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.

JSON Syntax

❮ PreviousNext ❯

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 vs XML

❮ PreviousNext ❯

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

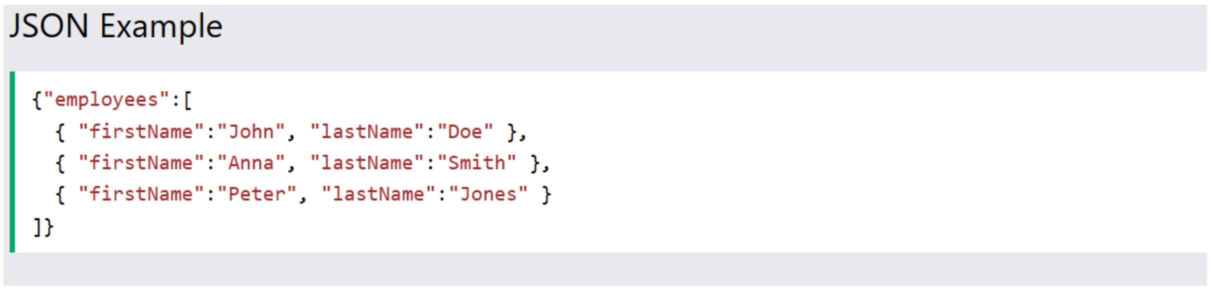
The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON Example

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

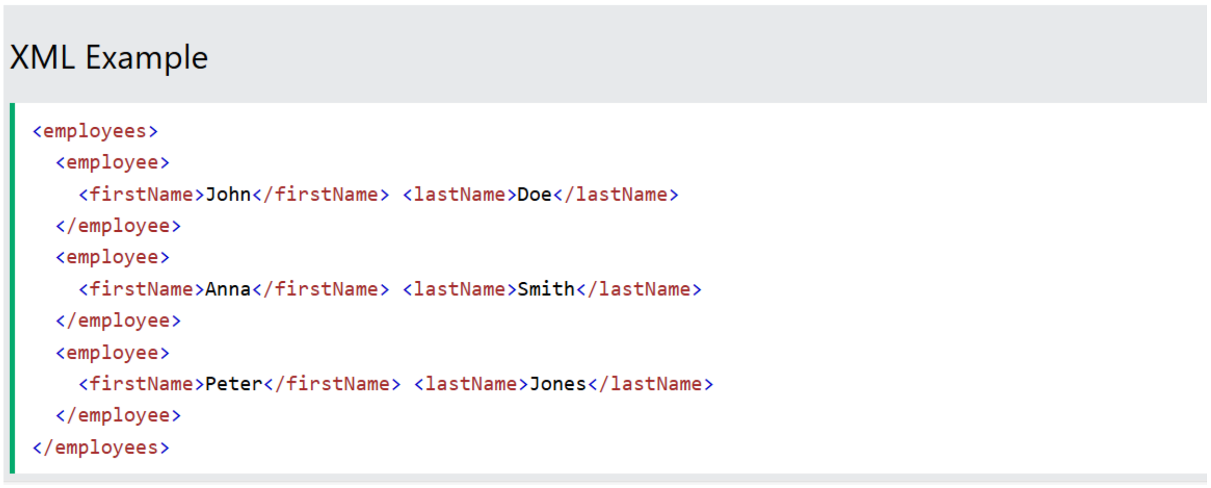
XML Example

```
<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 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

JSON Data Types

❮ PreviousNext ❯

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 Object Literals

❮ PreviousNext ❯

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.

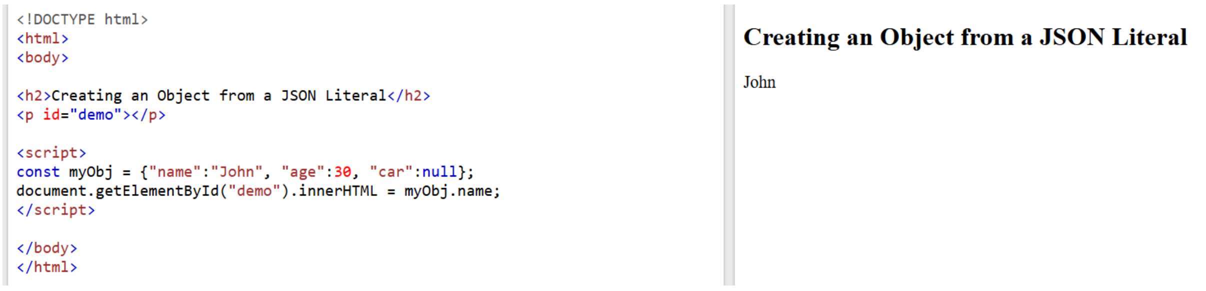
The data is only JSON when it is in a string format. When it is converted to a JavaScript variable, it becomes a JavaScript object.

JavaScript Objects

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

Example

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



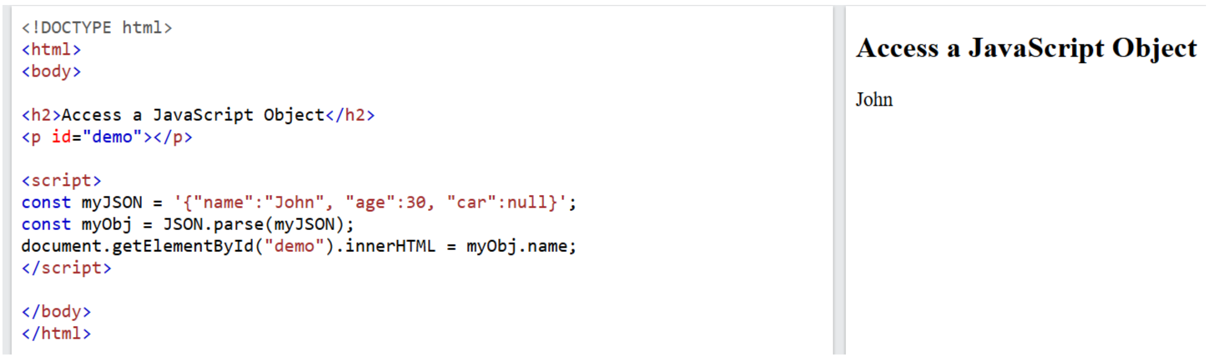
```
<!DOCTYPE html>
<html>
<body>

