# 09. JSON PHP.

## 9.1. 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.

## 9.2. 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():

**Example**: PHP file.

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

$myJSON = json_encode($myObj);

echo $myJSON;
?>
```

## 9.3. 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();
```

## 9.4. PHP Array

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

**Example**: PHP file

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

$myJSON = json_encode($myArr);

echo $myJSON;
?>
```

## 9.5. 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[2];
    }
};
xmlhttp.open("GET", "demo_file_array.php", true);
xmlhttp.send();
```

# 9.6. PHP Database

PHP is a server side programming language, and should be used for operations that can only be performed by a server, like accessing a database.

Imagine you have a database on the server, containing customers, products, and suppliers.

You want to make a request to the server where you ask for the first 10 records in the "customers" table:

**Example**: Use JSON.stringify() to convert the JavaScript object into JSON:

```
obj = { "table":"customers", "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 table property and a limit property.
- 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:

**Example**: 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 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.

## 9.7. 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:

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

<h2>Get data as JSON from a PHP file on the server.</h2>
<p id="demo"></p>

<script>
var obj, dbParam, xmlhttp, myObj, x, txt = "";
obj = { "table":"customers", "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("GET", "json_demo_db.php?x=" + dbParam, true);
xmlhttp.send();

</script>

<p>Try changing the "table" property from "customers" to
"products".</p>

</body>
</html>
```

## 9.8. 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 = { "table":"customers", "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.

**Example**: 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");
$result = $conn->query("SELECT name FROM ".$obj->table." LIMIT ".
$obj->limit);
$outp = array();
$outp = $result->fetch_all(MYSQLI_ASSOC);

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