PHP 7.1.0 Released

Keyboard Shortcuts

?
: This help

j
: Next menu item

k
: Previous menu item

g p
: Previous man page

g n
: Next man page

G
: Scroll to bottom

g g
: Scroll to top

json_decode

(PHP 5 >= 5.2.0, PECL json >= 1.2.0, PHP 7)

json_decode — Decodes a JSON string

Description ¶

mixed **json_decode** (string `$json` [, bool `$assoc` = false [, int `$depth` = 512 [, int `$options` = 0]]])

Takes a JSON encoded string and converts it into a PHP variable.

Parameters ¶

`json`

The `json` string being decoded.

This function only works with UTF-8 encoded strings.

Note:

PHP implements a superset of JSON as specified in the original » RFC 7159.

`assoc`

When **`TRUE`**, returned objects will be converted into associative arrays.

`depth`

User specified recursion depth.

`options`

Bitmask of JSON decode options. Currently only **JSON_BIGINT_AS_STRING** is supported (default is to cast large integers as floats)

Return Values ¶

Returns the value encoded in `json` in appropriate PHP type. Values *true*, *false* and *null* are returned as **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.

Examples ¶

Example #1 json_decode() examples

```php
<?php
$json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

var_dump(json_decode($json));
var_dump(json_decode($json, true));

?>
```

The above example will output:

```
object(stdClass)#1 (5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}

array(5) {
    ["a"] => int(1)
    ["b"] => int(2)
    ["c"] => int(3)
    ["d"] => int(4)
    ["e"] => int(5)
}
```

Example #2 Accessing invalid object properties

Accessing elements within an object that contain characters not permitted under PHP's naming convention (e.g. the hyphen) can be accomplished by encapsulating the element name within braces and the apostrophe.

```php
<?php

$json = '{"foo-bar": 12345}';

$obj = json_decode($json);
print $obj->{'foo-bar'}; // 12345

?>
```

Example #3 common mistakes using json_decode()

```php
<?php
```

```php
// the following strings are valid JavaScript but not valid JSON

// the name and value must be enclosed in double quotes
// single quotes are not valid
$bad_json = "{ 'bar': 'baz' }";
json_decode($bad_json); // null

// the name must be enclosed in double quotes
$bad_json = '{ bar: "baz" }';
json_decode($bad_json); // null

// trailing commas are not allowed
$bad_json = '{ bar: "baz", }';
json_decode($bad_json); // null

?>
```

Example #4 `depth` errors

```php
<?php
// Encode the data.
$json = json_encode(
    array(
        1 => array(
            'English' => array(
                'One',
                'January'
            ),
            'French' => array(
                'Une',
                'Janvier'
            )
        )
    )
);

// Define the errors.
$constants = get_defined_constants(true);
$json_errors = array();
foreach ($constants["json"] as $name => $value) {
    if (!strncmp($name, "JSON_ERROR_", 11)) {
        $json_errors[$value] = $name;
    }
}

// Show the errors for different depths.
foreach (range(4, 3, -1) as $depth) {
    var_dump(json_decode($json, true, $depth));
    echo 'Last error: ', $json_errors[json_last_error()], PHP_EOL, PHP_EOL;
}
?>
```

The above example will output:

```
array(1) {
  [1]=>
  array(2) {
    ["English"]=>
    array(2) {
      [0]=>
      string(3) "One"
      [1]=>
      string(7) "January"
    }
    ["French"]=>
    array(2) {
      [0]=>
      string(3) "Une"
      [1]=>
      string(7) "Janvier"
    }
  }
}
Last error: JSON_ERROR_NONE

NULL
Last error: JSON_ERROR_DEPTH
```

Example #5 json_decode() of large integers

```php
<?php
$json = '{"number": 12345678901234567890}';

var_dump(json_decode($json));
var_dump(json_decode($json, false, 512, JSON_BIGINT_AS_STRING));

?>
```

The above example will output:

```
object(stdClass)#1 (1) {
  ["number"]=>
  float(1.2345678901235E+19)
}
object(stdClass)#1 (1) {
  ["number"]=>
  string(20) "12345678901234567890"
}
```

Notes ¶

> **Note**:
>
> The JSON spec is not JavaScript, but a subset of JavaScript.
>
> **Note**:
>
> In the event of a failure to decode, json_last_error() can be used to determine the exact nature of the error.