<h2>Creating an Object from a JSON Literal</h2>
<p id="demo"></p>

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

</body>
</html>
```

```
Creating an Object from a JSON Literal

John
```

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

Example

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



```
<!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
```

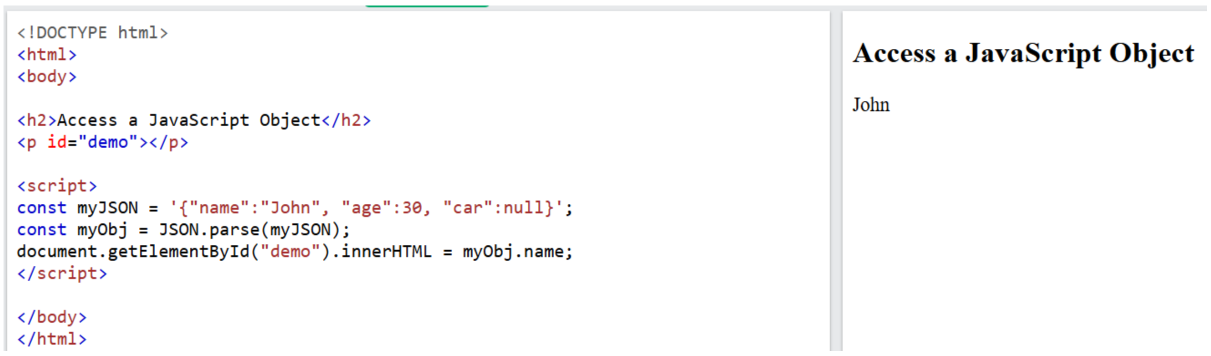
Accessing Object Values

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;



