Date 23/02/2017

PHP Interpreter ⟹ (means) Zend Engine

Notice that the JSON string contains no white space, which can make it difficult to read. A JSON_PRETTY_PRINT option can be used to format the JSON output.

Note: Javascript Array is/means only a numeric array as in PHP. Javascript object use 'named index' to access it's members like person.firstName. Javascript variables can be objects. Arrays are special kind of objects. Because of this, we can have variables of different types in the same Array.

Reference: https://www.w3schools.com/js/js_arrays.asp

Numerically Indexed Array Example:

This example demonstrates passing a numerically indexed PHP array consisting of string, numeric, boolean, and null values to json_encode. We echo the output of json_encode into a Javascript variable & the result is an array literal:

```
<?php
$ar = array('apple', 'orange', 1, false, null, true,
?>          3+5);
<script type="text/javascript">
var ar = <?php echo json_encode($ar); ?>;
// ["apple","orange",1,false,null,true,8];
// access 4th element in array
alert(ar[3]); // false
</script>
```

Date
23/02/24

we use an alert to demonstrate access to elements of the array. Notice that, here, the data types have been preserved.

Associative Array Example:

This example shows an associative PHP array output into Javascript using json_encode. Notice that, PHP's associative array becomes an object literal in Javascript:

```
<?php
$book = array(
    "title" => "Javascript: The Definitive Guide",
    "author" => "David Flanagan",
    "edition" => 6
);
?>
<script type="text/javascript">
var book = <?php echo json_encode($book, JSON_PRETTY_PRINT); ?>;
/* var book = {
    "title": "Javascript: The Definitive Guide",
    "author": "David Flanagan",
    "edition": 6
}; */
console.log(book);
alert(book.title);
</script>
```

We use the JSON_PRETTY_PRINT option as the second argument to json_encode to display the output in a readable format.

We can access object properties using dot syntax as

Date
23/02/2017

displayed with the alert included above, or square bracket syntax: book['title'].

Note: console.log() in javascript is equivalent to var_dump() in PHP. We can see the output of console.log() in chrome developer tool under "Console" tab. Under "Source" tab, we can see page source code including HTML, Javascript, and inline css. Under "Network" → "XHR" tab, we can see all ajax request/response details.

Using JavaScript's JSON.parse()

Javascript's JSON.parse method parses a JSON string and returns the Javascript equivalent.

Remember:

PHP numeric array ⇒ is equivalent to Javascript Array

PHP associative array ⇒ is equivalent to Javascript object

```
<?php
$numArr = array("apple", "rose", "table");   // PHP7 way of declaring array
$assArr = ["fruit"⇒"apple", "flower"⇒"rose", "furniture"⇒"table", "salary"⇒6];
?>
```

JSON representation of a numerically indexed array (equivalent to Javascript Array):

```
var json = '["apple", "rose", "table"]';   // string
```

JSON representation of a named indexed array (equivalent to Javascript object):

```
var json = '{"fruit": "apple", "flower": "rose", "furniture": "table", "salary": 6}';   // string
```

Date
23/02/2017

```
<?php $arr = ["apple", "rose", "table"]; } ⇒ ajax response
$json = json_encode($arr); ?>
```

```
<script type="text/javascript">
var json = JSON.stringify( ); var
```

ajax response

We pass the JSON string to JSON.parse() and assign the result to a data variable. We pass the data variable to console.log() to verify that the result is an array. Then we demonstrate access to an element of the data array with an alert.

```
<script type="text/javascript">
var json = '["apple","rose","table"]';
var data = JSON.parse(json);
console.log(data); //["apple","rose","table"]
alert(data[2]); // table
</script>
```

There may be situations where you are sure of your data and would not need to use JSON.parse. However, use of JSON.parse is advised if the data is coming from an untrusted source. JSON.parse checks the data passed to it and throws a SyntaxError if it's not valid JSON. Any data that could pose a risk is deleted.

Using JSON.parse() with PHP's json_encode():

The following demonstrates how to use JSON.parse() on the JSON string output by PHP's json_encode function:

```
<?php
$books = [
    [
        "title" => "title1";
```

Date 23/02/2017

var data = JSON.parse(json); </script>

```
        "author" => "author1"
    ],
    [
        "title" => "title2",
        "author" => "author2"
    ],
    [
        "title" => "title3",
        "author" => "author3"
    ]
];
?>
```

In the following Javascript segment, we show passing the PHP $books array to json_encode and the json_encode output to JSON.parse. The result is assigned to the Javascript variable books. We display the value of books commented out!

```
<script type="text/javascript">
// using JSON.parse on the output of json_encode
var books = JSON.parse('<?php echo json_encode(
                                $books); ?>');
/* output (with some whitespace added for readability)
[
    {"title": "title1", "author": "author1"},
    {"title": "title2", "author": "author2"},
    {"title": "title3", "author": "author3"}
]
*/
console.log(books[1].author); // author2
</script>
```