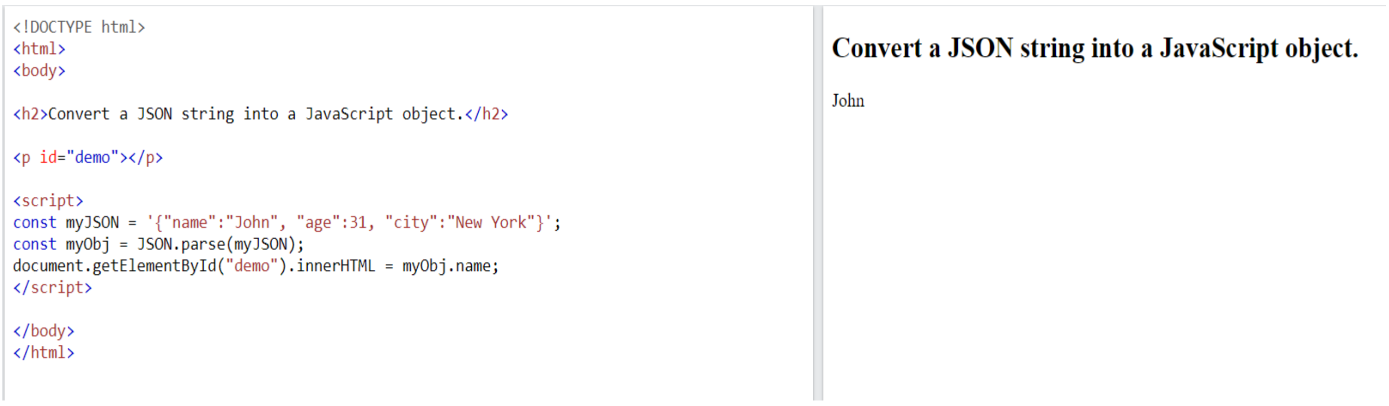
Receiving Data

If you receive data in JSON format, you can easily convert it into a JavaScript object:

Example

```
const myJSON = '{"name":"John", "age":31, "city":"New York"}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Try it Yourself »



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

<h2>Convert a JSON string into a JavaScript object.</h2>

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

<script>
const myJSON = '{"name":"John", "age":31, "city":"New York"}';
const myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
</script>

</body>
</html>
```

Convert a JSON string into a JavaScript object.

John

JSON From a 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.

Example

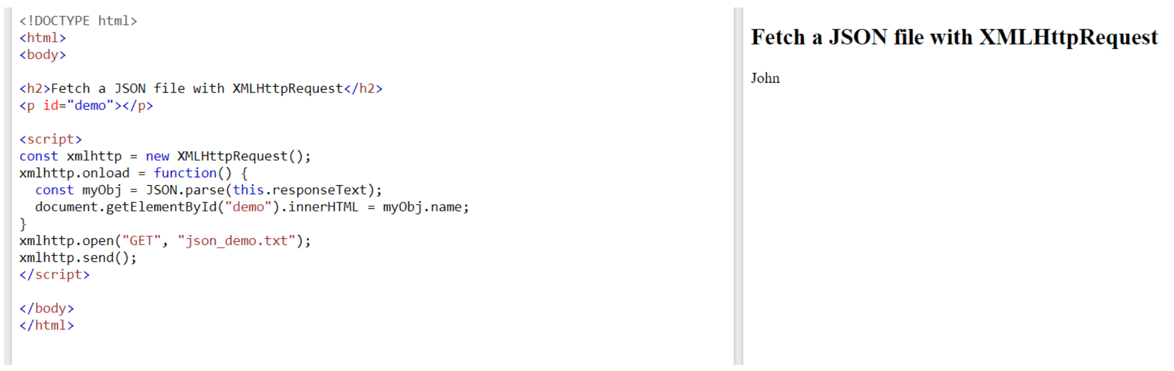
Use the XMLHttpRequest to get data from the server:

```
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myObj.name;
};
xmlhttp.open("GET", "json_demo.txt");
xmlhttp.send();
```

Try it Yourself » Take a look at json_demo.txt

Json_demo.txt

```
{
    "name":"John",
    "age":31,
    "pets":[
        { "animal":"dog", "name":"Fido" },
        { "animal":"cat", "name":"Felix" },
        { "animal":"hamster", "name":"Lightning" }
    ]
}
```



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

<h2>Fetch a JSON file with XMLHttpRequest</h2>
<p id="demo"></p>

<script>
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myObj.name;
}
xmlhttp.open("GET", "json_demo.txt");
xmlhttp.send();
</script>

</body>
</html>
```

Fetch a JSON file with XMLHttpRequest

John

Array as JSON

When using the `JSON.parse()` on JSON derived from an array, the method will return a JavaScript array, instead of a JavaScript object.

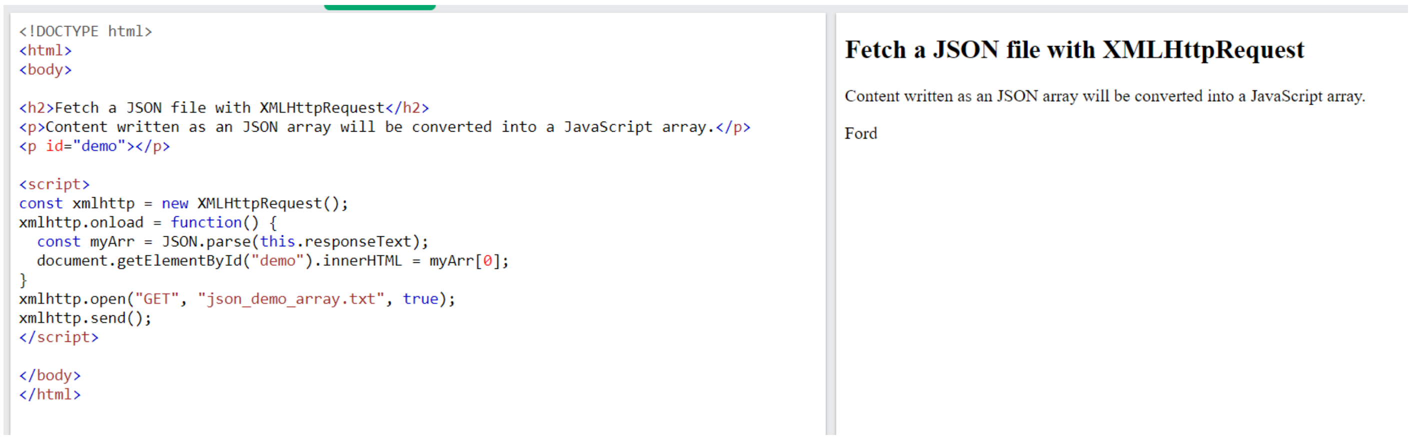
Example

JSON returned from a server as an array:

```
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myArr = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myArr[0];
  }
}
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
```

json_demo_array.txt

```
[ "Ford", "BMW", "Audi", "Fiat" ]
```



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

<h2>Fetch a JSON file with XMLHttpRequest</h2>
<p>Content written as an JSON array will be converted into a JavaScript array.</p>
<p id="demo"></p>

<script>
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myArr = JSON.parse(this.responseText);
  document.getElementById("demo").innerHTML = myArr[0];
}
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
</script>

</body>
</html>
```

Fetch a JSON file with XMLHttpRequest

Content written as an JSON array will be converted into a JavaScript array.

Ford