```
<!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>
```

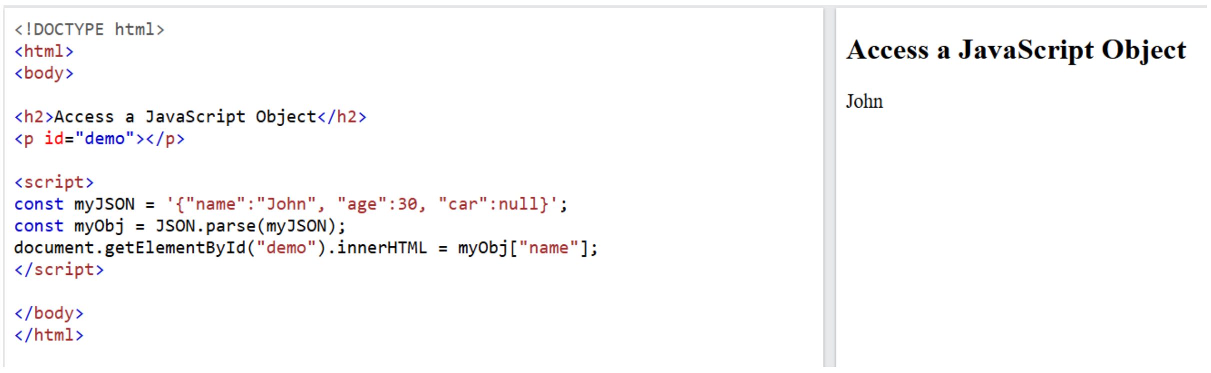
Access a JavaScript Object

John

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

Example

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



```
<!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

ADVERTISEMENT

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*:

Example

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

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

JSON Array Literals

❮ PreviousNext ❯

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*.

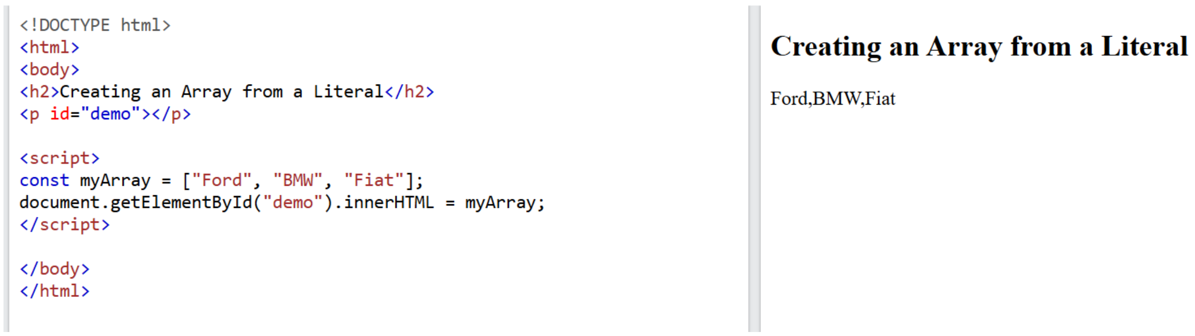
JavaScript Arrays

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>
```

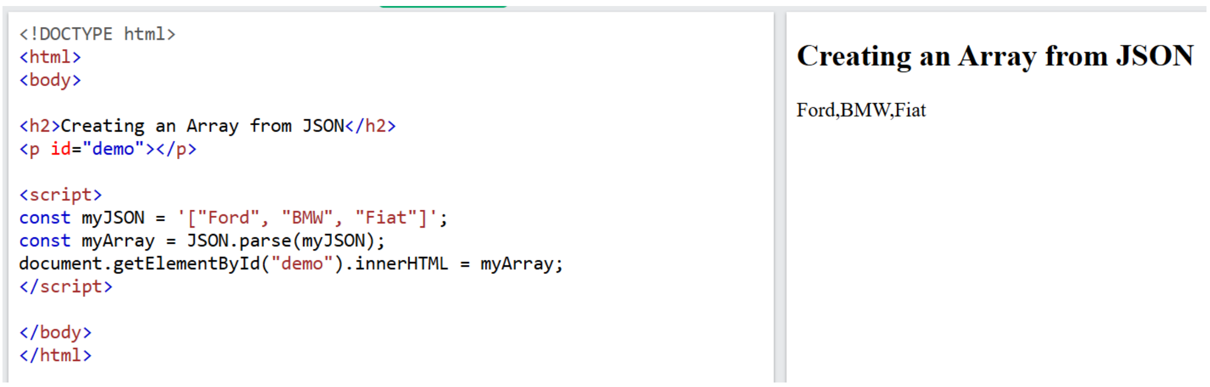
Creating an Array from a Literal

Ford,BMW,Fiat

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

Example

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



