9) HTTP & Web Services Lesson

JSON: The Preferred Data Format for Web Services

13 min to complete · By Ryan Desmond, Jared Larsen

Contents

Think back to the previous lesson on web services. Remember the part discussing the possible data formats a RESTful service can support? It listed things like plain text and binary, but also JSON. This lesson covers JSON (pronounced "JAY-son" or "JAY-sawn", depending on who you ask), an elegant data format that has become the de-facto standard for web service data interactions.

What is JSON

Even though it sounds like it, JSON was not created and named by someone called Jason. Instead, JSON is short for JavaScript Object Notation. The acronym may specify

JavaScript, but JSON is widely adopted and is used far beyond the world of JavaScript. One of the reasons it is so widely adopted is because of its simplicity. JSON assigns roles and meanings to a few symbols to create a standalone & comprehensive data format. This lesson will teach you all about these symbols and the formatting rules.

Info: One of the biggest pros for JSON is that it plays exceptionally well with the object-oriented paradigm. Before JSON, web services mainly used XML (Extensible Markup Language) which does this much less gracefully, is less expressive and is harder to read.

Syntax & Structure

Below, you can see a sample of some JSON. Take a look, but take your time and try not to get overwhelmed. We'll pick it apart bit by bit.


```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "emailAddresses": [
    "Sincere@april.biz",
    "bret@imail.com"
  ],
  "active": true,
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
```

```
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

Opening and Closing Brackets

JSON indicates the beginning and end of objects with curly `{ }` brackets. You can see this in the example above. This JSON as a whole represents an object, and here is a breakdown of the first several fields:

- `id` : number
- `name` : string
- `username` : string
- `emailAddresses` : array (of strings)
- `active` : boolean
- `address` : object

Make sure you pay attention to the curly brackets. Just like your Java classes and methods expect a specific structure, JSON also expects data to follow a particular structure. To put this into perspective, here is a Java class representing the same fields detailed above:

```
public class User {

    private Long id;
    private String name;
    private String username;
    private List<String> emailAddresses;
    private boolean active;
    private Address address;

    // ... getters, setters, etc.
}
```

Now that you've been introduced to JSON structure, it's time to break it down.

JSON Key-Value Pairs

Starting with the basics, you can see that JSON data is stored in key-value pairs. Like the `username` line in the example above, the key is the field name, and it must be surrounded with quotation marks. The key is then followed by a colon, which separates the value that belongs to the key. JSON values must be one of the following types:

- number
- string
- object
- array
- boolean
- null

Putting this together, the key-value format is as follows:

```
Numbers:
"insert_key": 1234567890

Strings:
"insert_key": "I'm a String"

Objects:
"insert_key": {"another_key": "I'm a String"}

Arrays:
"insert_key": ["I'm a String", "I'm a String too!"]

Boolean:
"insert_key": true/false

Null :
"insert_key": null
```

Depending on the value type, you may have to surround it with quotation marks. The general rule is if the datatype is anything other than a `number` , a `boolean` or `null` , it needs quotes.

Multiple Key-Value Pairs

You just learned how to express a single line of data in JSON, but you want to be able to store more than just one field. To do this, JSON employs commas to separate fields. Here's what it looks like:

```
"insert_key": insert_number,
"insert_key": "insert_String",
"insert_key": true/false
```

Note: Notice how the final key-value pair does not have a comma.

Let's Talk Arrays

As you know, arrays are some of the most intensely used coding data structures out there. They are great tools, and JSON also needs a way to express them. Arrays in JSON begin and end with square brackets, and a comma separates each field in the array. Here's an example:

```
[
1573857,
2573852,
3573959
]
```

Remember that the quotation rules for key-value pairs still hold true for data embedded in an array. An array with String values would look like this:

```
[
"this is part of the example",
"this is another part of the example",
"this is another another part of the example"
]
```

Info: Arrays can contain any type of data. You can mix and match data types within arrays (even though this is not recommended - you'll learn why later.) and can even embed arrays within arrays.

You now know what a JSON array looks like, but how do they fit it into the overall key-value structure? To do this, it may be easier to imagine that arrays are just another datatype like string and booleans. Following this reasoning, Strings or numbers in JSON key-value pairs only need to be replaced by an Array. Take a look:

```
{
  "thisIsAKey": null,
  "thisIsAlsoAKey": [
    "this",
    "is",
    "an",
    "array",
    "of",
    "strings"
  ]
}
```

Where do Objects Fit In?

As mentioned above, to indicate the start of an object and the end of an object, JSON uses `{` and `}` respectively. This is similar to an array, except that a JSON object is much more flexible. Instead of just storing data points, a JSON object stores key-value pairs. This means you can embed objects within objects, and arrays within objects (and objects within arrays!). Let's see what this looks like:

```
{
  "name": "Ben",
  "firstObject": {
    "embeddedKey": true,
    "embeddedArray": [
      1,
      2,
```

```
      3,
      4
    ],
    "embeddedObject": {
      "embeddedKeyWithinAnObjectWithinAnObject": 4759385,
      "thatLastKeyWasLong": "very long"
    }
  }
}
```

The above JSON begins with a simple key-value pair with key `name` and value "Ben". The next key-value pair is called `firstObject`, and it stores an object with three fields:

- A key-value pair called `embeddedKey` with value `true`.
- An array named `embeddedArray` which stores the values 1, 2, 3 and 4.
- Another object named `embeddedObject`. `embeddedObject` in turn contains two primitive key-value pairs.

Sending a Single JSON Value

Almost everything you send in JSON is technically an object and requires opening and closing curlys accordingly. There is however an exception to this rule. You can leave out the two curly brackets when sending a single array. They are effectively replaced by square brackets indicating the start and end of the array. Here's an example of an array of objects (enjoy the captivating lorem ipsum):

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat optio reprehenderit",
    "body": "quia et suscipit"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum temporen debitis qui neque nisi nulla"
```

```
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure occaecati omnis eligendi"
  },
  {
    "userId": 1,
    "id": 4,
    "title": "eum et est occaecati",
    "body": "ullam et saepe reiciendis luptatem rerum illo"
  }
]
```

The example above not only illustrates how to express a single array in JSON but also how to store objects within arrays.

Learn by Doing 

Location: `https://jsoneditoronline.org`

- Visit JSON Editor Online, and play with the code editor.
- Compare various JSON code structures to the "tree" view pane to get a better understanding of how JSON is used to represent objects, fields, and arrays.

Summary: JSON Format

- JSON provides a popular and elegant way of formatting data.
- JSON is short for JavaScript Object Notation.
- JSON indicates the beginning and end of objects with curly brackets.
- JSON data is stored in key-value pairs.
- Arrays in JSON begin and end with square brackets, and a comma separates each piece of data in the array.

Previous Next → Video: JSON Data Format