Changelog ¶

Version	Description

5.6.0	Invalid non-lowercased variants of the *true*, *false* and *null* literals are no longer accepted as valid input, and will generate warnings.
5.4.0	The `options` parameter was added.
5.3.0	Added the optional `depth`. The default recursion depth was increased from 128 to 512
5.2.3	The nesting limit was increased from 20 to 128
5.2.1	Added support for JSON decoding of basic types.

See Also ¶

- json_encode() - Returns the JSON representation of a value
- json_last_error() - Returns the last error occurred

add a note

User Contributed Notes 30 notes

up
down
112
Aaron Kardell ¶
8 years ago

Make sure you pass in utf8 content, or json_decode may error out and just return a null value. For a particular web service I was using, I had to do the following:

```
<?php
$contents = file_get_contents($url);
$contents = utf8_encode($contents);
$results = json_decode($contents);
?>
```

Hope this helps!

up
down
66
jrevillini ¶
8 years ago

When decoding strings from the database, make sure the input was encoded with the correct charset when it was input to the database.

I was using a form to create records in the DB which had a content field that was valid JSON, but it included curly apostrophes. If the page with the form did not have

```
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

in the head, then the data was sent to the database with the wrong encoding. Then, when json_decode tried to convert the string to an object, it failed every time.

up
down
35
Matt Browne ¶
3 years ago

The function by 1franck to allow comments works except if there is a comment at the very beginning of the file. Here's a modified version (only the regex was changed) that accounts for that.

```
<?php
function json_clean_decode($json, $assoc = false, $depth = 512, $options = 0) {
    // search and remove comments like /* */ and //
    $json = preg_replace("#(/\*([^*]|[\r\n]|(\*+([^*/]|[\r\n])))*\*+/)|([\s\t]//.*)|(^//.*)#", '',
$json);

    if(version_compare(phpversion(), '5.4.0', '>=')) {
        $json = json_decode($json, $assoc, $depth, $options);
    }
    elseif(version_compare(phpversion(), '5.3.0', '>=')) {
        $json = json_decode($json, $assoc, $depth);
    }
    else {
        $json = json_decode($json, $assoc);
    }

    return $json;
}
?>
```

up
down
40
skyppher at gmail dot com ¶
3 years ago

First of all, since JSON is not native PHP format or even native JavaScript format, everyone who wants to use JSON wisely should carefuly read official documentation. There is a link to it here, in "Introduction" section. Many questions like "it doesn't recognize my strings" and those like previous one (about zip codes) will drop if you will be attentive!

And second. I've found that there is no good, real working example of how to validate string if it is a JSON or not.
There are two ways to make this: parse input string for yourself using regular expressions or anything else, use json_decode to do it for you.
Parsing for yourself is like writing your own compiler, too difficult.
Just testing result of json_decode is not enough because you should test it with NULL, but valid JSON could be like this 'null' and it will evaluate to NULL. So you should use another function - json_last_error. It will return error code of the last encode/decode operation. If no error occured it will be JSON_ERROR_NONE. So here is the function you should use for testing:

```
<?php
function isValidJson($strJson) {
    json_decode($strJson);
    return (json_last_error() === JSON_ERROR_NONE);
}
?>
```

It's so simple, that there is no need to use it and slow down your script with extra delay for function call. Just do it manualy in you code while working with input data:

```
<?php
//here is my initial string
$sJson = $_POST['json'];
//try to decode it
$json = json_decode($sJson);
```

