

1blankspace jQuery Based Interface Design 1.4

Introduction

The following document covers the standard/default 1blankspace interface, which for some users may be the only way they interact with the system, for others it may just their "getting started" interface, with customisation for their particular needs

In all cases the default interface needs to support:

Different ways of seeking information:

- Search
- Navigate
- Browse

Have the following core attributes:

- Consistent
- Extensible
- Always in context
- Forward facing
- Reusable / leverage able

The interface depends on:

- Web-browser with Javascript Enabled
- jQuery (<http://jquery.com>) / jQueryUI (<http://jqueryui.com>)

Important Notes (Read First!)

1. If you are planning to extend/develop this interface, it is highly recommended that you use Firefox with the Firebug "Add-On" - it can substantially increase productivity.

Extensibility

The system needs to be extensible ensuring it can cater for "100%" of the business needs rather than almost cover their needs eg "95%".

Example request for "CRM":

1. Need to store contacts
2. Manage activity, pre and post sale
3. Do a quote using custom spreadsheet

Closed framework:

- 1 & 2: OK
- 3: say no or go deep into the platform layers - expensive, disruptive and slow to implement.

Open interface framework:

- 1 & 2: OK
- 3: handled within a secure website / web page and constructed using jQueryUI/jQuery/Javascript/onDemand (for persistence). Then blended into the relevant viewport (eg Opportunities).

Consistent

Consistency is important in any relationship involving a human, both at human-to-human and human to machine level.

This framework is based on single page interface for consistency, with two high-level viewport control layers.

1. Master Viewport Control

Controls the "movement/sliding" of the viewport at the highest level.

eg, between:

- Contacts
- Projects
- Actions
- Financials
- etc

2. In-Viewpost Behaviour Control

Controls the behaviour and control of sub-viewports with in the master viewport - based on the css "class" of the elements.

eg a button with the class of "save" will look like a button based on jQueryUI/css and also do something when clicked eg make the data on the page persistent, by sending an onDemand method command eg CONTACT_PERSON_MANGE


Sub-viewports have been traditionally referred to as "tabs".

css Classes

Typically in website methodology to date the css class of an object has controlled how it looked. Through the use of jQuery and its class selectors, it can now also controls how it behaves.

ie a particular class of human looks and behaves in a certain way, eg a soldier. Within reason they are consistent and have certain command actions/behaviours.

Layers

Ref	Layer	Description
	Human	Everyone different and constantly changing. Emotional.
1	Interface	<p>Responsible for engaging the human and servicing their needs.</p> <p>Document and division (DIV) driven – managing viewports and mashing in other frameworks as required (eg Google mapping etc)</p>
2	Interface Support Webservices Endpoints	<p>Works on behalf of the core application layer to meet the needs of the interface layer – a “broker”.</p> <p>Webservice endpoints servicing http GETs and POSTs. Responding with plain text, JSON & XML.</p>

User Interface

Ref	Layer	Description
1	Layout	XHTML
2	Presentation	CSS/jqueryUI
3	Construction	jQuery
4	Behaviour / Interaction	Javascript
5	Information Management	Search and Save, Persistence AJAX/onDemand

The following sections work through layers 1 to 4.

