

How to Use JSON Data with PHP or JavaScript

January 22nd, 2017 / [Share](#) / [Edit on Github](#)

data

javascript

json

php

[JSON](#) is used to transmit data between a server and a browser. Here is a basic example of what might be in a `.json` string.

```
{
  "name": "Tania",
  "title": "Web Developer",
  "website": ""
}
```

As you can see, it's a human readable format of data that might traditionally be stored in a table. Some companies might have public `.json` files located that you can access and extract data from (an API you can connect to). You might also save a `.json` file somewhere in your project that you want to extract data from.

Goals

JSON data can be accessed and utilized with many programming languages. In this tutorial, we'll learn how to access JSON with PHP and JavaScript.

Prerequisites

- You must either have [a local server](#) set up, or a host that runs PHP and some basic PHP knowledge.
- Basic [knowledge of programming concepts](#) (arrays and variables) and using JavaScript.

What is JSON?

JSON stands for **JavaScript Object Notation**. It is data saved in a `.json` file, and consists of a series of **key/value pairs**.

```
{ "key": "value" }
```

The **value** of any JSON key can be a string, Boolean, number, null, array, or object. Comments are not allowed in JSON.

Although JSON resembles an object or an array, **JSON is a string**. A serialized string, which means it can later be parsed and decoded into data types.

Using data from JSON with PHP

First, to drill in that JSON is simply a string, we're going to write JSON into a PHP string and apply it to a variable called `$data`.

```
$data = '{  
  "name": "Aragorn",  
  "race": "Human"  
';
```

Then we'll use the `json_decode()` function to convert the JSON string into a PHP object.

```
$character = json_decode($data);
```

Now we can access it as a PHP object.

```
echo $character->name;
```

Here's the whole file.

```
<?php  
  
$data = '{  
  "name": "Aragorn",  
  "race": "Human"  
';  
  
$character = json_decode($data);  
echo $character->name;
```

Here is the output.



Aragorn

Accessing a JSON feed from a URL

From here out, we'll put all JSON data into its own `.json` file. This way, we can retrieve the contents of the file instead of keeping it as a PHP string.

Here's what **data.json** will look like.

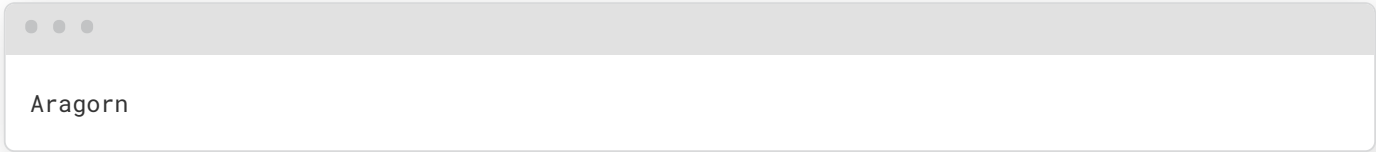
```
[{
  "name": "Aragorn",
  "race": "Human"
}, {
  "name": "Legolas",
  "race": "Elf"
}, {
  "name": "Gimli",
  "race": "Dwarf"
}]
```

And here's how we'll extract that data in PHP.

```
$url = 'data.json'; // path to your JSON file
$data = file_get_contents($url); // put the contents of the file into a variable
$characters = json_decode($data); // decode the JSON feed
```

In order to get one entry, we'll have to access the appropriate array number. Remember, counting begins with 0 in programming.

```
echo $characters[0]->name;
```



Aragorn

I can access all the data in the array with a `foreach` loop.

```
foreach ($characters as $character) {
  echo $character->name . '<br>';
}
```



Aragorn
Legolas
Gimli

Here is the full PHP file.

```
<?php

$url = 'data.json'; // path to your JSON file
$data = file_get_contents($url); // put the contents of the file into a variable
$characters = json_decode($data); // decode the JSON feed

echo $characters[0]->name;

foreach ($characters as $character) {
    echo $character->name . '<br>';
}
```

We can display the data in a table, for an example.

```
<table>
  <tbody>
    <tr>
      <th>Name</th>
      <th>Race</th>
    </tr>
    <?php foreach ($characters as $character) : ?>
      <tr>
        <td> <?php echo $character->name; ?> </td>
        <td> <?php echo $character->race; ?> </td>
      </tr>
    <?php endforeach; ?>
  </tbody>
</table>
```

Aragorn	Human
Gimli	Dwarf
Legolas	Elf

In this example, I'm using the alternate syntax for `foreach`, which looks like `foreach() : /* loop */ endforeach;` instead of `foreach() { /* loop */ }`. This is often preferable when outputting HTML.