```
if (json_last_error() === JSON_ERROR_NONE) {
    //do something with $json. It's ready to use
} else {
    //yep, it's not JSON. Log error or alert someone or do nothing
}
?>
```

up
down
33

1franck ¶

3 years ago

Sometime, i need to allow comments in json file. So i wrote a small func to clean comments in a json string before decoding it:

```
<?php
/**
* Clean comments of json content and decode it with json_decode().
* Work like the original php json_decode() function with the same params
*
* @param   string  $json    The json string being decoded
* @param   bool    $assoc   When TRUE, returned objects will be converted into associative arrays.
* @param   integer $depth   User specified recursion depth. (>=5.3)
* @param   integer $options Bitmask of JSON decode options. (>=5.4)
* @return  string
*/
function json_clean_decode($json, $assoc = false, $depth = 512, $options = 0) {

    // search and remove comments like /* */ and //
    $json = preg_replace("#(/\*([^*]|[\r\n]|(\*+([^*/]|[\r\n])))*\*+/)|([\s\t](//).*)#", '', $json);

    if(version_compare(phpversion(), '5.4.0', '>=')) {
        $json = json_decode($json, $assoc, $depth, $options);
    }
    elseif(version_compare(phpversion(), '5.3.0', '>=')) {
        $json = json_decode($json, $assoc, $depth);
    }
    else {
        $json = json_decode($json, $assoc);
    }

    return $json;
}
?>
```

up
down
32

T erkif ¶

6 years ago

it seems, that some of the people are not aware, that if you are using json_decode to decode a string it HAS to be a propper json string:

```
<?php
var_dump(json_encode('Hello'));

var_dump(json_decode('Hello'));  // wrong
```

```
var_dump(json_decode("Hello")); // wrong
var_dump(json_decode('"Hello"')); // correct
var_dump(json_decode("'Hello'")); // wrong

result:

string(7) ""Hello""
NULL
NULL
string(5) "Hello"
NULL
```

up
down
28
php at hm2k.org ¶
6 years ago

```
If var_dump produces NULL, you may be experiencing JSONP aka JSON with padding, here's a quick fix...

<?php

//remove padding
$body=preg_replace('/.+?({.+}).+/','$1',$body);

// now, process the JSON string
$result = json_decode($body);

var_dump($result);
?>
```

up
down
26
majca J ¶
6 years ago

```
Noted in a comment below is that this function will return NULL when given a simple string.

This is new behavior - see the result in PHP 5.2.4 :
php > var_dump(json_decode('this is a simple string'));
string(23) "this is a simple string"

in PHP 5.3.2 :
php > var_dump(json_decode('this is a simple string'));
NULL

I had several functions that relied on checking the value of a purported JSON string if it didn't
decode into an object/array. If you do too, be sure to be aware of this when upgrading to PHP 5.3.
```

up
down
18
php_net_01_weber at nachvorne de ¶
3 years ago

```
json_decode_nice + keep linebreaks:

function json_decode_nice($json, $assoc = TRUE){
    $json = str_replace(array("\n","\r"),"\\n",$json);
```

```
    $json = preg_replace('/([{,]+)(\s*)([^"]+?)\s*:/','$1"$3":',$json);
    $json = preg_replace('/(,)\s*}$/','}',$json);
    return json_decode($json,$assoc);
}

by phpdoc at badassawesome dot com, I just changed line 2.
If you want to keep the linebreaks just escape the slash.
```

up
down
13
phpdoc at badassawesome dot com ¶
3 years ago

I added a 3rd regex to the json_decode_nice function by "colin.mollenhour.com" to handle a trailing comma in json definition.