Layout Layer

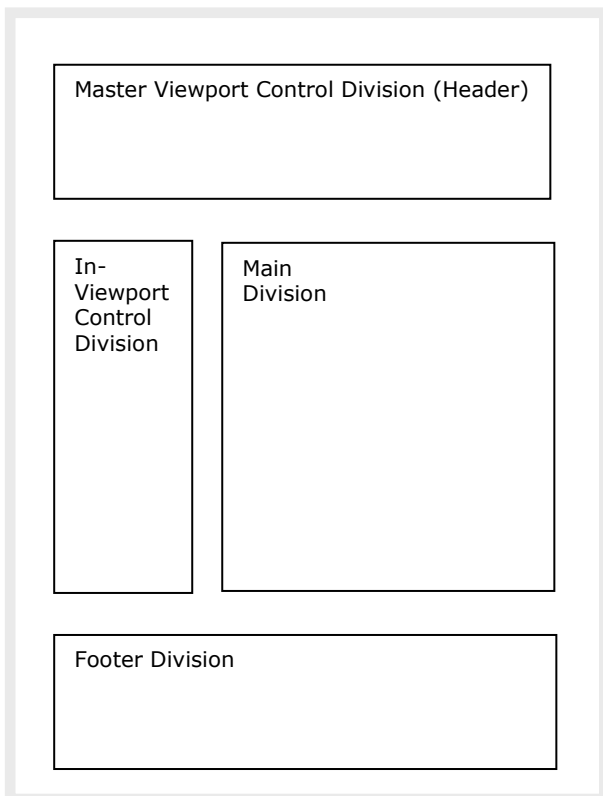
Overall layout is managed in a single XHTML based document/page eg the "home" page. It is never refreshed as such – but instead hosts a series of divisions "divs" that are refreshed. These divs are like the frames used in the existing system.

Each division element can be controlled via the other layers independent of each other. The position of the divs is abstracted from the other layers; within reason they can be moved around the document with no effect on the other interface layers.

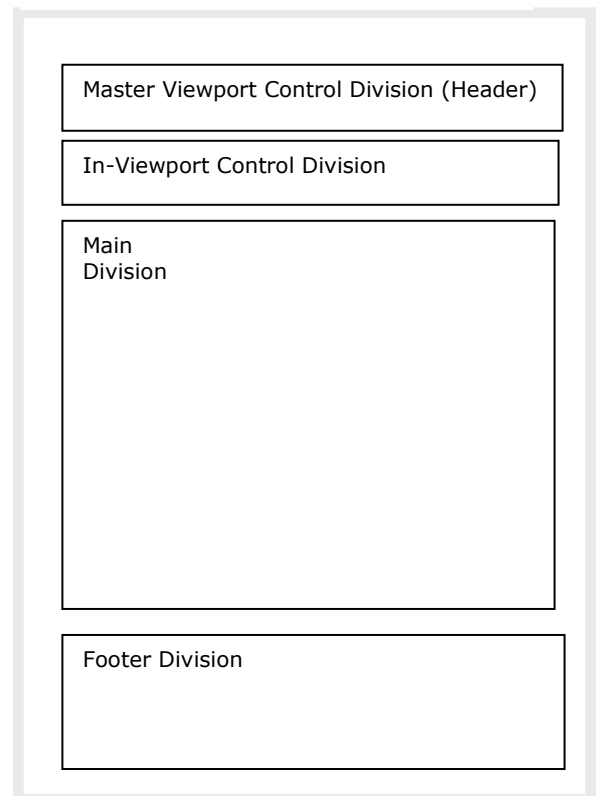
The document ("home page") will vary depending on the device layer/framework it is being hosted in eg if sitting inside a iPhone device layer then the divs will be arranged differently to support it's physical viewport, but a lot the resources in the other layers wont be effected and can be leveraged and reused.

eg

Webbrowser



iPhone



Presentation Layer

The presentation layer is controlled via Cascading Style Sheets (css).

The css layer being created via jQueryUI.com or other creative parties.

The css layer classing is also used by the behaviour/interaction layer to control how the elements react to user triggered events – eg when they click it – being the most common.

