JSON

JSON: JavaScript Object Notation.
JSON is a syntax for storing and exchanging data.
JSON is text, written with JavaScript object notation.

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.

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

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

Receiving Data

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

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

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

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

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

Why use JSON?

Since the JSON format is text only, it can easily be sent to and from a server, and used as a data format by any programming language.
JavaScript has a built in function to convert a string, written in JSON format, into native JavaScript objects:
JSON.parse()
So, if you receive data from a server, in JSON format, you can use it like any other JavaScript object.

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:
Example
"name":"John"
JSON names require double quotes. JavaScript names don't.

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 (JSON 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' }
```

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 object and assign data to it, like this:
Example
```
var person = { name: "John", age: 31, city: "New York" };
```
You can access a JavaScript object like this:
Example
```
// returns John
person.name;
```
It can also be accessed like this:
Example
```
// returns John
person["name"];
```
Data can be modified like this:
Example
```
person.name = "Gilbert";
```
It can also be modified like this:
Example
```
person["name"] = "Gilbert";
```
You will learn how to convert JavaScript objects into JSON later in this tutorial.

JavaScript Arrays as JSON

The same way JavaScript objects can be used as JSON, JavaScript arrays can also be used as JSON.
You will learn more about arrays as JSON later in this tutorial.

JSON Files

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

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

JSON 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 same rules as JSON object

---

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 }
```

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:

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

Make sure the text is written 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 + ", " + obj.age;
</script>
```

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.
Example
Use the XMLHttpRequest to get data from the server:

```
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();
```

Take a look at json_demo.txt

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

The JSON returned from the server is an array:

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
     var myArr = JSON.parse(this.responseText);
     document.getElementById("demo").innerHTML = myArr[0];
  }
};
xmlhttp.open("GET", "json_demo_array.txt", true);
xmlhttp.send();
```

Take a look at json_demo_array.txt

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:

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

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

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.

Example

Convert a string into a date, using the reviver function:

```
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;
```

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:

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

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

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:
var obj = { name: "John", age: 30, city: "New York" };
Use the JavaScript function JSON.stringify() to convert it into a string.
var 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
var obj = { name: "John", age: 30, city: "New York" };
var myJSON = JSON.stringify(obj);
document.getElementById("demo").innerHTML = myJSON;
You will learn how to send JSON to the server in the next chapter.

Stringify a JavaScript Array
It is also possible to stringify JavaScript arrays:
Imagine we have this array in JavaScript:
var arr = ["John", "Peter", "Sally", "Jane"];
Use the JavaScript function JSON.stringify() to convert it into a string.
var 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
var arr = ["John", "Peter", "Sally", "Jane"];
var myJSON = JSON.stringify(arr);
document.getElementById("demo").
innerHTML = myJSON;

JSON Objects

Object Syntax
Example
{ "name":"John", "age":30, "car":null }
JSON objects are surrounded by curly braces {}.
JSON objects are written in key/value pairs.
Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
Keys and values are separated by a colon.
Each key/value pair is separated by a comma.

Accessing Object Values

You can access the object values by using dot (.) notation:
Example
```
myObj = { "name":"John", "age":30, "car":null };
x = myObj.name;
```
You can also access the object values by using bracket ([]) notation:
Example
```
myObj = { "name":"John", "age":30, "car":null };
x = myObj["name"];
```

Looping an Object

You can loop through object properties by using the for-in loop:
Example
```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
  document.getElementById("demo").innerHTML += x;
}
```
In a for-in loop, use the bracket notation to access the property values:
Example
```
myObj = { "name":"John", "age":30, "car":null };
for (x in myObj) {
   document.getElementById("demo").innerHTML += myObj[x];
}
```

Nested JSON Objects

Values in a JSON object can be another JSON object.
Example
```
myObj = {
  "name":"John",
   "age":30,
   "cars": {
     "car1":"Ford",
     "car2":"BMW",
     "car3":"Fiat"
  }
 }
```
You can access nested JSON objects by using the dot notation or bracket notation:
Example
```
x = myObj.cars.car2;
// or:
x = myObj.cars["car2"];
```

Modify Values

You can use the dot notation to modify any value in a JSON object:

Example
```
myObj.cars.car2 = "Mercedes";
```
You can also use the bracket notation to modify a value in a JSON object:
Example
```
myObj.cars["car2"] = "Mercedes";
```

Delete Object Properties

Use the delete keyword to delete properties from a JSON object:
Example
```
delete myObj.cars.car2;
```

JSON Arrays

Arrays as JSON Objects
Example
```
[ "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.

Arrays in JSON Objects
Arrays can be values of an object property:
Example
```
{
"name":"John",
"age":30,
"cars":[ "Ford", "BMW", "Fiat" ]
}
```

Accessing Array Values
You access the array values by using the index number:
Example
```
x = myObj.cars[0];
```

Looping Through an Array
You can access array values by using a for-in loop:
Example
```
for (i in myObj.cars) {
  x += myObj.cars[i];
}
```
Or you can use a for loop:
Example
```
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:
Example
```
myObj = {
 "name":"John",
   "age":30,
   "cars": [
    { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
     { "name":"BMW", "models":[ "320", "X3", "X5" ] },
     { "name":"Fiat", "models":[ "500", "Panda" ] }
   ]
 }
```

To access arrays inside arrays, use a for-in loop for each array:

Example

```
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 Array Values

Use the index number to modify an array:

Example

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

Delete Array Items

Use the delete keyword to delete items from an array:

Example

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

JSON Arrays

Arrays as JSON Objects

Example

```
[ "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.