```
<?php
// http://www.php.net/manual/en/function.json-decode.php#95782
function json_decode_nice($json, $assoc = FALSE){
    $json = str_replace(array("\n","\r"),"",$json);
    $json = preg_replace('/([{,]+)(\s*)([^"]+?)\s*:/','$1"$3":',$json);
    $json = preg_replace('/(,)\s*}$/','}',$json);
    return json_decode($json,$assoc);
}
?>

Example:

<?php
$dat_json = <<<EOF
{
    "foo"    : "bam",
    "bar"    : "baz",
}
EOF;
$dat_array = json_decode_nice( $dat_json );
var_dump ( $dat_json, $dat_array );

/* RESULTS:

string(35) "{
    "foo"    : "bam",
    "bar"    : "baz",
}"
array(2) {
  ["foo"]=>
  string(3) "bam"
  ["bar"]=>
  string(3) "baz"
}
*/
?>
```

up
down
18

premiersullivan at gmail dot com ¶
7 years ago
This function will remove trailing commas and encode in utf8, which might solve many people's problems. Someone might want to expand it to also change single quotes to double quotes, and fix other kinds of json breakage.

```
<?php
    function mjson_decode($json)
    {
        return json_decode(removeTrailingCommas(utf8_encode($json)));
    }

    function removeTrailingCommas($json)
    {
        $json=preg_replace('/,\s*([\]}])/m', '$1', $json);
        return $json;
    }
?>
```

up
down
15
nix ¶
6 years ago
Be aware, when decoding JSON strings, where an empty string is a key, this library replaces the empty string with "_empty_".

So the following code gives an unexpected result:

```
<?php
var_dump(json_decode('{"":"arbitrary"}'));
?>
```

The result is as follows:

```
object(stdClass)#1 (1) {
  ["_empty_"]=>
  string(6) "arbitrary"
}
```

Any subsequent key named "_empty_" (or "" [the empty string] again) will overwrite the value.
up
down
15
Gravis ¶
7 years ago
with two lines you can convert your string from JavaScript toSource() (see http://www.w3schools.com/jsref/jsref_toSource.asp) output format to JSON accepted format. this works with subobjects too!
note: toSource() is part of JavaScript 1.3 but only implemented in Mozilla based javascript engines (not Opera/IE/Safari/Chrome).

```
<?php
  $str = '({strvar:"string", number:40, boolvar:true, subobject:{substrvar:"sub string", subsubobj:
{deep:"deeply nested"}, strnum:"56"}, false_val:false, false_str:"false"})'; // example javascript
object toSource() output
```

```
  $str = substr($str, 1, strlen($str) - 2); // remove outer ( and )
  $str = preg_replace("/([a-zA-Z0-9_]+?):/" , "\"$1\":", $str); // fix variable names

  $output = json_decode($str, true);
  var_dump($output);
?>

var_dump output:
array(6) {
  ["strvar"]=>
  string(6) "string"
  ["number"]=>
  int(40)
  ["boolvar"]=>
  bool(true)
  ["subobject"]=>
  array(3) {
    ["substrvar"]=>
    string(10) "sub string"
    ["subsubobj"]=>
    array(1) {
      ["deep"]=>
      string(13) "deeply nested"
    }
    ["strnum"]=>
    string(2) "56"
  }
  ["false_val"]=>
  bool(false)
  ["false_str"]=>
  string(5) "false"
}

hope this saves someone some time.
```

up
down
14

colin.mollenhour.com ¶

6 years ago

```
For those of you wanting json_decode to be a little more lenient (more like Javascript), here is a
wrapper:

<?php
function json_decode_nice($json, $assoc = FALSE){
    $json = str_replace(array("\n","\r"),"",$json);
    $json = preg_replace('/([{,]+)(\s*)([^"]+?)\s*:/','$1"$3":',$json);
    return json_decode($json,$assoc);
}
?>

Some examples of accepted syntax:

<?php
$json = '{a:{b:"c",d:["e","f",0]}}';
```

```
$json =
'{
   a : {
      b : "c",
      "d.e.f": "g"
   }
}';
?>
```

If your content needs to have newlines, do this:

```
<?php
$string = "This
Text
Has
Newlines";
$json = '{withnewlines:'.json_encode($string).'}';
?>
```

Note: This does not fix trailing commas or single quotes.

[EDIT BY danbrown AT php DOT net: Contains a bugfix provided by (sskaje AT gmail DOT com) on 05-DEC-2012 with the following note.]

Old regexp failed when json like
{aaa:[{a:1},{a:2}]}

up
down
3

gabriel dot bondaz at gmail dot com ¶
2 years ago

Get a mistery case:

if I try to json_decode this string: "[0-9]{5}", i get this results:

```
<?php
var_dump(json_decode("[0-9],{5}",));
?>
```

```
array(1) { [0] => int(0) }
```

But I expected to get an error, cause this is not a valid JSON !

up
down
1

kuroi dot neko at wanadoo dot fr ¶
2 years ago

My initial problem was to have PHP check a form in case JavaScript was disabled on the client.
I fiddled with json_decode for a while before realizing what I really wanted: to be able to initialize the same object in PHP and JavaScript from a common source file.

I ended up writing a tiny parser for a JavaScript object initializer, which is close to - but not the same thing as - a piece of JSON.
Among other things, it

- recognizes regexes (turning them into PHP strings),
- handles C/C++ comments
- accepts non-quoted field names.

This parser will accept a superset of real JS object initializer syntax (for instance non-quoted string litterals or improperly formed regexes). Error report and sanity checks are close to non-existent.
The whole idea is to share the code among JavaScript and PHP, so the syntactical checks are left to the JS interpreter.

Here is the code for those who are interrested :

```
----------------------------
class JSvarDecoderCtx {
    public function __construct ($type)
    {
        $this->fields = ($type == '[') ? array() : new stdClass();
    }

    public function add_name (&$text)
    {
        $this->name = $text;
        $text = '';
    }
    public function add_value (&$text)
    {
        // weird input like a mix of fields and array elements will cause warnings here
        if (!isset ($this->name)) $this->fields[             ] = $text;
        else                      $this->fields->{$this->name} = $text;
        $text = '';
    }
}

define ('JSVAL_TEXT'  , 12001);
define ('JSVAL_STRING', 12002);
define ('JSVAL_REGEXP', 12003);
define ('JSVAL_COMMT1', 12004);
define ('JSVAL_COMMT2', 12005);

function jsinit_decode ($json)
{
    // parse a JS initializer
    $stack = array ();
    $text = "";
    $state = JSVAL_TEXT;
    $len = strlen($json);
    for ($i = 0 ; $i != $len; $i++)
    {
        $c = $json[$i];
        switch ($state)
        {
        case JSVAL_TEXT:
            switch ($c)
            {
```

```
            case '{' :
            case '[' : array_unshift ($stack, new JSvarDecoderCtx ($c)); break;
            case '}' :
            case ']' : $stack[0]->add_value ($text); $text = array_shift ($stack)->fields; break;
            case ':' : $stack[0]->add_name  ($text); break;
            case ',' : $stack[0]->add_value ($text); break;
            case '"' :
            case "'" : $closer = $c; $state = JSVAL_STRING; break;
            case '/' :
                assert($i != ($len-1));
                switch ($json[$i+1])
                {
                case '/': $state = JSVAL_COMMT1; break;
                case '*': $state = JSVAL_COMMT2; break;
                default : $state = JSVAL_REGEXP; $text .= $c;
                }
                break;
            case "\r":
            case "\n":
            case "\t":
            case ' ' : break;
            default  : $text .= $c;
            }
            break;
        case JSVAL_STRING: if  ($c != $closer)              $text .= $c; else $state = JSVAL_TEXT;
break;
        case JSVAL_REGEXP: if (($c !=  ',') && ($c !=  '}')) $text .= $c; else { $i--; $state =
JSVAL_TEXT; } break;
        case JSVAL_COMMT1: if (($c == "\r") || ($c == "\n")) $state = JSVAL_TEXT; break;
        case JSVAL_COMMT2:
            if ($c != '*') break;
            assert($i != ($len-1));
            if ($json[$i+1] == '/') { $i++; $state = JSVAL_TEXT; }
        }
    }
    assert ($state == JSVAL_TEXT);
    return $text;
}
-----------------------------
Test pattern :
{
     errors: "form-errors",   // CSS class for validation errors tooltip
    i: /* integer   */ /^[0-9]+$/,
    err: {
      '*': "mandatory field",
      i: "must be an integer",
    },
    a:['ga','bu','zo','meuh'] // http://www.archimedes-lab.org/shadoks/shadoiku.html
}
-----------------------------
result :

