Ajax with jQuery

Ajax, which stands for *Asynchronous JavaScript and XML*, lets us fetch and send data to and from a server asynchronously, in the background, without interfering with the user’s experience.

In the last chapter, you saw examples of asynchronous behavior. While animations were running, you were able to execute other code, such as changing the element’s background color, and the user (in this case, you) was perfectly able to use the page while the animations were taking place. Nothing seemed different, other than the animation. Fetching data with Ajax is much like that. As a user, you’re unaware of what’s happening until the data has been fetched and then shown on the page.

In this chapter, we’ll thoroughly explore Ajax. Although Ajax stands for *Asynchronous JavaScript and XML,* the most common format for getting data back is now JSON, or JavaScript Object Notation; before you start fetching data, you’ll become familiar with this format. Next, you’ll look at some sample JSON and see how to work with it using JavaScript. Then, you’ll be introduced to jQuery’s `ajax()` method. Finally, you’ll use a real-world third-party API to pull in data and display it on a page. To do this, you’ll need to explore JSONP, a method of requesting data from third-party web sites.

*Ajax* has been somewhat of a buzzword in recent years, but what it actually means can be confusing. It’s simply a way of asynchronously fetching data. That’s it.

**JSON**

The “x” in Ajax may stand for XML, but the format nearly everyone prefers these days is JSON, which stands for *JavaScript Object Notation* ([http://json.org](http://json.org)).

JSON data looks very similar to a regular JavaScript object. Here’s an example:

```json
{
  "name":"Jack Franklin",
  "age":20,
  "location":"London, UK",
  "friends":[
    "Grant",
    "Jamie",
    "Dan",
    "Richard",
    "Alex"
  ]
}
```
There are two important things to note here. The first is that, unlike JavaScript, keys in a JSON object have to have double quotes around them. For example, in JavaScript, all of the following are valid:

```javascript
var jack = { "age": 20 };
var jack = { 'age': 20 };
var jack = { age: 20 };
```

However, only the first line would be valid in JSON. Values in JSON can be of the following types:

- String
- Number
- Array
- Boolean true/false
- null

Items in arrays can be any of these types, too. Strings need to be double quoted, but the rest don’t:

```json
{
    "num":2,
    "decimal":2.5,
    "boolean":true,
    "array":[
        1,
        2,
        3,
        true,
        null
    ]
}
```

Much like JavaScript objects, key value pairs need a comma after them, unless it’s the last pair in the object. Because JSON is newer than JavaScript, not all browsers come with native JS methods for parsing it. There are two main methods:

- `JSON.stringify()`: Takes a JavaScript object and produces a JSON string from it.
- `JSON.parse()`: Takes a JSON string and parses it to a JavaScript object.

The Can I Use? web site (http://caniuse.com/json) is useful for finding out which browsers natively support JSON (see Figure 8-1).