[Difference between @RemoteAction annotation and <apex:actionFunction>?](http://salesforce.stackexchange.com/questions/17375/difference-between-remoteaction-annotation-and-apexactionfunction)

@RemoteAction and apex:actionFunction are essentially polar opposites in terms of functionality, even though they both offer JavaScript functionality.

@RemoteAction methods are static, and therefore can't see the current page state directly, while apex:actionFunction methods are instance methods, and so can see the entire page state.

@RemoteAction methods require less bandwidth, and server processing time, because only the data you submit is visible and the view state is not transferred, while apex:actionFunction has to transfer the page view state.

@RemoteAction methods have to update the DOM manually using explicit JavaScript, while apex:actionFunction methods automatically update the Visualforce DOM and can refresh part or all of the page, and can provide a standard interface for showing a loading status through apex:actionStatus.

@RemoteAction methods can return data directly back to the calling JavaScript, but cannot update the page's view state. apex:actionFunction methods can update the page's view state and DOM structure, but cannot return data directly back to JavaScript (although you can do this with some extra effort using oncomplete).

Because of these polar differences, there is almost always "one right answer" for which method should be used in a given situation. An auto-complete-as-you-type feature would benefit from @RemoteAction, because less data is used, and so it is faster and more responsive; we don't need to modify the view state, so this is a major bonus for the user. Updating the page's view state to show new Visualforce elements, in contrast, would best be served with apex:actionFunction; using a @RemoteAction would force the developer to manage the DOM themselves, which can be more burdensome.

Also, you generally can't modify the DOM in native JavaScript if you use apex:actionFunction, because those changes won't appear in the view state and will be overwritten if the DOM node the changes appeared in is refreshed. This means that JavaScript designed to use @RemoteAction and update the DOM as a result will generally be incompatible with apex:actionFunction. There's exceptions to this rule, of course, as long as both uses are isolated from each other or interact only in a carefully designed order of operations.

Comparing JavaScript Remoting and **<apex:actionFunction>**

The <apex:actionFunction> component also lets you call controller action methods through JavaScript.

In general, <apex:actionFunction> is easier to use and requires less code, while JavaScript remoting offers more flexibility.

Here are some specific differences between the two.

**•** The <apex:actionFunction> tag:

**–** lets you specify rerender targets

**–** submits the form

**–** doesn’t require you to write any JavaScript

**•** JavaScript remoting:

**–** lets you pass parameters

**–** provides a callback

**–** requires you to write some JavaScript

Comparing JavaScript Remoting and Remote Objects

JavaScript Remoting and Remote Objects offer similar features, and both are useful tools for creating dynamic, responsive pages. They have some important differences that you should consider before choosing which to use.

In general, Remote Objects is well-suited to pages that need to perform only simple Create-Read-Update-Delete, or “CRUD”, object access. JavaScript Remoting is better suited to pages that access higher-level server actions. Remote Objects lets you get up and running quickly without a lot of ceremony, while JavaScript Remoting is suited for more complex applications that require some up front API-style design work.

Visualforce Remote Objects:

**•** Makes basic “CRUD” object access easy

**•** Doesn’t require any Apex code

Using JavaScript in Visualforce Pages When to Use JavaScript Remoting

**•** Supports minimal server-side application logic

**•** Doesn’t provide automatic relationship traversals; you must look up related objects yourself

JavaScript Remoting:

**•** Requires both JavaScript and Apex code

**•** Supports complex server-side application logic

**•** Handles complex object relationships better

**•** Uses network connections (even) more efficiently