stdClass Object
(
```

```
     [errors] => form-errors
     [i] => /^[0-9]+$/
     [err] => stdClass Object
        (
            [*] => mandatory field
            [i] => must be an integer
        )
    [a] => Array
        (
            [0] => ga
            [1] => bu
            [2] => zo
            [3] => meuh
        )
)
```

up
down
5
alexvonweiss at googlemail dot com ¶
5 years ago

Consider that JSON can differ between int and string. So

```
<?php
var_dump(json_decode('{"foo": 12}'));
// array(1) { ["foo"]=> int(12) }

var_dump(json_decode('{"foo": "12"}'));
// array(1) { ["foo"]=> string(12) }
?>
```

Numbers that cannot be handled by integer seems to become float casted. This can be a problem if you transfer big numbers like facebook ids over JSON. Either you avoid numbers by cast everything to string before JSON.stringify or you have to use number_format if the value bacome a float value.

```
<?php
// test
$x = json_decode('{"foo": 123456789012345}');
echo sprintf('%1$f', $x->foo).PHP_EOL;
echo sprintf('%1$u', $x->foo).PHP_EOL;
echo sprintf('%1$s', $x->foo).PHP_EOL;
echo strval($x->foo).PHP_EOL;
echo (string) $x->foo.PHP_EOL;
echo number_format($x->foo, 0, '', '').PHP_EOL;

// output
123456789012345.000000  // printf %f
2249056121              // printf %u
1.2345678901234E+14     // printf %s
1.2345678901234E+14     // strval()
1.2345678901234E+14     // cast (string)
2249056121              // cast (int)
123456789012345         // number_format()
?>
```

up

down
2
christian dot knoop at gmx dot net ¶
3 years ago
If you store your json-string in an utf8-file and read it with file_get_contents, please make sure to strip leading BOM (byte order mark) before decoding it with json_decode. Otherwise json_decode will fail creating an associative array. Instead it will return your data as a string.
up
down
6
steven at acko dot net ¶
8 years ago
json_decode()'s handling of invalid JSON is very flaky, and it is very hard to reliably determine if the decoding succeeded or not. Observe the following examples, none of which contain valid JSON:

The following each returns NULL, as you might expect:

```
<?php
var_dump(json_decode('['));             // unmatched bracket
var_dump(json_decode('{'));             // unmatched brace
var_dump(json_decode('{}}'));           // unmatched brace
var_dump(json_decode('{error error}')); // invalid object key/value
notation
var_dump(json_decode('["\"]'));         // unclosed string
var_dump(json_decode('[" \x "]'));      // invalid escape code
```

Yet the following each returns the literal string you passed to it:

```
var_dump(json_decode(' ['); // unmatched bracket
var_dump(json_decode(' {'); // unmatched brace
var_dump(json_decode(' {}}'); // unmatched brace
var_dump(json_decode(' {error error}')); // invalid object key/value notation
var_dump(json_decode('"\"')); // unclosed string
var_dump(json_decode('" \x "')); // invalid escape code
?>
```

(this is on PHP 5.2.6)

Reported as a bug, but oddly enough, it was closed as not a bug.

[NOTE BY danbrown AT php DOT net: This was later re-evaluated and it was determined that an issue did in fact exist, and was patched by members of the Development Team. See http://bugs.php.net/bug.php?id=45989 for details.]
up
down
0
Nathan Salter ¶
3 months ago
Remember that some strings are valid JSON, despite not returning very PHP-friendly values:

```
<?php

var_dump(json_decode('null')); // NULL
echo json_last_error_msg(); // No error
```