Construction/Elements Layer

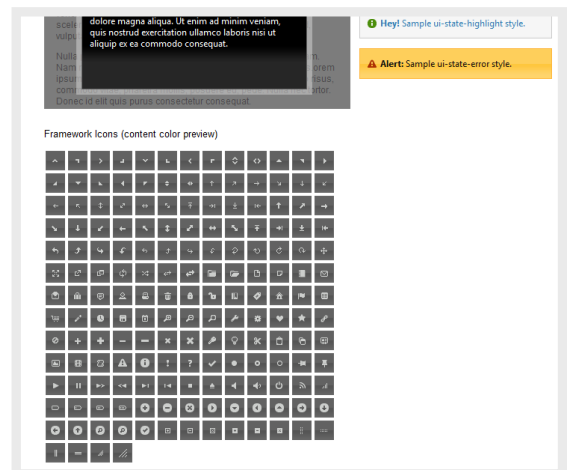
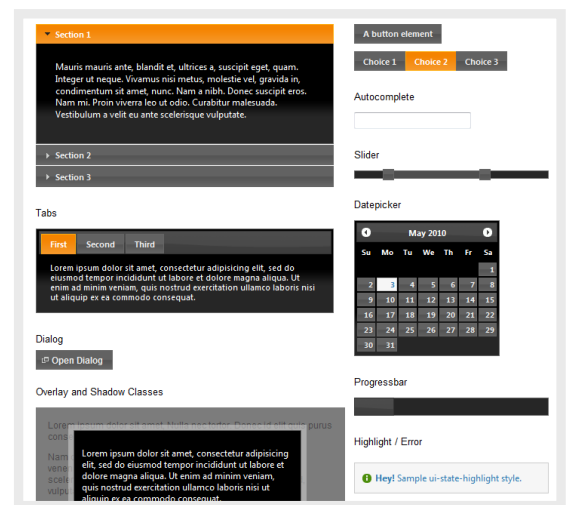
This layer is based around jQueryUI.com foundation elements.

eg textbox, dropdown box, datapicker, progressbar, slider, tabs, accordion etc...

jQueryUI.com also supports the ThemeRoller – that allows you to change the colour scheme, via css.

eg the example image to the right shows a dark theme – but could just as easily be a light and white layer.

The “look and feel” is not controlled in this layer as such, it is in presentation layer and thus controlled by css.



Behaviour/Interaction Layer

This is the layer that takes it from being static page to a dynamic system.

Building the viewports within the XHTML layout, CSS look and feel and jQueryUI elements and reacting to events based on human interaction and also sending requests to the interface support layer eg onDemand webservice end points.

It is the "worker".

This layer/"worker" relies heavily on jQuery (a Javascript based library of functions) to get its work done. eg updating divs, managing human interaction (eg click) and system events (eg new email) and using the onDemand layer to interact with the core platform.

Broker

In classic "IT speak" this is seen as the broker layer – sitting between the human and the core systems.

Ensuring that the needs of the human are met in the confines of the system they are interacting with.

Example Interface

Remember due to the layering of the framework, the look and feel of the human interface can be changed more easily than ever before.

This is just an example based on research into the leading interfaces currently in the marketplace (picking out the best bits), human interface design literature and covering the 3 core types of seeking information based around personal behaviours; the particular viewport or how the person is feeling at the time.

