PHP JSON Parsing

In this tutorial you will learn how to encode and decode JSON data in PHP.

What is JSON

JSON stands for **J**ava**S**cript **O**bject **N**otation. JSON is a standard lightweight data-interchange format which is quick and easy to parse and generate.

JSON, like XML, is a text-based format that's easy to write and easy to understand for both humans and computers, but unlike XML, JSON data structures occupy less bandwidth than their XML versions. JSON is based on two basic structures:

- **Object:** This is defined as a collection of key/value pairs (i.e. `key:value`). Each object begins with a left curly bracket `{` and ends with a right curly bracket `}`. Multiple key/value pairs are separated by a comma `,`.
- **Array:** This is defined as an ordered list of values. An array begins with a left bracket `[` and ends with a right bracket `]`. Values are separated by a comma `,`.

In JSON, keys are always strings, while the value can be a **`string`**, **`number`**, **`true`** or **`false`**, **`null`** or even an **`object`** or an **`array`**. Strings must be enclosed in double quotes `"` and can contain escape characters such as `\n`, `\t` and `\`. A JSON object may look like this:

Example — Run this code »

```
{
    "book": {
        "name": "Harry Potter and the Goblet of Fire",
        "author": "J. K. Rowling",
        "year": 2000,
        "genre": "Fantasy Fiction",
        "bestseller": true
    }
}
```

Whereas an example of JSON array would look something like this:

Example — Run this code »

```
{
    "fruits": [
        "Apple",
        "Banana",
        "Strawberry",
        "Mango"
    ]
}
```

Tip: A data-interchange format is a text format which is used to interchange or exchange data between different platforms and operating systems. JSON is the most popular and lightweight data-interchange format for web applications.

Parsing JSON with PHP

JSON data structures are very similar to PHP arrays. PHP has built-in functions to encode and decode JSON data. These functions are `json_encode()` and `json_decode()`, respectively. Both functions only works with UTF-8 encoded string data.

Encoding JSON Data in PHP

In PHP the `json_encode()` function is used to encode a value to JSON format. The value being encoded can be any PHP data type except a resource, like a database or file handle. The below example demonstrates how to encode a PHP associative array into a JSON object:

Example **Run this code »**

```
<?php
// An associative array
$marks = array("Peter"=>65, "Harry"=>80, "John"=>78, "Clark"=>90);

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

The output of the above example will look like this:

```
{"Peter":65,"Harry":80,"John":78,"Clark":90}
```

Similarly, you can encode the PHP indexed array into a JSON array, like this:

Example **Run this code »**

```
<?php
// An indexed array
$colors = array("Red", "Green", "Blue", "Orange", "Yellow");

```

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

The output of the above example will look like this:

```
["Red","Green","Blue","Orange","Yellow"]
```

You can also force `json_encode()` function to return an PHP indexed array as JSON object by using the `JSON_FORCE_OBJECT` option, as shown in the example below:

Example **Run this code »**

```php
<?php
// An indexed array
$colors = array("Red", "Green", "Blue", "Orange");

echo json_encode($colors, JSON_FORCE_OBJECT);
?>
```

The output of the above example will look like this:

```
{"0":"Red","1":"Green","2":"Blue","3":"Orange"}
```

As you can see in the above examples a non-associative array can be encoded as array or object. However, an associative array always encoded as object.

Decoding JSON Data in PHP

Decoding JSON data is as simple as encoding it. You can use the PHP `json_decode()` function to convert the JSON encoded string into appropriate PHP data type. The following example demonstrates how to decode or convert a JSON object to PHP object.

Example **Run this code »**

```php
<?php
// Store JSON data in a PHP variable
$json = '{"Peter":65,"Harry":80,"John":78,"Clark":90}';

var_dump(json_decode($json));
?>
```

The output of the above example will look something like this:

```
object(stdClass)#1 (4) { ["Peter"]=> int(65) ["Harry"]=> int(80) ["John"]=>
int(78) ["Clark"]=> int(90) }
```

By default the `json_decode()` function returns an object. However, you can optionally specify a second parameter **`$assoc`** which accepts a boolean value that when set as `true` JSON objects are decoded into associative arrays. It is `false` by default. Here's an example:

Example **Run this code »**

```php
<?php
// Store JSON data in a PHP variable
$json = '{"Peter":65,"Harry":80,"John":78,"Clark":90}';

var_dump(json_decode($json, true));
?>
```

The output of the above example will look something like this:

```
array(4) { ["Peter"]=> int(65) ["Harry"]=> int(80) ["John"]=> int(78)
["Clark"]=> int(90) }
```

Now let's check out an example that will show you how to decode the JSON data and access individual elements of the JSON object or array in PHP.

Example **Run this code »**

```php
<?php
// Assign JSON encoded string to a PHP variable
$json = '{"Peter":65,"Harry":80,"John":78,"Clark":90}';

// Decode JSON data to PHP associative array
$arr = json_decode($json, true);
// Access values from the associative array
echo $arr["Peter"];  // Output: 65
echo $arr["Harry"];  // Output: 80
echo $arr["John"];   // Output: 78
echo $arr["Clark"];  // Output: 90

// Decode JSON data to PHP object
$obj = json_decode($json);
// Access values from the returned object
echo $obj->Peter;   // Output: 65
echo $obj->Harry;   // Output: 80
echo $obj->John;    // Output: 78
echo $obj->Clark;   // Output: 90
?>
```

You can also loop through the decoded data using `foreach()` loop, like this:

Example

Run this code »

```php
<?php
// Assign JSON encoded string to a PHP variable
$json = '{"Peter":65,"Harry":80,"John":78,"Clark":90}';

// Decode JSON data to PHP associative array
$arr = json_decode($json, true);

// Loop through the associative array
foreach($arr as $key=>$value){
    echo $key . "=>" . $value . "<br>";
}
echo "<hr>";
// Decode JSON data to PHP object
$obj = json_decode($json);

// Loop through the object
foreach($obj as $key=>$value){
    echo $key . "=>" . $value . "<br>";
}
?>
```

Extracting Values from Nested JSON Data in PHP

JSON objects and arrays can also be nested. A JSON object can arbitrarily contains other JSON objects, arrays, nested arrays, arrays of JSON objects, and so on. The following example will show you how to decode a nested JSON object and print all its values in PHP.

Example

Run this code »

```php
<?php
// Define recursive function to extract nested values
function printValues($arr) {
    global $count;
    global $values;

    // Check input is an array
    if(!is_array($arr)){
        die("ERROR: Input is not an array");
    }
    /*
    Loop through array, if value is itself an array recursively call
the
```