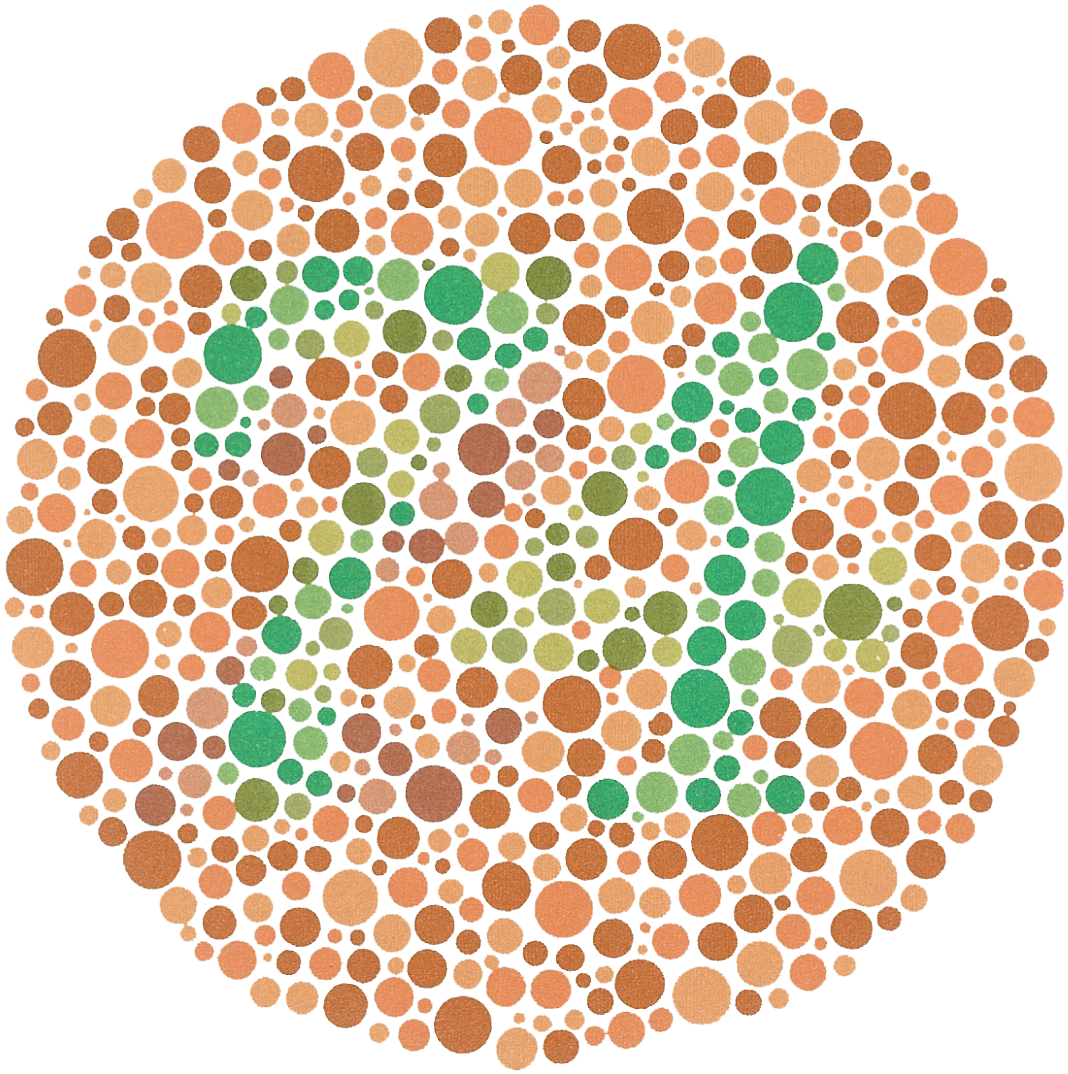
Simple test automation, with no moving parts.