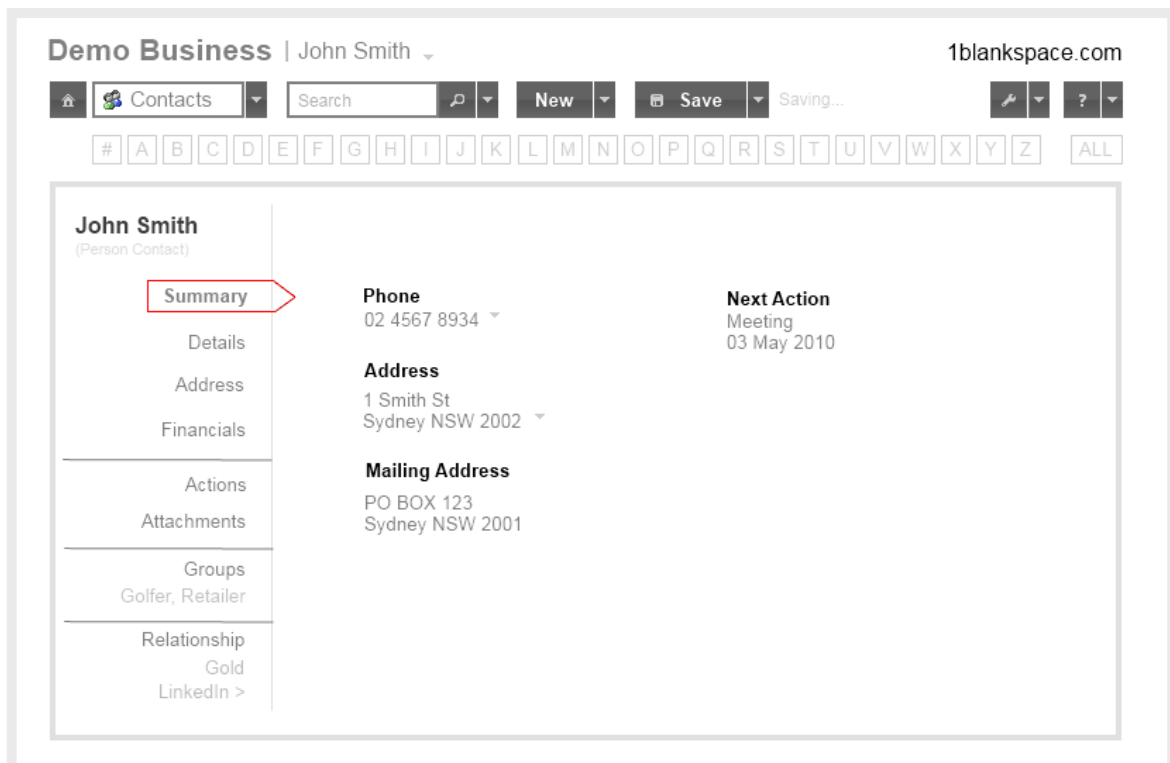
eg

The same person may:

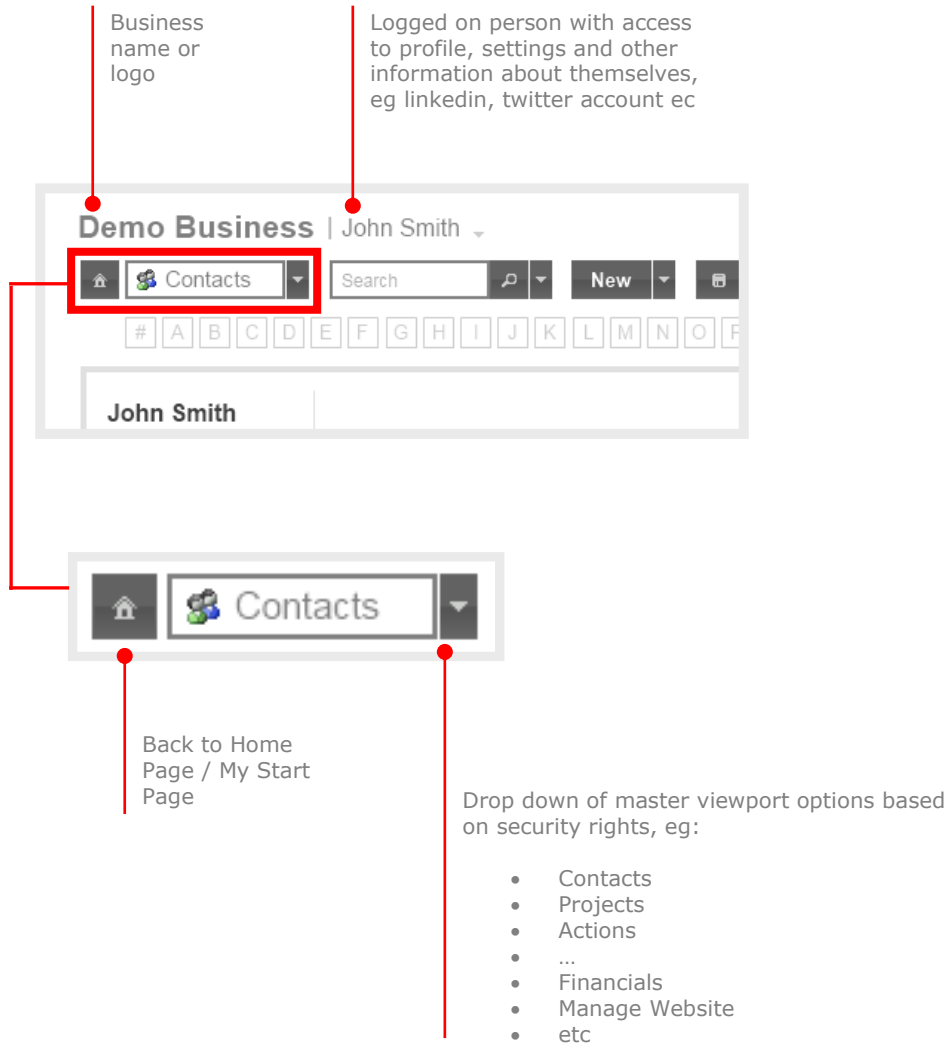
- like to do search for contacts eg type "john"
- like to look for projects by drilling down a "tree"
- like to look for invoices by clicking business name in "A|B|C|D..." list.
- or a Friday afternoon because they are sick of typing when looking for "john" click in the "A|B|C|D..." list OR open up their iPhone...