```html
<!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>
```

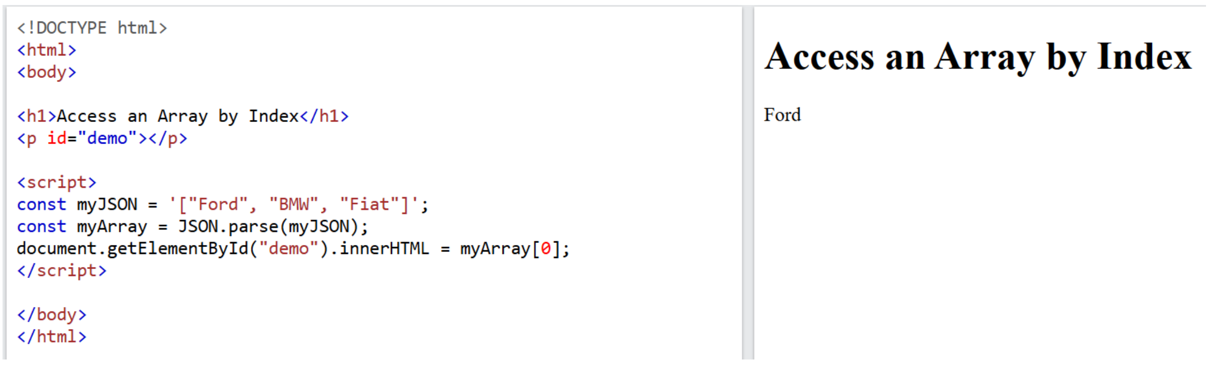
```
Creating an Array from JSON

Ford,BMW,Fiat
```

Accessing Array Values

You access array values by index:

Example

myArray[0];



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

<h1>Access an Array by Index</h1>
<p id="demo"></p>

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

</body>
</html>
```

```
Access an Array by Index

Ford
```

Arrays in Objects

Objects can contain arrays:

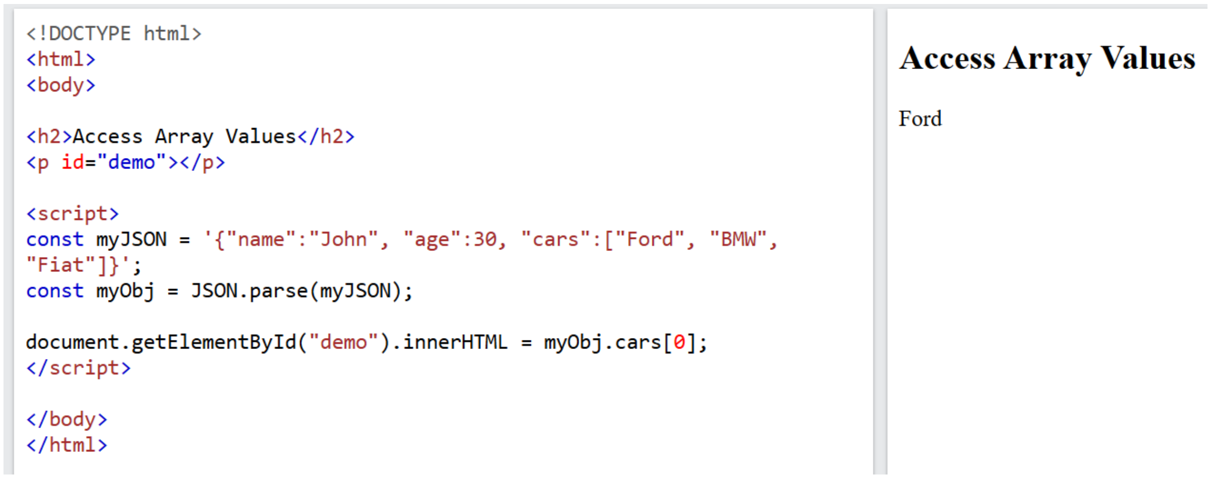
Example

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

You access array values by index:

Example

myObj.cars[0];



```
<!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>
```

JSON HTML

❮ PreviousNext ❯

JSON can very easily be translated into JavaScript.

