**Key Concepts: Part 2**

**A is for Asynchronous**

The asynchronicity of Ajax catches many new jQuery users off guard. Because Ajax calls are asynchronous by default, the response is not immediately available. Responses can only be handled using a callback. So, for example, the following code will not work:

var response;

$.get('foo.php', function(r) { response = r; });

console.log(response); *// undefined!*

Instead, we need to pass a callback function to our request; this callback will run when the request succeeds, at which point we can access the data that it returned, if any.

$.get('foo.php', function(response) { console.log(response); });

**Same-Origin Policy and JSONP**

In general, Ajax requests are limited to the same protocol (http or https), the same port, and the same domain as the page making the request. This limitation does not apply to scripts that are loaded via jQuery's Ajax methods.

The other exception is requests targeted at a JSONP service on another domain. In the case of JSONP, the provider of the service has agreed to respond to your request with a script that can be loaded into the page using a <script> tag, thus avoiding the same-origin limitation; that script will include the data you requested, wrapped in a callback function you provide.

**Ajax and Firebug**

Firebug (or the Webkit Inspector in Chrome or Safari) is an invaluable tool for working with Ajax requests. You can see Ajax requests as they happen in the Console tab of Firebug (and in the Resources > XHR panel of Webkit Inspector), and you can click on a request to expand it and see details such as the request headers, response headers, response content, and more. If something isn't going as expected with an Ajax request, this is the first place to look to track down what's wrong.