Using associative arrays

There's another way we can access the data in PHP. If you pass `true` as the argument in `json_decode()`, the data becomes an associative array instead of an object. This means we'll be using square bracket notation `[]` instead of the skinny arrow `->`.

```
$characters = json_decode($data, true); // decode the JSON feed and make an associative array
```

Instead of `->race`, we will access the value with `['race']`.

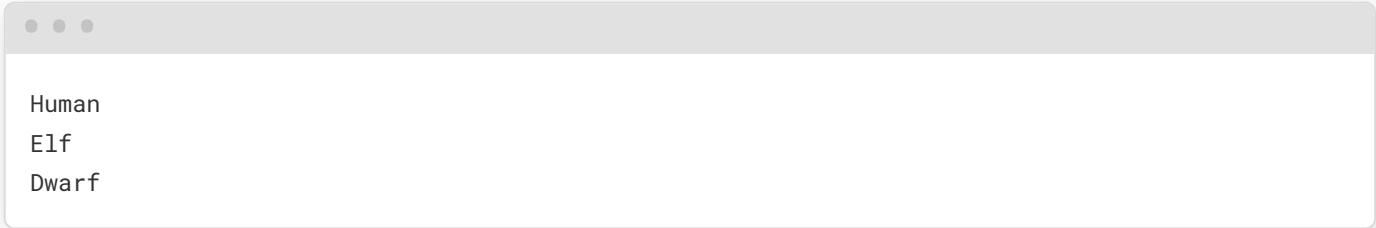
```
echo $characters[0]['race'];
```



Human

And here's how to access the loop.

```
foreach ($characters as $character) {  
    echo $character['race'] . "\n";  
}
```



Human
Elf
Dwarf

Getting data from nested arrays

So far, we've only used JSON feeds with key/value pairs, but it's common to encounter nesting. Here's another nerdy example, which we can save in a new file called **wizards.json**.

```
[  
  {  
    "name": "Harry Potter",  
    "wand": [  
      {  
        "core": "phoenix feather",  
        "length": "11 inches",  
        "wood": "holly"  
      }  
    ]  
  },  
]
```

```
{
  "name": "Hermione Granger",
  "wand": [
    {
      "core": "dragon heartstring",
      "length": "10 and 3/4 inches",
      "wood": "vine"
    }
  ]
}
```

Decoding the feed.

```
$url = 'wizards.json';
$data = file_get_contents($url);
$wizards = json_decode($data, true);
```

We'll be able to access the nested array using `$wizard['key'][0]['key']` in a loop, or whatever number corresponds correctly if you only want to print one.

```
foreach ($wizards as $wizard) {
    echo $wizard['name'] . '\s wand is ' .
    $wizard['wand'][0]['wood'] . ', ' .
    $wizard['wand'][0]['length'] . ', with a ' .
    $wizard['wand'][0]['core'] . ' core. <br>';
}
```

```
Harry Potter's wand is holly, 11 inches, with a phoenix feather core.
Hermione Granger's wand is vine, 10 and 3/4 inches, with a dragon heartstring core.
```

Converting a PHP object or array into JSON

Just as you use `json_decode()` to turn JSON into PHP, you can turn PHP into JSON with `json_encode()`.

```
$data = [
    'name' => 'Aragorn',
    'race' => 'Human'
];

echo json_encode($data);
```

We made a PHP array and encoded it. Here's the output:

```
{ "name": "Aragorn", "race": "Human" }
```

Using data from JSON with JavaScript

We're going to create a JavaScript variable called `data` and apply the JSON string.

```
var data = '[ { "name": "Aragorn", "race": "Human" }, { "name": "Gimli", "race": "Dwarf" } ]'
```

Now we'll use JavaScript built in `JSON.parse()` function to decode the string.

```
data = JSON.parse(data);
```

From here we can access the data like a regular JavaScript object.

```
console.log(data[1].name)
```

```
Gimli
```