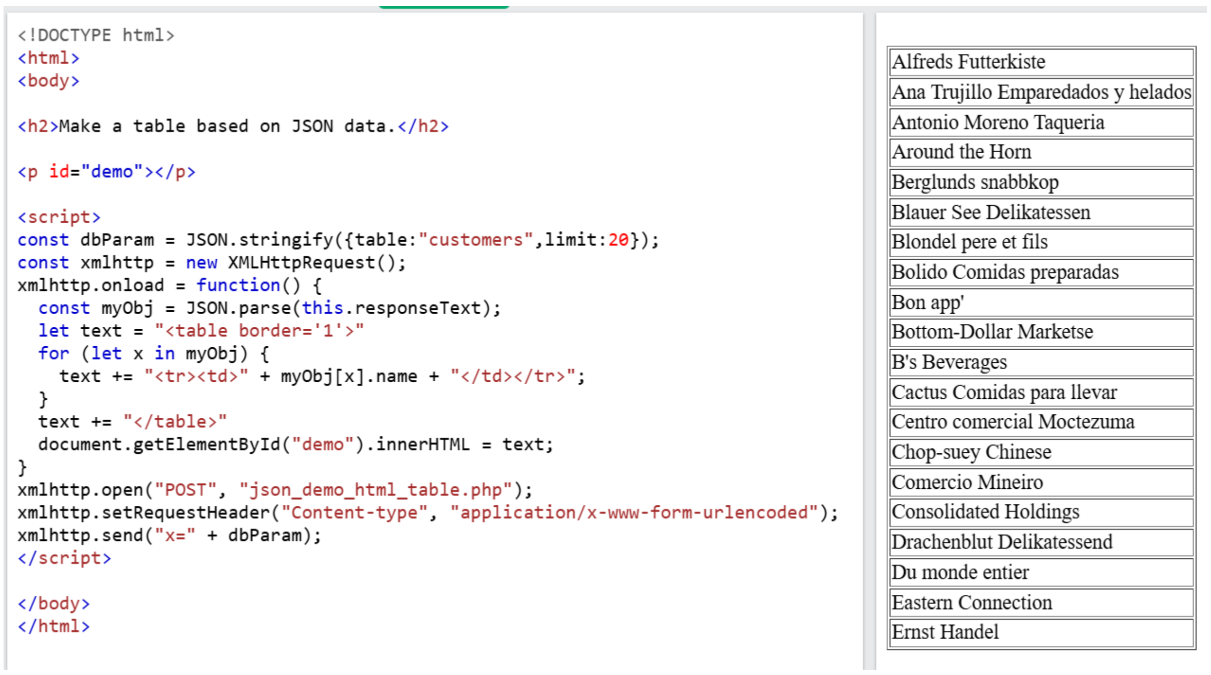
JavaScript can be used to make HTML in your web pages.

HTML Table

Make an HTML table with data received as JSON:

Example

```
const dbParam = JSON.stringify({table:"customers",limit:20});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  myObj = JSON.parse(this.responseText);
  let text = "<table border='1'>"
  for (let x in myObj) {
    text += "<tr><td>" + myObj[x].name + "</td></tr>";
  }
  text += "</table>"
  document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_html_table.php");
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```



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

<h2>Make a table based on JSON data.</h2>

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

<script>
const dbParam = JSON.stringify({table:"customers",limit:20});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  let text = "<table border='1'>"
  for (let x in myObj) {
    text += "<tr><td>" + myObj[x].name + "</td></tr>";
  }
  text += "</table>"
  document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_html_table.php");
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
</script>

</body>
</html>
```

Alfreds Futterkiste
Ana Trujillo Emparedados y helados
Antonio Moreno Taqueria
Around the Horn
Berglunds snabbkop
Blauer See Delikatessen
Blondel pere et fils
Bolido Comidas preparadas
Bon app'
Bottom-Dollar Marketse
B's Beverages
Cactus Comidas para llevar
Centro comercial Moctezuma
Chop-suey Chinese
Comercio Mineiro
Consolidated Holdings
Drachenblut Delikatessend
Du monde entier
Eastern Connection
Ernst Handel

Try it Yourself »

