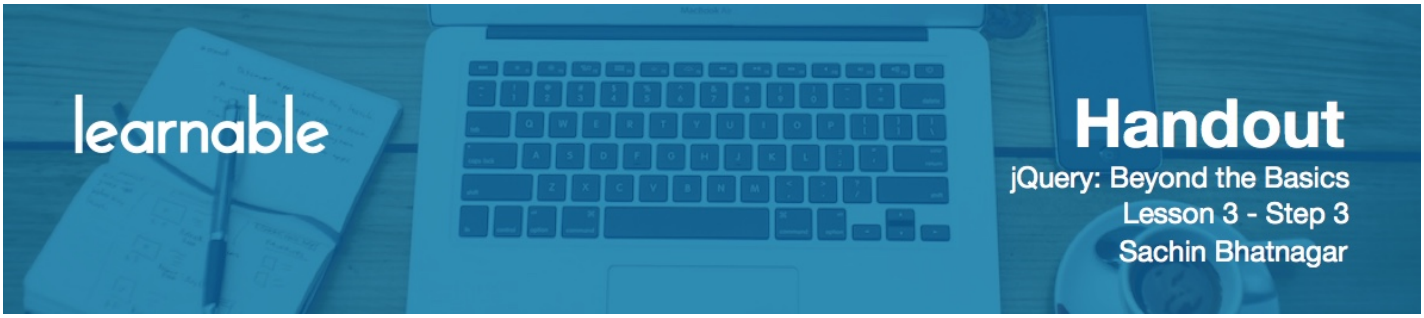

Introduction to JSON format

By definition AJAX seems to imply that it only accepts XML, but in reality AJAX works with all kinds of formats. Perhaps the most widely used format these days is **JSON* (Java Script Object Notation)**. It is a standard JavaScript object with key-value pairs that describe fields and their contents. A block of data enclosed within curly brackets can contain multiple key-value pairs separated by commas. Values can range from strings to numbers to arrays and even JavaScript objects.

```
[{
    "itemcode": "M001",
    "itemkeywords": ["mobile", "Apple"]
},
{
    "itemcode": "M002",
    "itemkeywords": ["mobile", "Motorola"]
}]
```

Benefits of JSON

In terms of benefits, JSON is easy to understand, and it is parsed faster than XML. JSON is also compact and lightweight when compared to XML, and hence almost every API provider supports JSON as the preferred data interchange format.

Fetching JSON data:

```
var xhr = new XMLHttpRequest();

xhr.open("GET", "json/catalog.json");
xhr.onload = function() {
    var JSONData = JSON.parse(xhr.response);
};

xhr.send();
```

Here we are loading a JSON file *catalog.json*, and once it is loaded in we use `JSON.parse` to convert it into a valid JSON array.

In practical implementation instead of a static JSON file, you would most likely have an API or a server-side script such as those made in PHP to provide data in the JSON format.

jQuery provides a shorthand `$.getJSON` method for easy loading and incorporating JSON in your projects. Once the data is loaded in, the `response` element in the callback function provides access to the data directly, thereby eliminating the need to use the `JSON.parse` method

Populating data using jQuery getJSON

Open *ShoppingCart_Begin.htm* file from the accompanying archive.

We have a simple interface that represents a typical shopping catalog. A JSON file containing data for all these products is provided as well. There is a `div` that represents an item in the shopping catalog and a "Load Catalog" button which we're going to use to trigger the AJAX action to load the catalog in. Once we have the contents of the catalog loaded in, we will iterate through it and display items.

The actual content for our shopping catalog resides in a *json/catalog.json* file. Each item contains key-value pairs, that defines various properties such as the name, price, file name of the thumbnail image, description and so on.

Return to the *.htm* file and comment out the sample item there first. Now code the "Load catalog" button:

```
$('#loadCatalog').on('click', function() {
    $.getJSON("json/catalog.json", function(data) {
        renderCatalog(data);
    });
});
```

Since we already have the `renderCatalog` function that is designed to iterate through the array and populate the user interface, we are simply passing the `data` variable to it from within our callback function.

Go on and check the result!