Who knows, the point is the human interface can engage them no matter who they are and how they feel and thus behave.

Conceptual layout as way of illustration, with minimal colourisation / presentation layer.



Master Viewport Control



In-Viewport Control

Core Behaviours / Actions



Browse for information eg Contacts.
Click A shows a list of businesses and people starting with A.



Search with
autofill after 3
letters.

Access to advanced
search and also
browse by tree etc.

In context new, with
remember last. eg If
in Contacts master
viewport then New
Person / Business; if
financials: Invoice,
Expense, Payment...

Action button in
context of the form
– eg Save, Delete,
Print, Copy, Email
etc

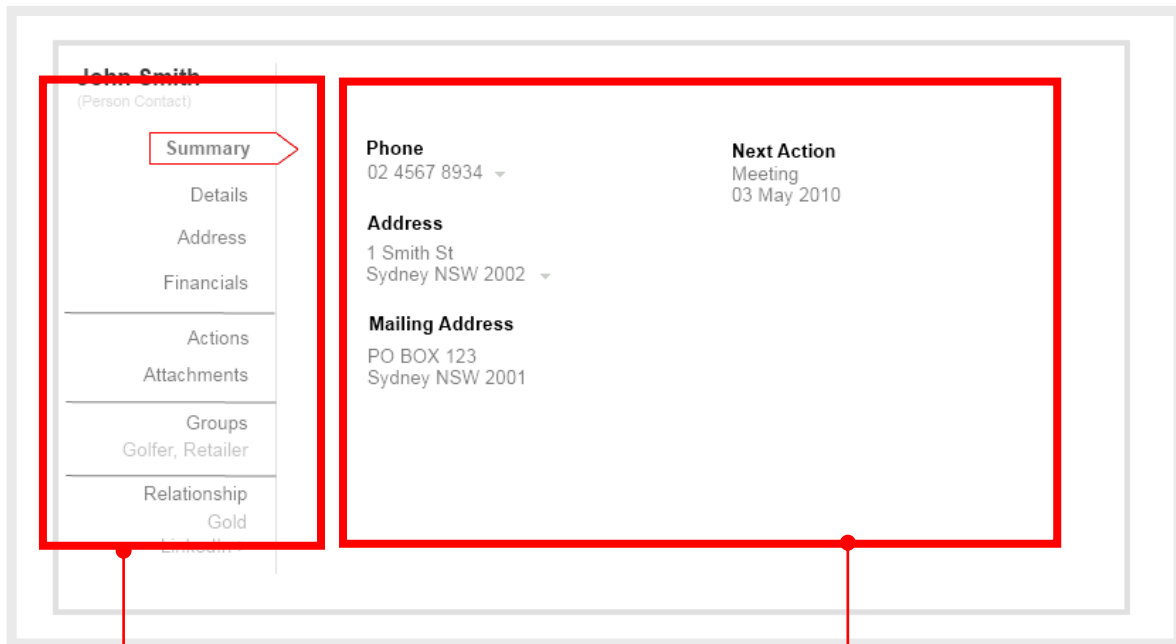


If system administrator, access to set up in context
of this viewport and generally. Could change to
wrench icon. As traditional cog is in the Action
button now.

Access to help – including
in-situate video and access
to log issue, get help from
the network/directory etc

Sub-viewports

This controls the information in the main div and also the behaviour actions.



Used to control the main division and also other key bits of information – eg when viewing a contact which groups they are in.

Could also be used to show getting started tips for new users, with links to videos and if system admin – links to configuration.

The main division, used for presentation of information and editing depending on the viewport – eg in the initial setting of the master viewport – this can show dashboards and give feedback.

Statistics

Source: <http://www.w3counter.com/globalstats.php>

Screen Resolutions	%
1024x768	24.44%
1280x800	18.94%
1280x1024	10.32%
1440x900	8.13%
1366x768	5.98%
1680x1050	5.14%
800x600	4.12%
1024x1024	2.47%
1152x864	2.18%
1920x1080	2.06%

Notes:

- 800x600 has truly now become the exception.
- Minimum screen widths 1024px.
- 50.11% have a minimum greater than 1280px.

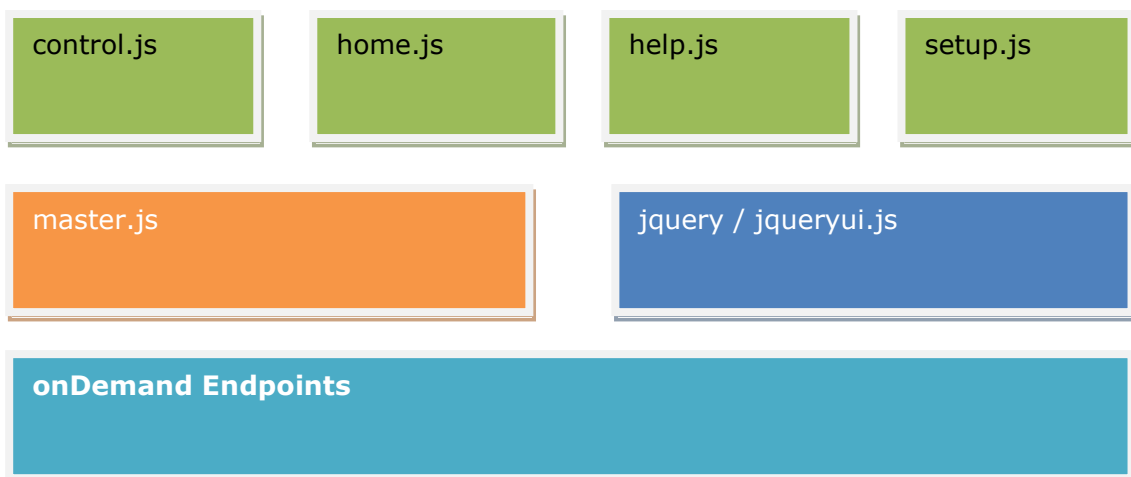
Construction Methodology

The story of construction, layer by layer.

