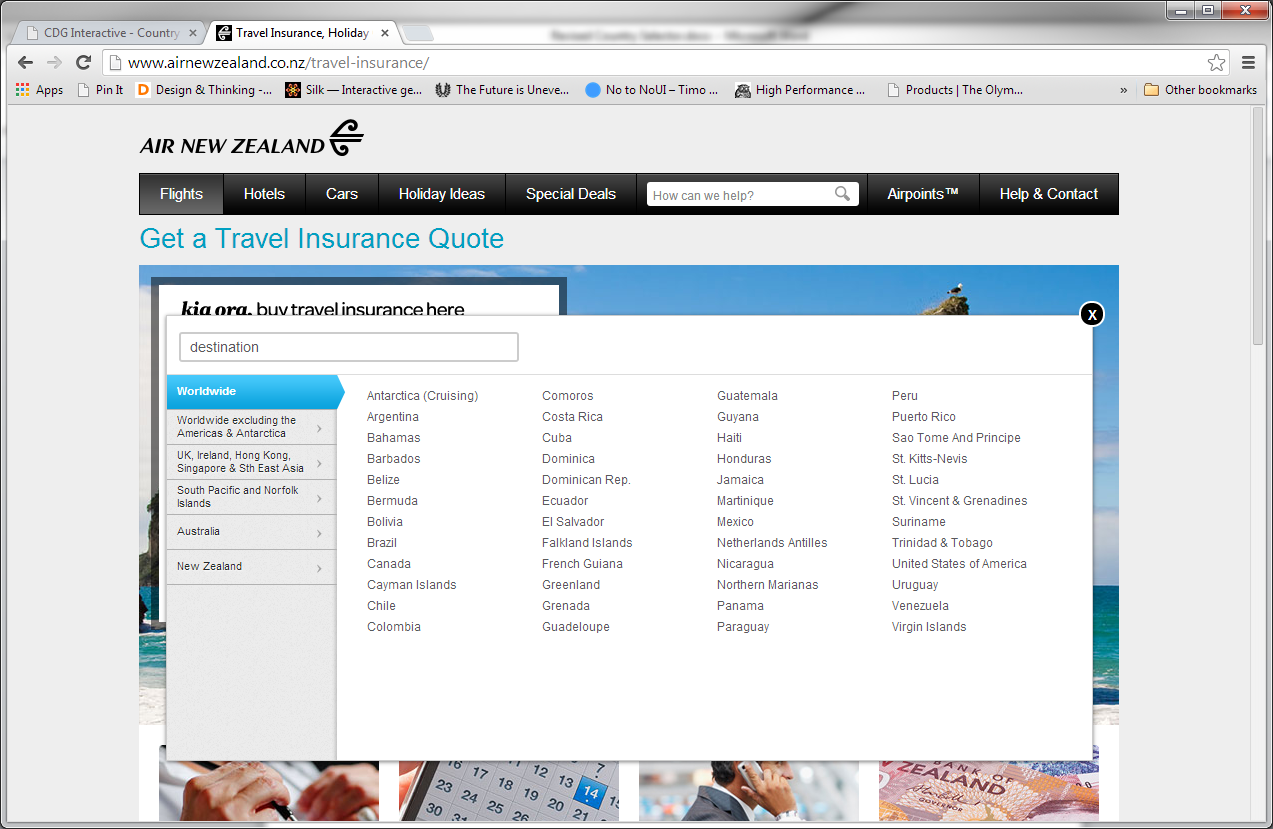
# Revised Country Selector



This is the current country selector for Air NZ and base for the version used by Cover-More/IAL.

We used this UI to match the one currently used on Air NZ homepage. Interesting UI but caused problems since it was not designed to be responsive (screens smaller than laptop/desktop) and we had ended up forking the pages into two versions for support.

Another point against this UI is the lack of accessibility. Basically, this version has no real form elements [radio/checkboxes/etc.] that would make it usable without major JavaScript assistance to select and pass on form values.

## Desktop

Starting out, I started with accessibility. Currently, the country selector is simply, without visuals, a list of countries.

<div class="group">

<div class="group-label region-1"><span>**(Region Name)**</span></div>

<div class="columns">

<ul class="column">

<li id="(Region ID):(Country ID)"> (Country Name)</li>

…  
 etc.

</ul>

</div>

</div>

Originally, there were links but I removed those and streamlined it to just list items. The same JavaScript was refined and adjusted accordingly for the revised code. However, this whole collection of links/text had no semantic value. Therefore, this is the first thing I did was to rewrite the code to be functional, semantic, and easily styled.

