PHP – JSON

Standard distributions of PHP have the JSON support enabled by default. The PHP extension implements the JavaScript Object Notation (JSON) data interchange format. The JSON extension in PHP parser handles the JSON data.

JSON (JavaScript Object Notation) is a lightweight, text-based, language-independent data interchange format. JSON defines a small set of formatting rules for the portable representation of structured data. It is a text based data format that is easy for the humans as well as machines to read.

The JSON extension in PHP version 5.2 onwards provides a number of predefined constants, JSON related functions, and also a JsonException class.

PHP JSON Functions

PHP has the following JSON functions −

json_encode()

This function returns a string containing the JSON representation of the supplied value. If the parameter is an array or object, it will be serialized recursively.

```
json_encode(mixed $value, int $flags = 0, int $depth = 512): string|false
```

json_decode()

This function takes a JSON encoded string and converts it into a PHP value.

```
json_decode(
   string $json,
   ?bool $associative = null,
   int $depth = 512,
   int $flags = 0
): mixed
```

When the associative parameter of this function is true, JSON objects will be returned as associative arrays; when false, JSON objects will be returned as objects.

The encode/decode operations are affected by the supplied flags. The predefined constants and their integer values are as below −

Predefined Constant	Values
JSON_HEX_TAG	1
JSON_HEX_AMP	2
JSON_HEX_APOS	4
JSON_HEX_QUOT	8
JSON_FORCE_OBJECT	16
JSON_NUMERIC_CHECK	32
JSON_UNESCAPED_SLASHES	64
JSON_PRETTY_PRINT	128
JSON_UNESCAPED_UNICODE	256

json_last_error_msg()

This function returns the error string of the last json_encode() or json_decode() call.

```
json_last_error_msg(): string
```

"No error" message is returned if no error has occurred.

json_last_error()

This function returns an integer.

```
json_last_error(): int
```

The function returns an integer corresponding to one of the following constants −

Sr.No	Constant & Meaning
1	**JSON_ERROR_NONE** No error has occurred
2	**JSON_ERROR_DEPTH** The maximum stack depth has been exceeded
3	**JSON_ERROR_STATE_MISMATCH** Invalid or malformed JSON

4	**JSON_ERROR_CTRL_CHAR** Control character error, possibly incorrectly encoded
5	**JSON_ERROR_SYNTAX** Syntax error
6	**JSON_ERROR_UTF8** Malformed UTF-8 characters, possibly incorrectly encoded
7	**JSON_ERROR_RECURSION** One or more recursive references in the value to be encoded
8	**JSON_ERROR_INF_OR_NAN** One or more **NAN** or **INF** values in the value to be encoded
9	**JSON_ERROR_UNSUPPORTED_TYPE** A value of a type that cannot be encoded was given
10	**JSON_ERROR_INVALID_PROPERTY_NAME** A property name that cannot be encoded was given
11	**JSON_ERROR_UTF16** Malformed UTF-16 characters, possibly incorrectly encoded

Example

The following PHP code encodes a given array to JSON representation, and decodes the JSON string back to PHP array.

Open Compiler

```
<?php
   $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
   $encoded = json_encode($arr);
   echo "The initial array: " . PHP_EOL;
   var_dump($arr);
   echo "Encoded JSON: $encoded" . PHP_EOL;

   $decoded = json_decode($encoded);
   echo "Array obtained after decoding: " . PHP_EOL;
   var_dump($decoded);
?>
```

It will produce the following **output** −

```
The initial array:
array(5) {
  ["a"]=>
  int(1)
  ["b"]=>
  int(2)
  ["c"]=>
  int(3)
  ["d"]=>
  int(4)
  ["e"]=>
  int(5)
}
Encoded JSON: {"a":1,"b":2,"c":3,"d":4,"e":5}
Array obtained after decoding:
object(stdClass)#1 (5) {
  ["a"]=>
  int(1)
  ["b"]=>
  int(2)
  ["c"]=>
  int(3)
  ["d"]=>
  int(4)
  ["e"]=>
  int(5)
}
```