The Design What the space is to be used for.		
1	<p>The Foundation Work</p> <p>Four foundation divs are set up which are the fundamental building blocks of the default interface.</p> <p>They are held in one document, which is effectively the application.</p> <p>The position of the divs is managed within the interface.css.</p> <p>This XHTML is available by inserting the "Website - Web Application Foundation" dynamic tag.</p>	<p>XHTML Document "Home Page"</p> <pre>divInterfaceMasterViewportControl divInterfaceViewportControl divInterfaceMain divInterfaceFooter divInterfaceBox divInterfaceDialog</pre>
2	<p>The Frame Work</p> <p>Next step is setting up the frame work (divs) that sit on top of the foundation and the standard construction elements to be used through the application space (eg drawing of browse bar).</p>	<p>master.js</p>
3	<p>The Fit Out</p> <p>How it looks, including the location of the foundation and framework elements.</p>	<p>format.css</p>
4	<p>The Master Controls</p> <p>Setting up the master controls (eg buttons, search boxes) for the application space, eg the "steering wheel, pedals etc"</p>	<p>master.js</p>

5	The Machinery <p>The guts of the application space, the mechanics of what happens when humans interact with the application space. ie how it behaves. eg what happens when they “turn the steering wheel”.</p>	<p>home.js - function for the home page XHTML</p> <p>control.js – function for the viewport set XHTML</p> <p>setup.js – function for the system configuration/setup ge “My System Profile” in existing interface.</p> <p>help.js for managing links to help resources/website.</p> <p>contacts.js etc</p>
<p>The Final “Fix” / Completion</p> <p>The configuration of the space, who has access, code tables etc</p> <p>Allowing it to become habitable/usable by humans.</p>		

File Relationships



Foundation / Frame Work Element Reference

Foundation		
	Element ID (eg id=)	Description
1	divInterface	Container for foundation divs.
2	divInterfaceMasterViewportControl	"Header"
3	divInterfaceViewportControl	"Side menu"
4	divInterfaceMain	The main user interaction zone
5	divInterfaceFooter	"Footer"
6	divInterfaceBox	For popups eg logon etc

As rendered using dynamic tag:

```
<div id="divInterface" class="interface">
  <div id="divInterfaceMasterViewportControl" class="interfaceMasterViewportControl"></div>
  <div id="divInterfaceViewportControl" class="interfaceViewportControl"></div>
  <div id="divInterfaceMain" class="interfaceMain"></div>
  <div id="divInterfaceFooter" class="interfaceFooter"></div>
  <div id="divInterfaceBox" class="interfaceBox"></div>
</div>
```

Framework (master.js)		
	Element ID (eg id=)	Class
5	divInterfaceMasterViewportSpaceName	interfaceMasterViewport
6	divInterfaceMasterViewportLogonName	interfaceMasterViewport
7	divInterfaceMasterViewportSite	interfaceMasterViewport
8	divInterfaceMasterViewportControl	interfaceMasterViewport
9	divInterfaceMasterViewportControlHome spanInterfaceMasterViewportControlHome spanInterfaceMasterViewportControlHomeOption	interfaceMasterViewport
10	divInterfaceMasterViewportControlSet	interfaceMasterViewport
11	divInterfaceMasterViewportControlSearch inputInterfaceMasterViewportControlSearch	interfaceMasterViewport
12	divInterfaceMasterViewportControlNew spanInterfaceMasterViewportControlNew spanInterfaceMasterViewportControlNewOption	interfaceMasterViewport
13	divInterfaceMasterViewportControlAction spanInterfaceMasterViewportControlAction spanInterfaceMasterViewportControlActionOption	interfaceMasterViewport
14	divInterfaceMasterViewportControlActionStatus	interfaceMasterViewport
15	divInterfaceMasterViewportControlSetup spanInterfaceMasterViewportControlSetup	interfaceMasterViewport
16	divInterfaceMasterViewportControlHelp spanInterfaceMasterViewportControlHelp	interfaceMasterViewport
17	divInterfaceMasterViewportControlBrowse	interfaceMasterViewport
18	divInterfaceMasterViewportControlOptions Floating div used by other elements to present dynamically created HTML.	interfaceMasterViewportOptions

