# How Injecting JavaScript into Casus works

The first part is an include file that Casus reads as part of the page. It is a simple file and includes a <div> where the document is to be attached, and a script line to get the whole thing rolling. Here are the contents of the file to include adding cards to questions:

<div id="card\_questions"></div>

<script src="http://cryptic-bastion-5586.herokuapp.com/card\_questions/init?cardid=<PMW cardid>&userid=<PMW userid>" type="text/javascript"></script>

The <PMW…> tags are Casus specific tags for passing information like the card id. In theory the include file can be any html, etc. but so far all we have done is pass a <div> and some script.

When the include file is read a request is sent to the **src** url. The request is parsed and a decision is made on whether or not to proceed. The decision is usually based on something like the cardid being passed.

If the card is of interest then a text file is read which is more JavaScript, any necessary substitutions are made and the request return is rendered as JavaScript.

The script being returned creates an <iframe> which attaches itself to the <div> in the include file. When the <iframe> is loaded a request to the url in the **src** property of the <iframe> is generated. That request creates an html document and loads in into the <iframe>.

The include file can be found locally in Projects/casus\_includes. It is also in github where we share it with Martin. If we want to have Martin pick up changes then we need to update the copy in github.

1/6/13

Added uri encoding and decoding of the card and case names. It’s all done in the controller so that the URI encoding is consistent. Found and instance where the <%= %> template in erb file automatically encoded some quotes in a string. The encoding was &quot; which doesn’t match uri encoding. So we keep it all on the controller side.