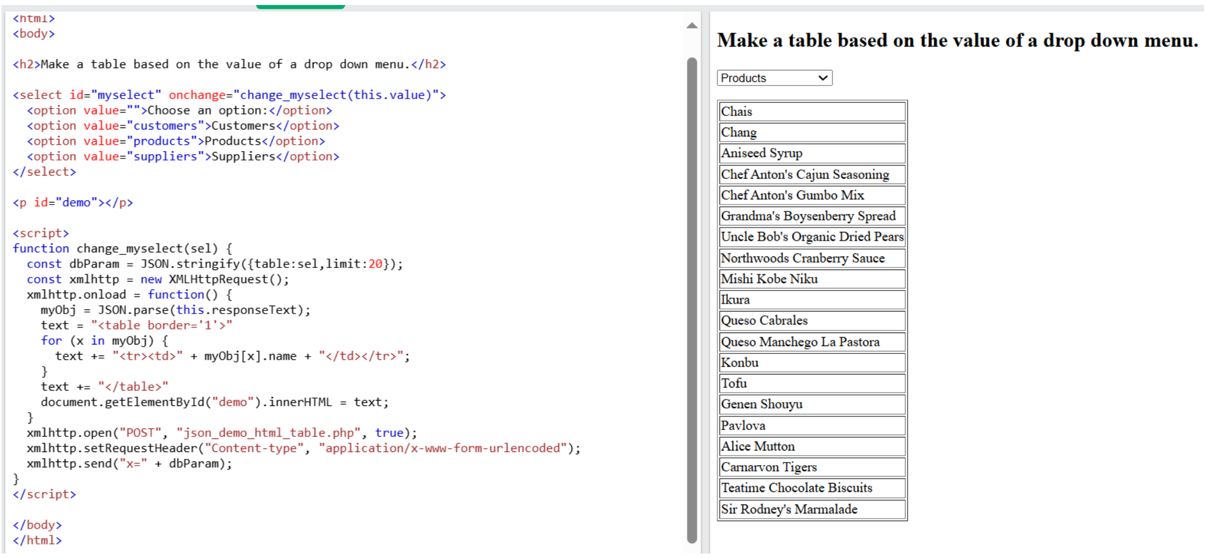
Dynamic HTML Table

Make the HTML table based on the value of a drop down menu: Choose an option: Customers Products Suppliers

Example

```
<select id="myselect" onchange="change_myselect(this.value)">
  <option value="">Choose an option:</option>
  <option value="customers">Customers</option>
  <option value="products">Products</option>
  <option value="suppliers">Suppliers</option>
</select>

<script>
function change_myselect(sel) {
  const dbParam = JSON.stringify({table:sel,limit:20});
  const xmlhttp = new XMLHttpRequest();
  xmlhttp.onload = function() {
    const myObj = JSON.parse(this.responseText);
    let text = "<table border='1'>"
    for (let x in myObj) {
      text += "<tr><td>" + myObj[x].name + "</td></tr>";
    }
    text += "</table>"
    document.getElementById("demo").innerHTML = text;
  }
  xmlhttp.open("POST", "json_demo_html_table.php");
```

```
 xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
 xmlhttp.send("x=" + dbParam);
}
</script>
```



```
<html>
<body>

<h2>Make a table based on the value of a drop down menu.</h2>

<select id="myselect" onchange="change_myselect(this.value)">
  <option value="">Choose an option:</option>
  <option value="customers">Customers</option>
  <option value="products">Products</option>
  <option value="suppliers">Suppliers</option>
</select>

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

<script>
function change_myselect(sel) {
  const dbParam = JSON.stringify({table:sel,limit:20});
  const xmlhttp = new XMLHttpRequest();
  xmlhttp.onload = function() {
    myObj = JSON.parse(this.responseText);
    text = "<table border='1'>"
    for (x in myObj) {
      text += "<tr><td>" + myObj[x].name + "</td></tr>";
    }
    text += "</table>"
    document.getElementById("demo").innerHTML = text;
  }
  xmlhttp.open("POST", "json_demo_html_table.php", true);
  xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xmlhttp.send("x=" + dbParam);
}
</script>

</body>
</html>
```

Make a table based on the value of a drop down menu.