I started with two select boxes but that did not work because of the difficulty in styling select boxes/options with CSS (select boxes are one of the hardest elements to style next to file inputs). I decided to use the combination of labels and radio buttons styled into buttons that allowed for data input with a single click/tap. We’ve used this technique in other projects recently (youGo, Claims, etc.). In this case, I rewrote the list of countries into a functional collection of radio buttons/labels, styled by CSS.

<fieldset class="group">

<legend>Region – (Name of Region)</legend>

<input type="radio" name="regionID" id="region-(Region ID)" value="(Region ID)">

<label for=" region-(Region ID )" class="group-label region-1"><strong>**(Region Name)**</strong></label>

<div class="columns">

<ul class="column">

<li>

<input type="radio" name="destinationCountryID" id="country-(Country ID)" value="(Country ID)">

<label for="country-(Country ID)">(Country Name)</label>

</li>

…  
 etc.

</ul>

</div>

</fieldset>

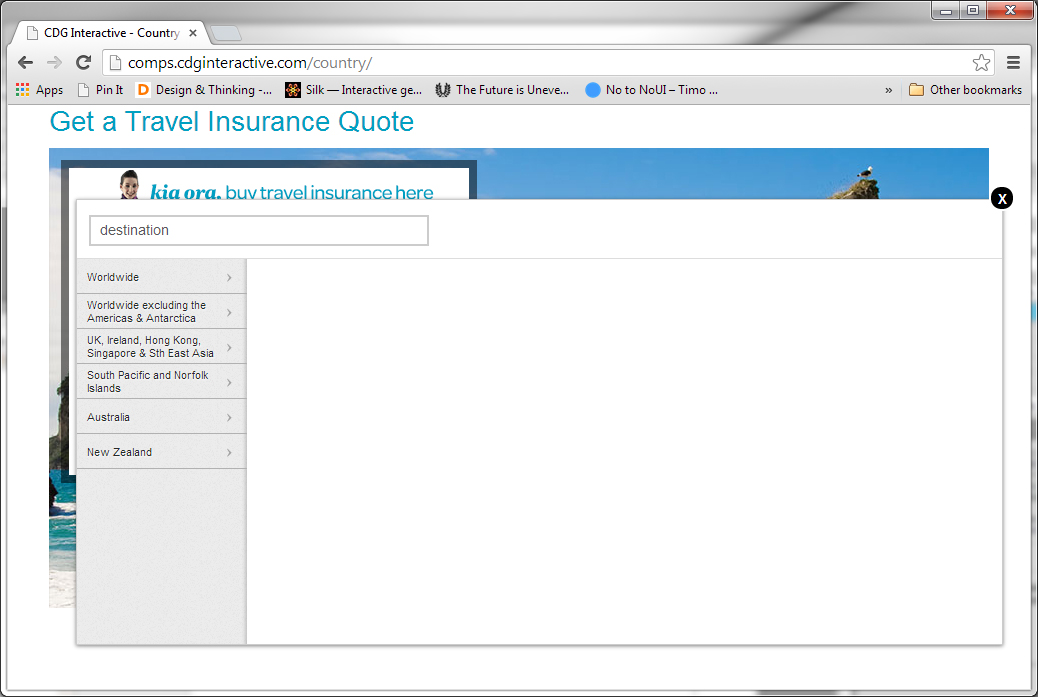
In the process, country/region selection became more straight forward (one group of radio buttons for region, and the other group for country) reducing the amount of JavaScript for data entry. To go even further, using CSS only