Interface Presentation Layer Layout

master.css defaults styles:

```
div, span, td
{
  font-family:Verdana, Arial, Helvetica, sans-serif;
  font-size:10px;
  position: static;
}

#divInterface
{
  position: relative;
  width: 800px;
}

#divInterfaceViewportControl
{
  position: absolute;
  width: 200px;
  left: 0px;
  top: 100px;
}

#divInterfaceMain
{
  position: absolute;
  width: 600px;
  left: 210px;
  top: 100px;
}

#divInterfaceFooter
{
  position: absolute;
  width: 800px;
  left: 0px;
  bottom: 25px;
}

#divInterfaceMasterViewport
{
  position: relative;
  width: 800px;
}

#divInterfaceMasterViewportLogonName
{
  position:absolute;
  top: 0;
  right: 0;
}

#divInterfaceMasterViewportControl
{
  position: absolute;
  top: 35px;
  width: 800px;
}
```

```
#divInterfaceMasterViewportControlHome
{
  position: absolute;
  top: 0px;
  left: 0px;
}

#divInterfaceMasterViewportControlSet
{
  position: absolute;
  top: 0px;
  left: 45px;
  width: 175px;
}

#divInterfaceMasterViewportControlSearch
{
  position: absolute;
  top: 1px;
  left: 230px;
  width: 150px;
}

#divInterfaceMasterViewportControlNew
{
  position: absolute;
  top: 0px;
  left: 387px;
}

#divInterfaceMasterViewportControlAction
{
  position: absolute;
  top: 0px;
  left: 450px;
}

#divInterfaceMasterViewportControlActionStatus
{
  position: absolute;
  top: 4px;
  left: 525px;
}

#divInterfaceMasterViewportControlSetup
{
  position: absolute;
  top: 0px;
  right: 30px;
}

#divInterfaceMasterViewportControlHelp
{
  position: absolute;
  top: 0px;
  right: 0px;
}

#divInterfaceMasterViewportControlBrowse
{
  position: absolute;
  top: 0px;
  left: 45px;
}

#divInterfaceMasterViewportControlOptions
{
  position: relative;
  width: 100px;
  z-index: 1;
}
```

The Frame Work In Depth

1	Master Viewport Main application control. Master divisions divs Behaviour/mechanics for home and viewport set controls Show home page.	Depends on home.js for the content within the home page
2	Viewport Set Options Moving to a different viewport within the application framework. When user clicks on option, calls the viewportMasterContacts.js	Dependio n control.js to get the content.

Using Contacts As Example		
3	<p>Viewport Contacts</p> <pre>viewportMasterContacts()</pre> <p>Set the behavior/mechanics on the master controls.</p> <pre>searchContacts() searchOptionsContacts() newContacts() newOptionsContacts() saveContacts() saveOptionsContacts() setupContacts() setupOptionsContacts() helpContacts() helpOptionsContacts()</pre> <p>Set global object context. Set the first page / dashboard.</p>	contacts.js
4	<p>Search For Contact</p> <p>Using the master control search input box and div is displayed with matching people and businsses.</p> <pre>searchContacts()</pre> <p>This function then calls <code>interfaceShowContact()</code></p>	

5	View Contact <code>viewportContacts()</code> Add the elements to the Viewport Control (eg "side menu"). <code>divInterfaceViewportControl</code> Divide up the main division for the submenu options. <code>divInterfaceViewportMainSummary</code> <code>divInterfaceViewportMainDetails</code> <code>divInterfaceViewportMainAddress</code> <code>divInterfaceViewportMainFinancials</code> <code>etc</code> Shows the summary information	<code>contacts.js</code>
6	Save Contact Checking to see if div has been set and if so – send the values back for saving. eg make persistent.	