**Key Concepts: Part 1**

Proper use of Ajax-related jQuery methods requires understanding some key concepts first.

**GET vs. Post**

The two most common “methods” for sending a request to a server are GET and POST. It’s important to understand the proper application of each.

The GET method should be used for non-destructive operations — that is, operations where you are only “getting” data from the server, not changing data on the server. For example, a query to a search service might be a GET request. GET requests may be cached by the browser, which can lead to unpredictable behavior if you are not expecting it. GET requests generally send all of their data in a query string.

The POST method should be used for destructive operations — that is, operations where you are changing data on the server. For example, a user saving a blog post should be a POST request. POST requests are generally not cached by the browser; a query string can be part of the URL, but the data tends to be sent separately as post data.

**Data Types**

jQuery generally requires some instruction as to the type of data you expect to get back from an Ajax request; in some cases the data type is specified by the method name, and in other cases it is provided as part of a configuration object. There are several options:

text

For transporting simple strings

html

For transporting blocks of HTML to be placed on the page

script

For adding a new script to the page

json

For transporting JSON-formatted data, which can include strings, arrays, and objects

**Note**

As of jQuery 1.4, if the JSON data sent by your server isn't properly formatted, the request may fail silently. See <http://json.org> for details on properly formatting JSON, but as a general rule, use built-in language methods for generating JSON on the server to avoid syntax issues.

Jsonp

For transporting JSON data from another domain

Xml

For transporting data in a custom XML schema

*You should use the JSON format in most cases, as it provides the most flexibility. It is especially useful for sending both HTML and data at the same time.*