input[type="radio"] + label ~ div.columns **(Close Panel)**

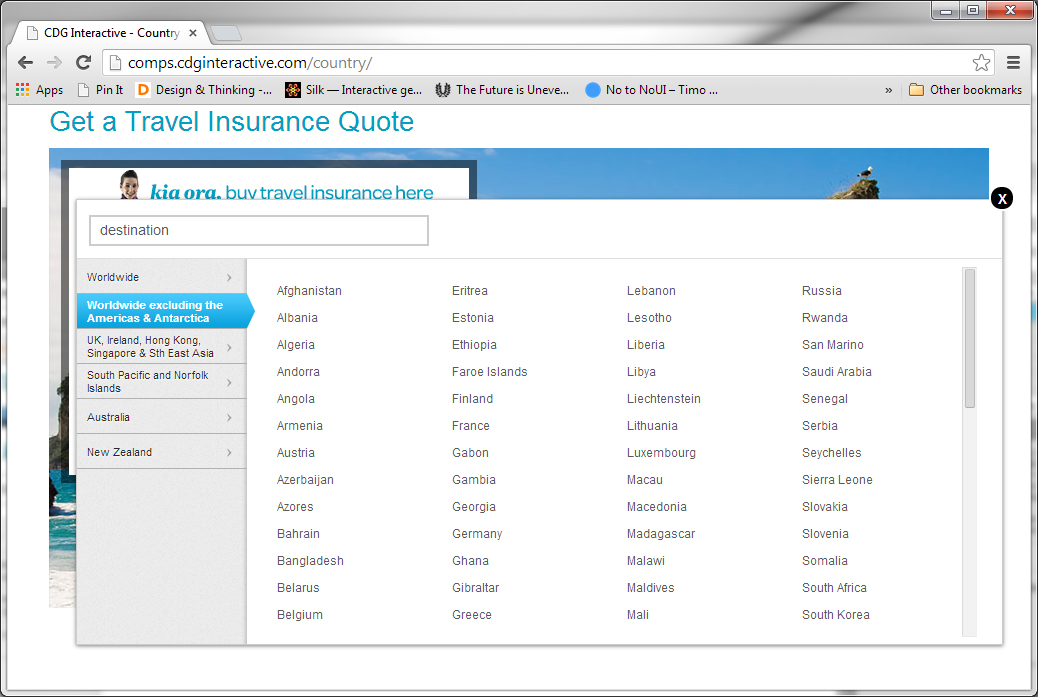
input[type="radio"]:checked + label ~ div.columns **(Open Panel, when selecting region)**

the use of CSS adjacent selectors (CSS3), allows for reducing the JS need further by displaying each panel after selecting a region. Unlike the current version, there is no hover state. The user will need to actively select a region, which is consistent with behavior on touch-enabled devices already.

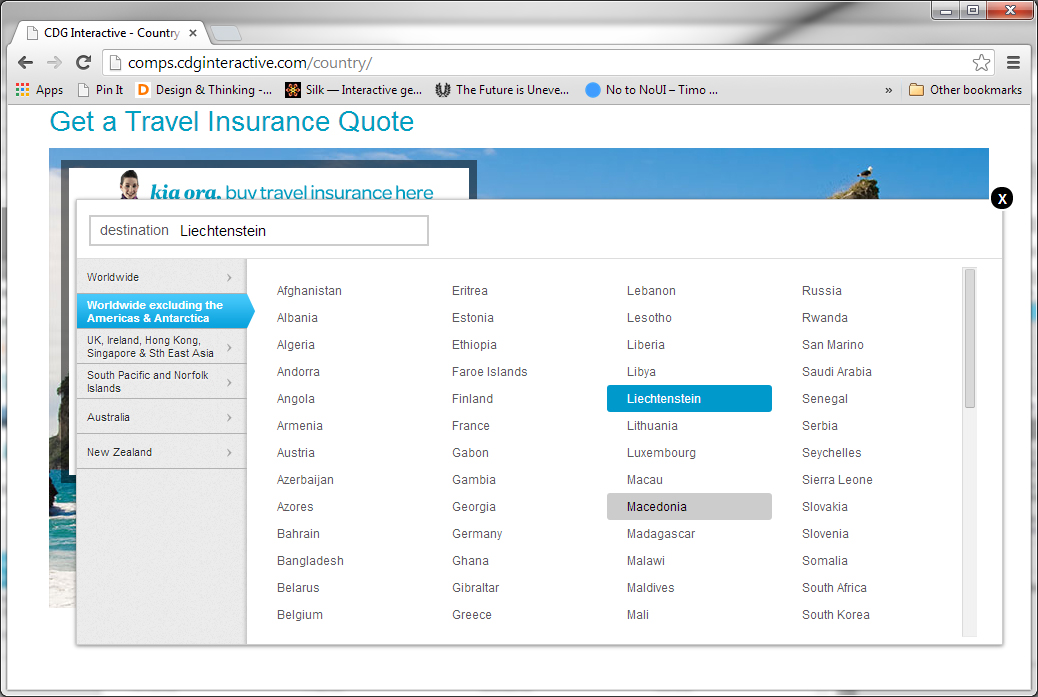
At this point, the only JavaScript required is the opening/closing of the country selector via buttons/links.



