This explanation was found on http://stackoverflow.com/questions/3839966/can-anyone-explain-what-jsonp-is-in-layman-terms. It is a good high level description of JSONP and helps give a good understanding of JSONP.

JSONP (as in "JSON with Padding") is a method commonly used to bypass the cross-domain policies in web browsers (you are not allowed to make AJAX requests to a webpage perceived to be on a different server by the browser).

JSON and JSONP behave differently on both the client and the server. JSONP requests are not dispatched using the XMLHTTPRequest (and the associated browser methods), instead a `<script>`tag is created, whose source is set to the target URL. This script tag is then added to the DOM (normally the `<head>`).

JSON Request:

```
var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function () {
  if (xhr.readyState == 4 && xhr.status == 200) {
    // success
  };
};

xhr.open("GET", "somewhere.php", true);
xhr.send();
```

JSONP Request:

```
var tag = document.createElement("script");
tag.src = 'somewhere_else.php?callback=foo';

document.getElementsByTagName("head")[0].appendChild(tag);
```

The difference between a JSON response and a JSONP response, is that the JSONP response is formulated such that the response object is passed as an argument to a callback function.

JSON:

```
{
    "bar": "baz"
}
```

JSONP:

```
foo({
    "bar": "baz"
});
```

This is why you see JSONP requests containing the "callback" parameter; so the server knows the name of the function to wrap the response around.

This function must exist in the global scope at the time the `<script>` tag is evaluated by the browser (once the request has completed).

Another difference to be aware of between the handling of a JSON response and a JSONP response, is that any parse errors in a JSON response can potentially be caught (by wrapping the attempt to evaluate the responseText in a try/catch statement). Because of the nature of a JSONP response however, parse errors in the response will yield an uncatchable JS Parse error. Both formats however, can implement timeout errors (by setting a timeout before initiating the request, and clearing the timeout in the response handler.

The usefulness of using jQuery to make JSONP requests, is that jQuery does allllllll of the work for you in the background.

jQuery requires (by default), for you to include `&callback=?` in the URL of your AJAX request. jQuery will take the `success` function you specify, assign it a unique name and publish it in the global scope. It will then replace the `?` in `&callback=?` with the name it's just assigned the function.

Comparable JSON/ JSONP Implementations (assuming response object is `{"bar":"baz"}`:

JSON

```
var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function () {
  if (xhr.readyState == 4 && xhr.status == 200) {
    document.getElementById("output").innerHTML = eval('(' + this.responseText +
')').bar;
  };
};

xhr.open("GET", "somewhere.php", true);
xhr.send();
```

JSONP:

```
function foo(response) {
  document.getElementById("output").innerHTML = response.bar;
};

var tag = document.createElement("script");
tag.src = 'somewhere_else.php?callback=foo';

document.getElementsByTagName("head")[0].appendChild(tag);
```