**Allowing Graceful degradation and being band­width-conscientious**

1. First, it locates the div.slides element inside of the .welcome module, using jQuery’s very CSS-friendly selec­tor syntax. ($(".welcome .slides")).

2. Once it’s located that element, it wraps the contents with some mark-up that’s required by the carousel script. (.wrapInner(…))

3. With that new HTML in place, our script runs the .carousel() function, creating the slideshow. And since we’ve marked up individual slides with the .figure class, that’s what we’ve told the script to use.

4. A single slide is left whiles other remaining have been removed and saved as a separate stub file. This will allow feature phones and less capable phones to download just a single slide image whiles saving bandwidth. Note, this is not valid html, but we’ll use jQuery to open slides-stub.html and load the images into the slideshow.

5. Navigation has been removed and placed in core.js for graceful degradation.

6. The jQuery .get() function opens our HTML snippet (slides-stub.html), and inserts its contents into our mod­ule by using append(). If JavaScript isn’t available, or if jQuery can’t load that file, then the user is presented with a single image at the top of the page: a perfectly acceptable fallback for our design .

7. We could po­tentially optimise further the slideshpow and restrict it to only appear on certain types of displays, making the script resolution dependent. For example, if we wanted to prevent it from loading at all on smaller screens, we could work a simple resolution test into our script:

**if (screen.width > 480) { $(document).ready(function() { … });}**

That opening if statement is the JavaScript equivalent of a min-width: 480px media query: if the screen is narrower than 480 pixels, then the enclosed JavaScript won’t fire.

And we could refine this approach even further. For example, we would ideally use a lightweight JavaScript loader like LabJS (http://labjs.com/) or Head JS (http://headjs.com/) to dynami­cally load jQuery, the carousel plugin, and our own custom.js, perhaps including them only if the user’s screen is above a certain resolution threshold. That would help ensure that users on smaller screens aren’t saddled with the overhead of downloading all that JavaScript, especially if we’re keeping the carousel from loading for them.

While’s we’re being band­width-conscientious, I’d probably use Filament Group’s fan­tastic “responsive images” library (http://bkaprt.com/rwd/56/), which would allow us to serve lighter, more bandwidth-friendly images to smaller displays, with the full-sized images served only to wider screens.