**Opened View – regions only**



**Opened view – region selected, opening list of selected country**



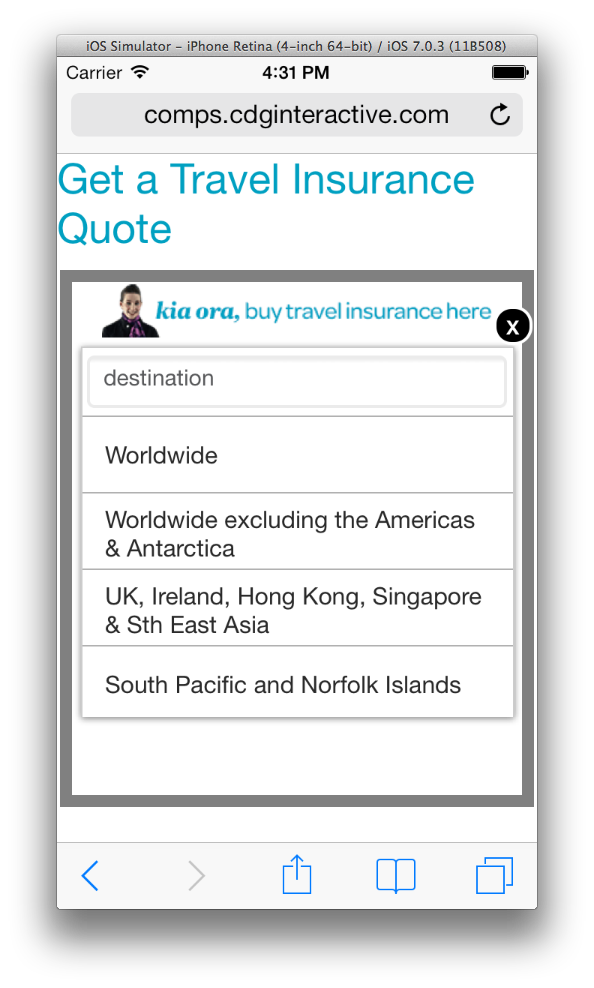
**Country selected – selected choice (same colors as the desktop version). Gray is hover state.**

## Small Screen (Phone/Tablet)

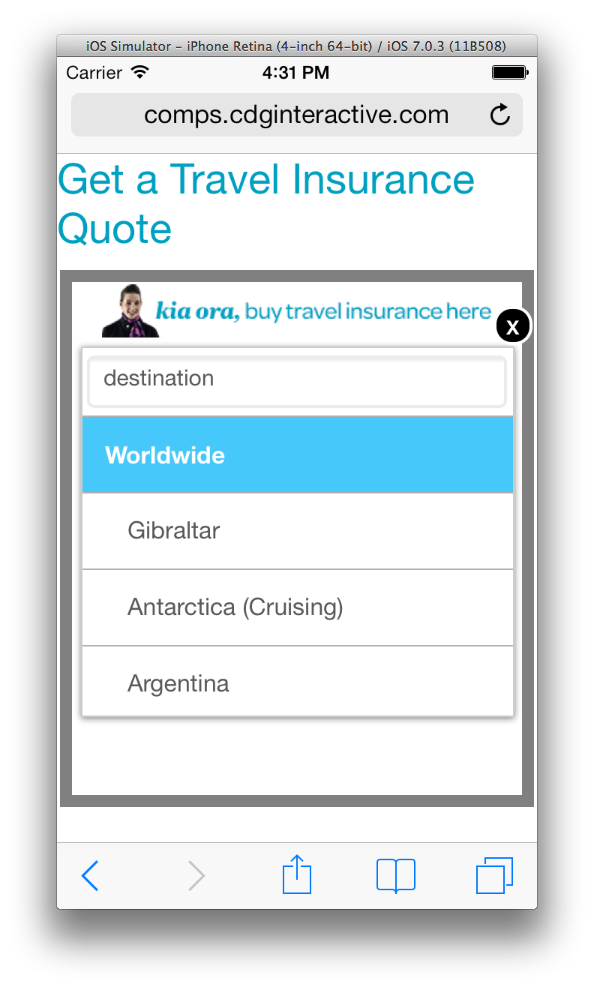
With a configurable CSS, it was now possible to create a small screen version for phone/tablet. With the reduced space, the only option is to have the regions/countries stacked on top each other. This is similar to a native select menu but different in the following:

1. Options are fully displayed, no cutting off or ellpesis (…) for overflow
2. If region has not been selected, all countries under that region are hidden, saving as much space/scrolling area as possible