Arrays in JSON Objects

Arrays can be values of an object property:

Example

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

Accessing Array Values

You access the array values by using the index number:

Example

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

Looping Through an Array

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

Example

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

Or you can use a for loop:

Example

```
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:

Example

```
myObj = {
 "name":"John",
  "age":30,
  "cars": [
   { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
    { "name":"BMW", "models":[ "320", "X3", "X5" ] },
    { "name":"Fiat", "models":[ "500", "Panda" ] }
  ]
 }
```

To access arrays inside arrays, use a for-in loop for each array:

Example

```
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 Array Values

Use the index number to modify an array:
Example
```
 myObj.cars[1] = "Mercedes";
```

Delete Array Items

Use the delete keyword to delete items from an array:
Example
```
delete myObj.cars[1];
```

JSON PHP

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 how to exchange JSON data between the client and a PHP server.

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 file
```
<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";

$myJSON = json_encode($myObj);

echo $myJSON;
?>
```
The Client JavaScript
Here is a JavaScript on the client, using an AJAX call to request the PHP file from the example above:
Example
Use JSON.parse() to convert the result into a JavaScript object:
```
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", "demo_file.php", true);
xmlhttp.send();
```

PHP Array

Arrays in PHP will also be converted into JSON when using the PHP function json_encode():
PHP file

```
<?php
$myArr = array("John", "Mary", "Peter", "Sally");

$myJSON = json_encode($myArr);

echo $myJSON;
?>
```

The Client JavaScript
Here is a JavaScript on the client, using an AJAX call to request the PHP file from the array example above:
Example
Use JSON.parse() to convert the result into a JavaScript array:

```
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[2];
  }
};
xmlhttp.open("GET", "demo_file_array.php", true);
xmlhttp.send();
```

PHP Database

PHP is a server side programming language, and can be used to access a database.
Imagine you have a database on your server, and you want to send a request to it from the client where you ask for the 10 first rows in a table called "customers".
On the client, make a JSON object that describes the numbers of rows you want to return.
Before you send the request to the server, convert the JSON object into a string and send it as a parameter to the url of the PHP page:
Example
Use JSON.stringify() to convert the JavaScript object into JSON:

```
obj = { "limit":10 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML = this.responseText;
  }
};
xmlhttp.open("GET", "json_demo_db.php?x=" + dbParam, true);
xmlhttp.send();
```

Example explained:
Define an object containing a "limit" property and value.
Convert the object into a JSON string.
Send a request to the PHP file, with the JSON string as a parameter.
Wait until the request returns with the result (as JSON)

Display the result received from the PHP file.
Take a look at the PHP file:
PHP file

```
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_GET["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");
$stmt = $conn->prepare("SELECT name FROM customers LIMIT ?");
$stmt->bind_param("s", $obj->limit);
$stmt->execute();
$result = $stmt->get_result();
$outp = $result->fetch_all(MYSQLI_ASSOC);

echo json_encode($outp);
?>
```

PHP File explained:

Convert the request into an object, using the PHP function json_decode().
Access the database, and fill an array with the requested data.
Add the array to an object, and return the object as JSON using the json_encode() function.
Loop Through the Result
Convert the result received from the PHP file into a JavaScript object, or in this case, a JavaScript array:
Example
Use JSON.parse() to convert the JSON into a JavaScript object:

```
...
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
     myObj = JSON.parse(this.responseText);
     for (x in myObj) {
       txt += myObj[x].name + "<br>";
    }
     document.getElementById("demo").innerHTML = txt;
  }
};
...
```

PHP Method = POST

When sending data to the server, it is often best to use the HTTP POST method.
To send AJAX requests using the POST method, specify the method, and the correct header.
The data sent to the server must now be an argument to the send() method:
Example

```
obj = { "limit":10 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
     myObj = JSON.parse(this.responseText);
     for (x in myObj) {
      txt += myObj[x].name + "<br>";
```

```
    }
     document.getElementById("demo").innerHTML = txt;
  }
};
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```

The only difference in the PHP file is the method for getting the transferred data.

PHP file

Use $_POST instead of $_GET:

```
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_POST["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");
$stmt = $conn->prepare("SELECT name FROM customers LIMIT ?");
$stmt->bind_param("s", $obj->limit);
$stmt->execute();
$result = $stmt->get_result();
$outp = $result->fetch_all(MYSQLI_ASSOC);

echo json_encode($outp);
?>
```

JSON HTML

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

```
obj = { table: "customers", limit: 20 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
     myObj = JSON.parse(this.responseText);
     txt += "<table border='1'>"
    for (x in myObj) {
      txt += "<tr><td>" + myObj[x].name + "</td></tr>";
     }
    txt += "</table>"
     document.getElementById("demo").innerHTML = txt;
```

```
   }
}
xmlhttp.open("POST", "json_demo_db_post.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```

Dynamic HTML Table

Make the HTML table based on the value of a drop down menu:

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) {
  var obj, dbParam, xmlhttp, myObj, x, txt = "";
  obj = { table: sel, limit: 20 };
  dbParam = JSON.stringify(obj);
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
      myObj = JSON.parse(this.responseText);
       txt += "<table border='1'>"
       for (x in myObj) {
        txt += "<tr><td>" + myObj[x].name + "</td></tr>";
      }
       txt += "</table>"
      document.getElementById("demo").innerHTML = txt;
    }
  };
  xmlhttp.open("POST", "json_demo_html_table.php", true);
  xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xmlhttp.send("x=" + dbParam);
}
</script>
```

HTML Drop Down List

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

Example

```
obj = { table: "customers", limit: 20 };
dbParam = JSON.stringify(obj);
xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
   if (this.readyState == 4 && this.status == 200) {
    myObj = JSON.parse(this.responseText);
     txt += "<select>"
    for (x in myObj) {
      txt += "<option>" + myObj[x].name;
     }
    txt += "</select>"
     document.getElementById("demo").innerHTML = txt;
   }
}
xmlhttp.open("POST", "json_demo_html_table.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
```

JSONP

JSONP is a method for sending JSON data without worrying about cross-domain issues.
JSONP does not use the XMLHttpRequest object.
JSONP uses the <script> tag instead.

JSONP Intro

JSONP stands for JSON with Padding.
Requesting a file from another domain can cause problems, due to cross-domain policy.
Requesting an external script from another domain does not have this problem.
JSONP uses this advantage, and request files using the script tag instead of the XMLHttpRequest object.

```
<script src="demo_jsonp.php">
```

The Server File
The file on the server wraps the result inside a function call:
Example

```
<?php
$myJSON = '{ "name":"John", "age":30, "city":"New York" }';

echo "myFunc(".$myJSON.");";
?>
```

The result returns a call to a function named "myFunc" with the JSON data as a parameter.
Make sure that the function exists on the client.
The JavaScript function
The function named "myFunc" is located on the client, and ready to handle JSON data:
Example

```
function myFunc(myObj) {
  document.getElementById("demo").innerHTML = myObj.name;
}
```

Creating a Dynamic Script Tag

The example above will execute the "myFunc" function when the page is loading, based on where you put the script tag, which is not very satisfying.
The script tag should only be created when needed:
Example
Create and insert the <script> tag when a button is clicked:

```
function clickButton() {
  var s = document.createElement("script");
  s.src = "demo_jsonp.php";
  document.body.appendChild(s);
}
```

Dynamic JSONP Result

The examples above are still very static.
Make the example dynamic by sending JSON to the php file, and let the php file return a JSON object based on the information it gets.

PHP file

```
<?php
header("Content-Type: application/json; charset=UTF-8");
$obj = json_decode($_GET["x"], false);

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");
$result = $conn->query("SELECT name FROM ".$obj->$table." LIMIT ".$obj->$limit);
$outp = array();
$outp = $result->fetch_all(MYSQLI_ASSOC);

echo "myFunc(".json_encode($outp).")";
?>
```

PHP File explained:

Convert the request into an object, using the PHP function json_decode().
Access the database, and fill an array with the requested data.
Add the array to an object.
Convert the array into JSON using the json_encode() function.
Wrap "myFunc()" around the return object.
JavaScript Example
The "myFunc" function will be called from the php file:

```
function clickButton() {
  var obj, s
  obj = { table: "products", limit: 10 };
   s = document.createElement("script");
   s.src = "jsonp_demo_db.php?x=" + JSON.stringify(obj);
   document.body.appendChild(s);
}
function myFunc(myObj) {
   var x, txt = "";
   for (x in myObj) {
     txt += myObj[x].name + "<br>";
   }
   document.getElementById("demo").innerHTML = txt;
}
```

Callback Function

When you have no control over the server file, how do you get the server file to call the correct function?
Sometimes the server file offers a callback function as a parameter:
Example
The php file will call the function you pass as a callback parameter:

```
function clickButton() {
  var s = document.createElement("script");
  s.src = "jsonp_demo_db.php?callback=myDisplayFunction";
  document.body.appendChild(s);
}
```