* **json\_encode() Function**

PHP json\_encode() function converts a PHP value into a JSON value. For example, from a PHP array, it can create a JSON representation of that array. json\_encode() function returns a string, if the function works.

**Syntax:**

json\_encode(value, options)

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Type** | **Description** |
| Value | Mixed | Any PHP type except resource. Must be UTF character encoded data. |
| Options | Integer | Bitmask comprising of JSON\_HEX\_QUOT, JSON\_HEX\_TAG, JSON\_HEX\_AMP, JSON\_HEX\_APOS, JSON\_NUMERIC\_CHECK, JSON\_PRETTY\_PRINT, JSON\_UNESCAPED\_SLASHES, JSON\_FORCE\_OBJECT. |

* **json\_decode**

It Decodes a JSON string. Takes a JSON encoded string and converts it into a PHP variable. This function only works with UTF-8 encoded strings. Returns the value encoded in **json** in appropriate PHP type. Values true,  false and null are returned **TRUE**, **FALSE** and **NULL** respectively. **NULL** is returned if the **json** cannot be decoded or if the encoded data is deeper than the recursion limit.

PHP's json\_decode function takes a JSON string and converts it into a PHP variable. Typically, the JSON data will represent a JavaScript array or object literal which json\_decode will convert into a PHP array or object.

**json\_decode** ( string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0]]] )

Where

**Json:** The **json** [string](http://php.net/manual/en/language.types.string.php) being decoded.

**Assoc:** When **TRUE**, returned [object](http://php.net/manual/en/language.types.object.php)s will be converted into associative [array](http://php.net/manual/en/language.types.array.php)s.

**Depth:**User specified recursion depth.

**Options:** Bitmask of JSON decode options. Currently there are two supported options. The first is **JSON\_BIGINT\_AS\_STRING** that allows casting big integers to string instead of floats which is the default. The second option is **JSON\_OBJECT\_AS\_ARRAY** that has the same effect as setting **assoc** to **TRUE**.

The following two examples demonstrate, first with an array, then with an object:

$json = '["apple","orange","banana","strawberry"]';

$ar = json\_decode($json);

// access first element of $ar array

echo $ar[0]; // apple

By default, objects are converted to standard objects by json\_decode:

$json = '{

"title": "JavaScript: The Definitive Guide",

"author": "David Flanagan",

"edition": 6

}';

$book = json\_decode($json);

// access title of $book object

echo $book->title; // JavaScript: The Definitive Guide

**Convert JSON String to PHP Array**

The json\_decode function provides an optional second argument to convert objects to associative arrays. The following uses the object from the previous example and passes true as the second argument. The result, held in $book, is now an array, and its title and other elements can be accessed using array syntax:

// $json same as example object above

// pass true to convert objects to associative arrays

$book = json\_decode($json, true);

// access title of $book array

echo $book['title']; // JavaScript: The Definitive Guide

**JSON String to Multidimensional Array**

The json\_decode function can also be used to convert more complex data structures held in JSON strings. The JSON string ($json) in the following example represents an array of objects. That is, the outer level is an array literal whose elements are object literals. By default the result of json\_decode will be a numerically indexed array of objects:

$json = '[

{

"title": "Professional JavaScript",

"author": "Nicholas C. Zakas"

},

{

"title": "JavaScript: The Definitive Guide",

"author": "David Flanagan"

},

{

"title": "High Performance JavaScript",

"author": "Nicholas C. Zakas"

}

]';

$books = json\_decode($json);

// access property of object in array

echo $books[1]->title; // JavaScript: The Definitive Guide

If we pass true as the second argument to json\_decode, the result is a multidimensional array that is numerically indexed at the outer level and associative at the inner level:

// $json same as example object above

// pass true to convert objects to associative arrays

$books = json\_decode($json, true);

// numeric/associative array access

echo $books[1]['title']; // JavaScript: The Definitive Guide

Then we use a combination of numeric and associative array syntax to access the desired element in our multidimensional array.

**Errors or Unexpected Results with json\_decode?**

The json\_decode function will return null if the string passed to it cannot be converted. For example, the following demonstrates the result when our JSON string contains invalid characters:

$json = "{

'title': 'JavaScript: The Definitive Guide',

'author': 'David Flanagan',

'edition': 6

}";

$book = json\_decode($json);

echo $book->title; // Notice: Trying to get property of non-object...

echo json\_last\_error(); // 4 (JSON\_ERROR\_SYNTAX)

echo json\_last\_error\_msg(); // unexpected character

Our attempt to access a property of the object we are expecting results in an error. The PHP functions json\_last\_error and json\_last\_error\_msg provide some information about errors encountered.