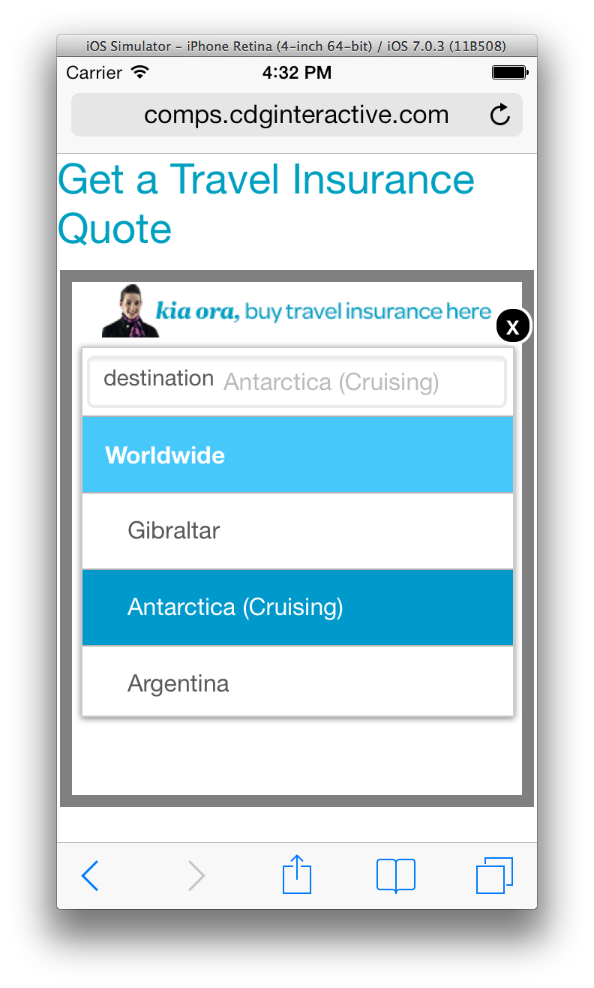
Still keeping the panel open/close functionality, the result is the following views:



**Opened view – regions only**



**Opened view – region selected, opening list of selected country  
Two different color background differentiate selected regions with related countries**



**Country selected – selected choice (same colors as the desktop version).**

## Problems/Solutions

At this point, I went through and tested the solution on desktop/mobile browsers and observed how this worked on various devices. During the process, found various problems and their solutions.

1. Both iOS mobile/Android had inconsistent behavior with CSS3 adjacent selectors. This showed up immediately with the opening/closing of each panel. After some trial and research, found the code that works for all (See above with open/close panel code)
2. Android, unfortunately, has a bug that exists even in the newest version (4.4/KitKat). I was able to find a CSS hack to have it work as well.   
     
   /\* Android Bug Fix - Country Selector/Radio/Labels \*/  
   .android .columns, .android-2 .columns { -webkit-animation: bugfix infinite 1s; }  
   @-webkit-keyframes bugfix { from { fill: 0; } to { fill: 0; } }
3. Empty spacing problem (Firefox) – first encountered when testing mobile screen view on IE desktop, found it was occurring on Firefox mobile (both). Basically, large amount of empty space after selecting region, as if the height was not readjusted. Corrected problem by changing hiding method (clip instead of left with absolute positioning)

### Android 2.3

Everything else works but the old problem of non-functioning scrolling divs remains (Cover-More app Checklist). This was fixed with selective application (Javascript OS detection) of iScroll plugin. Additional code was added for overcome some deficiencies.

The UI was tested on Android Mobile (WebKit stock browser), Firefox, and Opera.

This OS is only available on older/less powerful phones only, not on tablets.

### Legacy(IE8)

This browser does not support CSS attribute selectors and required additional JS for selection display.

## Other Differences

1. In the original, Air NZ added the auto complete functionality (jQuery UI plugin). That went through an initial install and testing but I decided not to keep it in the final revised UI. The plugin was not made for mobile (touch), especially in Android 2.3 (another scrolling div which required modifications).
2. The input field originally used for country display has been replaced with a span for the following reasons:
   1. Bigger touch area – needed for mobile and did not have to deal with default behavior of input fields (cursor, text keyboard)
   2. The button in the current version is now a background (just an indicator for the user to touch the larger interface)
   3. Inner text have semantic meaning – removing the input field, restructured inner text to  
      <em>destination</em><span>(Country Name)</span>
3. Unlike the original, this has a preset state (country/region initially selected)
4. Hover states still exists for regions but for desktop only and only for visual clues