```
var_dump(json_decode('false')); // bool(false)
echo json_last_error_msg(); // No error

?>
```

up
down
-1
Eko ¶
1 year ago

UTF8 decoding for all json array values :

```
function JsonUtf8Decode(&$v, $k) { $v = utf8_decode($v); }
array_walk_recursive($config, "JsonUtf8Decode");
```

up
down
-1
qrworld.net ¶
2 years ago

I found this post http://softontherocks.blogspot.com/2014/11/funcion-jsondecode-de-php.html

They show there an example of using json_decode:

```
$json = '{"users":[{ "user": "Carlos", "age": 30, "country": "Spain" }, { "user": "John", "age": 25,
"country": "United States" }]}';

// Decodificamos la cadena JSON
$obj = json_decode($json);
var_dump($obj);
/* devuelve
object(stdClass)#1 (1) { ["users"]=> array(2) { [0]=> object(stdClass)#2 (3) { ["user"]=> string(6)
"Carlos" ["age"]=> int(30) ["country"]=> string(5) "Spain" } [1]=> object(stdClass)#3 (3) {
["user"]=> string(4) "John" ["age"]=> int(25) ["country"]=> string(13) "United States" } } }
*/

// Devolvemos el resultado como array
$obj = json_decode($json, true);
var_dump($obj);
/* devuelve
array(1) { ["users"]=> array(2) { [0]=> array(3) { ["user"]=> string(6) "Carlos" ["age"]=> int(30)
["country"]=> string(5) "Spain" } [1]=> array(3) { ["user"]=> string(4) "John" ["age"]=> int(25)
["country"]=> string(13) "United States" } } }
*/

// Limitamos la profundidad de la recursividad a 2
$obj = json_decode($json, true, 2);
var_dump($obj);
/* devuelve
NULL -> la profundidad es 4
*/
```

up
down
-2
nsardo ¶
2 years ago

Just a quick note, as I got caught with this:

The json that is input into json_decode MUST be well formed, such that key's are QUOTED. I wasted time debugging, a$$suming that the following would be fine:

```
[{id:2, name:"Bob",phone:"555-123-4567"}]
```

It returned null, basically. I had to QUOTE the keys:

```
[{"id":2, "name":"Bob", "phone":"555-123-4567"}]
```

THEN, everything worked as it should.

up
down
-2

Antony ¶
8 months ago

Tried this on PHP Version 5.3.3 and json_decode returns NULL:

```
<?php
$payload = '{"DATA":{"KEY":"VALUE"},"SOME_OTHER_DATA":["value1","value2"]}';
$results = json_decode($payload, true, 512, 0);
var_dump($results); // NULL
?>
```

But this works:

```
<?php
$payload = '{"DATA":{"KEY":"VALUE"},"SOME_OTHER_DATA":["value1","value2"]}';
$results = json_decode($payload, true);
var_dump($results); // array(2) { ["DATA"]=> array(1) { ["KEY"]=> string(5) "VALUE" }
["SOME_OTHER_DATA"]=> array(2) { [0]=> string(6) "value1" [1]=> string(6) "value2" } }
?>
```

Seems that json_decode doesn't work properly when passing defaults $depth (512) and $options (0) parameters.

But the signature of json_decode is:

```
mixed json_decode ( string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0 ]]] )
```

And the min required version of PHP is PHP 5 >= 5.2.0.

up
down
-8

nospam (AT) hjcms (DOT) de ¶
9 years ago

You can't transport Objects or serialize Classes, json_* replace it bei stdClass!

