How To

How To Write Safe Widgets

Developing Widgets in this framework can be easy, but it's worth taking a few moments to consider the implications of writing a Widget, that will make your widget far more robust and reliable. The How To pages will help you to get started quickly, but before you dive in here are some things to think about. More detail can be found on the Widget Best Practice page. To learn how to write your first widget please see Widget Development Manual - How to write a widget successfully.

Widget Structure

Widgets are HTML fragments, normally loaded from static HTML files hosted on a Web server (eg Apache). The Widget page that has been registered with the portal will almost certainly reference other javascript and css files. It is important to remember that there will only be one copy of your widget on a page, but it will share that same browser DOM and javascript context as all the other widgets on that page. All the widgets will share this all with the portal page that contains them. This means that you have to good to your neighbors.

Markup

Your markup should be able to co-exist with other markup. This is quite easy to achieve. You need to make sure that the widget that you produce is valid XML, since the portal page into which it will be inserted will be valid XHTML. You also need to be certain that the markup in your widget does not conflict with markup from other widgets. So where you use ID's they must be unique to all pages and all widgets. It makes sense to select a prefix to all your widgets and use that as an ID prefix. eg a FileManager widget might use filemanager_ as a prefix eg <div id="filemanager_outer" >. To recap, the base requirements are

- Make certain your HTML fragment is valid XML.
- Prefix all ID's

CSS

Your css should be limited to what is required for you widget and should use css from the parent page as far as possible. You should not put any in line formatting tags, and should avoid in line styles. It will also make your life easier if you scope your css. If you surround your widget markup with a <div class="filemanager" > then your css can locate div.filemanager to restrict the scope of all your css. In general you should not mess with any html tags outside your own widget and certainly not mess with any HTML tags outside your widget space. To recap, if you don't want to be fixing your css all the time.

- Scope your CSS with a containing div around all the widget markup
- Don't supply css outside your scoped space

Javascript

Javascript offers the greatest opportunity to come unstuck. Do not use global variables in you Widget. This will probably cause havoc with your own widget and will cause problems with other widgets that have made the same mistake. You must encapsulate your javascript within name spaces that are unique to your widget so that you have full control over all you javascript, and so that you don't pollute the global space with too many variables. You may want to think as this as your public interface the javascript.

- Don't use global variables exect for your global namespace
- Do namespace all your javascript

More Information

The above is only a brief introduction to safe widget writer behaviour. If none of this makes any sense to you there are more extensive examples on Widget Best Practice page.