And we can loop through each iteration with a `for` loop.

```
for (var i = 0; i < data.length; i++) {  
  console.log(data[i].name + ' is a ' + data[i].race + '.')  
}
```

```
Aragorn is a Human.  
Gimli is a Dwarf.
```

That was easy! Now, we'll probably need to access JSON from a URL. There's an extra step involved, where we have to make a request to the file. Let's just take the above JSON string and put it in **data.json**.

```
;[
  {
    name: 'Aragorn',
    race: 'Human',
  },
  {
    name: 'Gimli',
    race: 'Dwarf',
  },
]
```

Now we'll make an `XMLHttpRequest()` .

```
var request = new XMLHttpRequest()
```

We'll open the file (**data.json**) via GET (URL) request.

```
request.open('GET', 'data.json', true)
```

From here, we'll parse and work with all our JSON data within the `onload` function.

```
request.onload = function() {
  // begin accessing JSON data here
}
```

Then finally, submit the request.

```
request.send()
```

Here's the final code.

```
var request = new XMLHttpRequest()

request.open('GET', 'data.json', true)

request.onload = function() {
  // begin accessing JSON data here
  var data = JSON.parse(this.response)

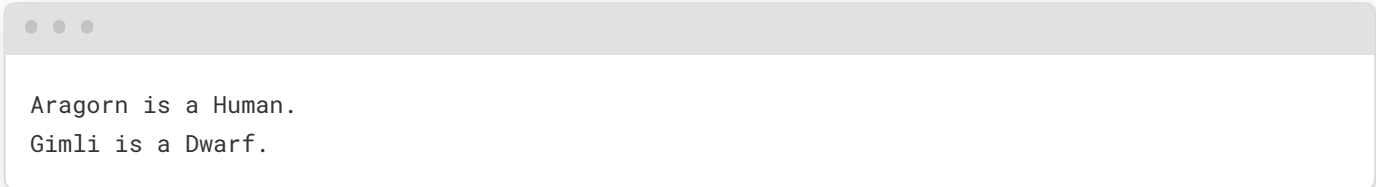
  for (var i = 0; i < data.length; i++) {
```



```
    console.log(data[i].name + ' is a ' + data[i].race + '.')
  }
}

request.send()
```

And the output.



```
Aragorn is a Human.
Gimli is a Dwarf.
```

Using Fetch

Now you can also use the Fetch API to do the same thing. Read [How to Use the JavaScript Fetch API to Get JSON Data](#) for an easier method to get this data.

```
// Replace ./data.json with your JSON feed
fetch('./data.json')
  .then(response => {
    return response.json()
  })
  .then(data => {
    // Work with JSON data here
    console.log(data)
  })
  .catch(err => {
    // Do something for an error here
  })
```

Using jQuery

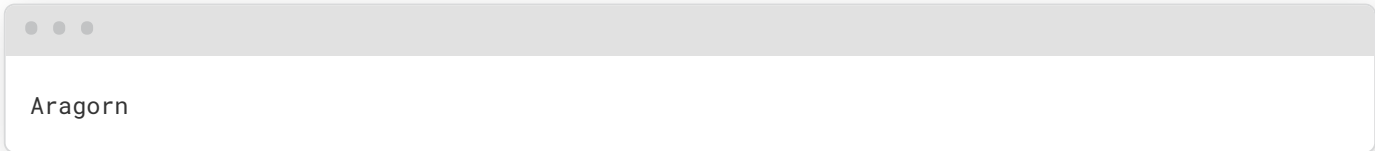
As you can see, it's not too difficult to retrieve a JSON feed with plain JavaScript. However, it's even easier with jQuery, using the `getJSON()` function. If you don't know how jQuery works, you'll need to load the [jQuery JavaScript library](#) before any of this custom code.

```
$(document).ready(function() {
  $.getJSON('data.json', function(data) {
    // begin accessing JSON data here
    console.log(data[0].name)
  })
})
```

You might also see jQuery access JSON via an AJAX request, which is a little more verbose.

```
$(document).ready(function() {  
  var data  
  $.ajax({  
    dataType: 'json',  
    url: 'data.json',  
    data: data,  
    success: function(data) {  
      // begin accessing JSON data here  
      console.log(data[0].name)  
    },  
  })  
})
```

Both will have the same output.



Aragorn