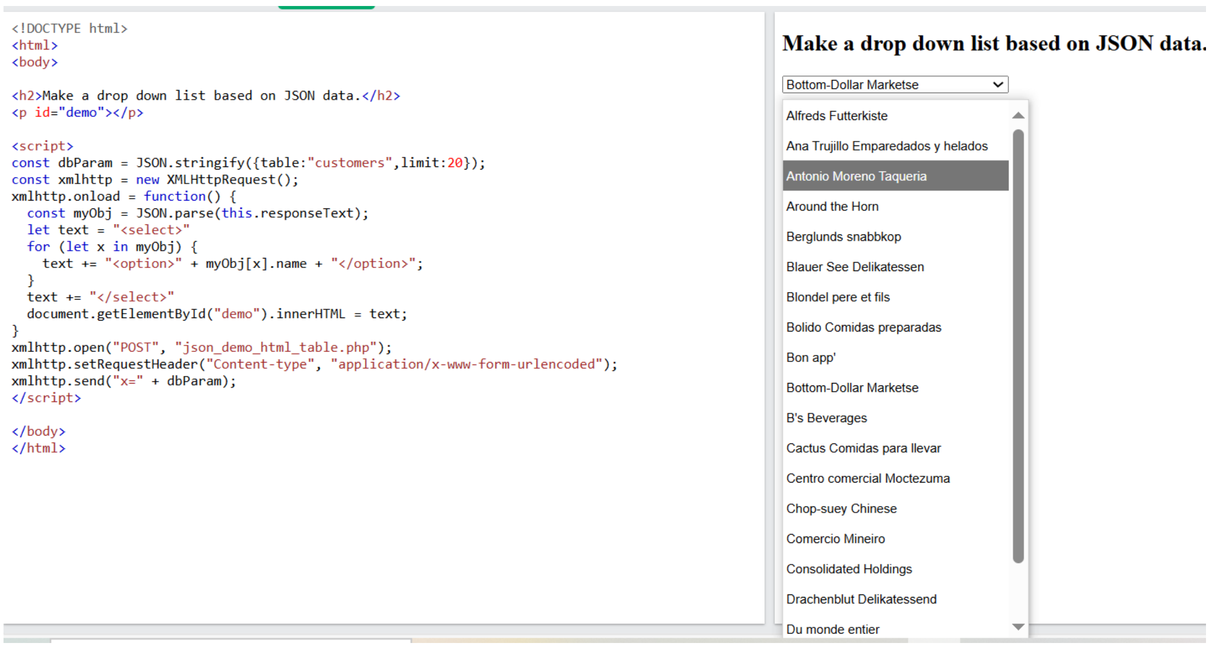
Chais
Chang
Aniseed Syrup
Chef Anton's Cajun Seasoning
Chef Anton's Gumbo Mix
Grandma's Boysenberry Spread
Uncle Bob's Organic Dried Pears
Northwoods Cranberry Sauce
Mishi Kobe Niku
Ikura
Queso Cabrales
Queso Manchego La Pastora
Konbu
Tofu
Genen Shouyu
Pavlova
Alice Mutton
Carnarvon Tigers
Teatime Chocolate Biscuits
Sir Rodney's Marmalade

HTML Drop Down List

Make an HTML drop down list with data received as JSON:

Example

```
const dbParam = JSON.stringify({table:"customers",limit:20});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
 const myObj = JSON.parse(this.responseText);
 let text = "<select>"
 for (let x in myObj) {
  text += "<option>" + myObj[x].name + "</option>";
 }
 text += "</select>"
 document.getElementById("demo").innerHTML = text;
 }
}
xmlhttp.open("POST", "json_demo_html_table.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```



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

<h2>Make a drop down list based on JSON data.</h2>
<p id="demo"></p>

<script>
const dbParam = JSON.stringify({table:"customers",limit:20});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
  const myObj = JSON.parse(this.responseText);
  let text = "<select>"
  for (let x in myObj) {
    text += "<option>" + myObj[x].name + "</option>";
  }
  text += "</select>"
  document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_html_table.php");
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
</script>

</body>
</html>
```

Make a drop down list based on JSON data.

```
Bottom-Dollar Marketse
Alfreds Futterkiste
Ana Trujillo Emparedados y helados
Antonio Moreno Taqueria
Around the Horn
Berglunds snabbkop
Blauer See Delikatessen
Blondel pere et fils
Bolido Comidas preparadas
Bon app'
Bottom-Dollar Marketse
B's Beverages
Cactus Comidas para llevar
Centro comercial Moctezuma
Chop-suey Chinese
Comercio Mineiro
Consolidated Holdings
Drachenblut Delikatessend
Du monde entier
```