```
<?php

$dom = new DomDocument( '1.0', 'utf-8' );
$body = $dom->appendChild( $dom->createElement( "body" ) );
$body->appendChild( $dom->createElement( "forward", "Hallo" ) );
```

```
$JSON_STRING = json_encode(
   array(
      "aArray" => range( "a", "z" ),
      "bArray" => range( 1, 50 ),
      "cArray" => range( 1, 50, 5 ),
      "String" => "Value",
      "stdClass" => $dom,
      "XML" => $dom->saveXML()
   )
);

unset( $dom );

$Search = "XML";
$MyStdClass = json_decode( $JSON_STRING );
// var_dump( "<pre>" , $MyStdClass , "</pre>" );

try {

   throw new Exception( "$Search isn't a Instance of 'stdClass' Class by json_decode()." );

   if ( $MyStdClass->$Search instanceof $MyStdClass )
      var_dump( "<pre>instanceof:" , $MyStdClass->$Search , "</pre>" );

} catch( Exception $ErrorHandle ) {

   echo $ErrorHandle->getMessage();

   if ( property_exists( $MyStdClass, $Search ) ) {
      $dom = new DomDocument( "1.0", "utf-8" );
      $dom->loadXML( $MyStdClass->$Search );
      $body = $dom->getElementsByTagName( "body" )->item(0);
      $body->appendChild( $dom->createElement( "rewind", "Nice" ) );
      var_dump( htmlentities( $dom->saveXML(), ENT_QUOTES, 'utf-8' ) );
   }
}

?>
```

up
down
-9

hellolwq at gmail dot com ¶

3 years ago

```
json_decode can not handle string like:\u0014,it will return null,and with the error
<?php json_last_error()=JSON_ERROR_CTRL_CHAR ?>
according ascii_class from json module, use the next codes to fix the bug:
<?php
$str = preg_replace_callback('/([\x{0000}-\x{0008}]|[\x{000b}-\x{000c}]|[\x{000E}-\x{001F}])/u',
function($sub_match){return '\u00' . dechex(ord($sub_match[1]));},$str);
var_dump(json_decode($str));
?>
```

up
down
-11

hellolwq at gmail dot com ¶
3 years ago

```
json_decode can not handle string like:\u0014,it will return null,and with the error
<?php json_last_error()=JSON_ERROR_CTRL_CHAR ?>
according ascii_class from json module, use the next codes to fix the bug:
<?php
$str = preg_replace_callback('/([\x{0000}-\x{0008}]|[\x{000b}-\x{000c}]|[\x{000E}-\x{001F}])/u',
function($sub_match){return '\u00' . dechex(ord($sub_match[1]));},$str);
var_dump(json_decode($str));
?>
```

up
down
-1

Patanjali ¶
2 months ago

A recursive function to convert a decoded JSON object into a nested array.

```
<?php
function to_array(&$object=''){
  // IF OBJECT, MAKE ARRAY
  if(is_object($object)){$object = (array)$object;}

  // IF NOT ARRAY OR EMPTY ARRAY, RETURN = LEAVES SCALARS
  if(!is_array($object)||empty($object)){return;}

  // FOR EACH ITEM, RECURSE VALUE
  foreach($object as &$Value){to_array($Value);}
}
?>

<?php
// MAKE INTO ARRAY
$Object=json_decode($Json);
to_array($Object);
?>
```

up
down
-4

descartavel1+php at gmail dot com ¶
1 year ago

```
<?php
/* wrapper to php's json_encode to work around the fact that some options we use are
* not available on php5.2, which we still support
* call to json_encode is silenced as we don't care about encoding warnings
*/
public static function php_json_encode_as_object($input){
    if( (PHP_MAJOR_VERSION == 5 && PHP_MINOR_VERSION < 3)
        || PHP_MAJOR_VERSION < 5
    ){
        // no support for JSON_FORCE_OBJECT
        return @json_encode((object)$input);
    }else{
        return @json_encode($input, JSON_FORCE_OBJECT);
    }
```

```
}
?>
```
up
down
-46
evengard at trioptimum dot com ¶
5 years ago
There is a problem when passing to json_decode a string with the "\" symbol. It seems to identify it as an escape character and trying to follow it. So sometimes it leads to failed parsing.

It seems that just replacing it with "\\" helps.

```
<?php
print_r(json_decode(str_replace('\\', '\\\\', '{"name":"/\"}')));
?>
```

where /\ is the string which doesn't worked.

add a note