03 October 2012 · automation, fishing, heuristics, system, testing, unicode



(http://en.wikipedia.org/wiki/File:Ishihara_9.png)

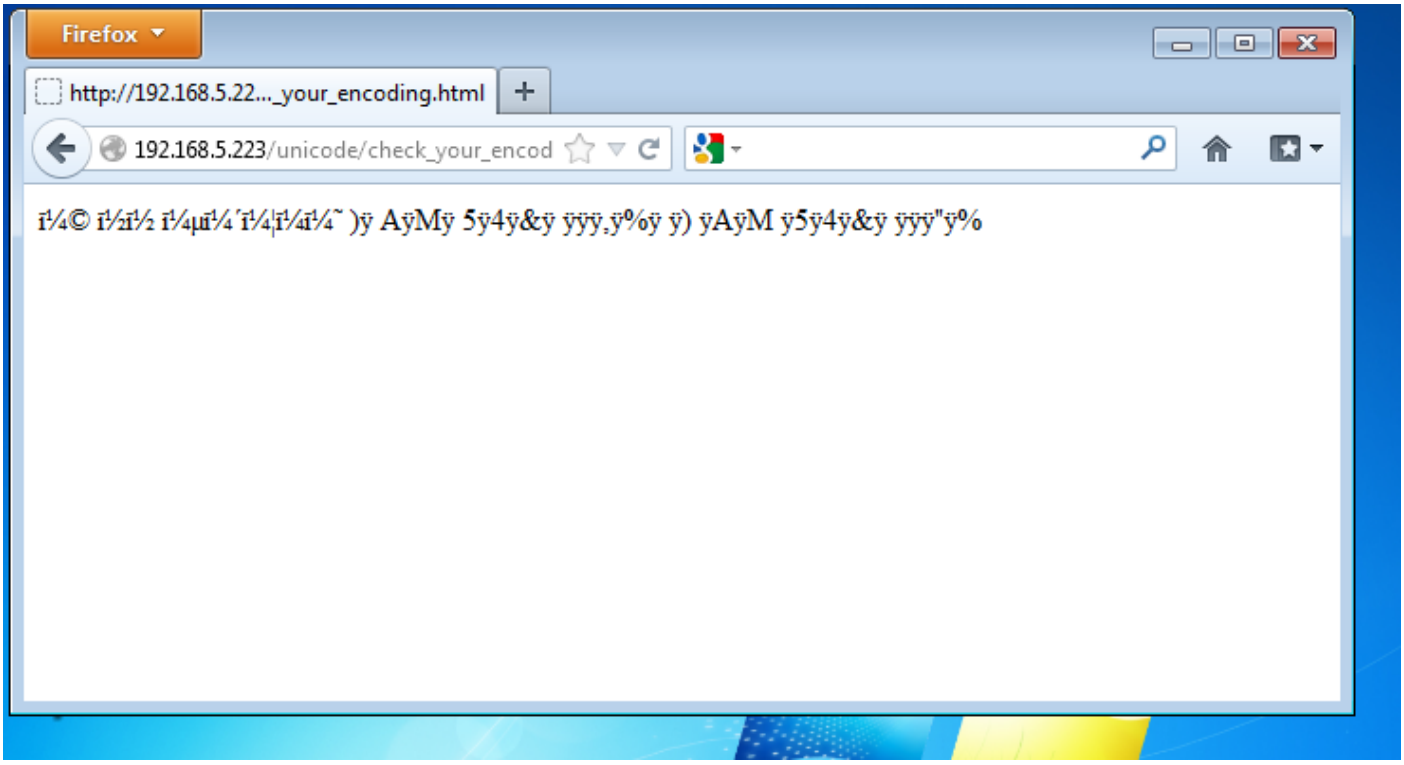
Can you see the 74? (http://en.wikipedia.org/wiki/File:Ishihara_9.png)

**This is an Ishihara Color Test. Its used to help diagnose colour blindness, people with certain forms of colour blindness would not be able to read the text contained in the image. The full set of 38 plates would allow a doctor to accurately diagnose the colour-perception deficiencies affecting a patient.

The test is ingenious in its concept, yet remarkably simple in its execution. No complicated lenses, lighting, tools or measuring devices are required. The doctor or nurse can quickly administer the test with a simple and portable pack of cards.

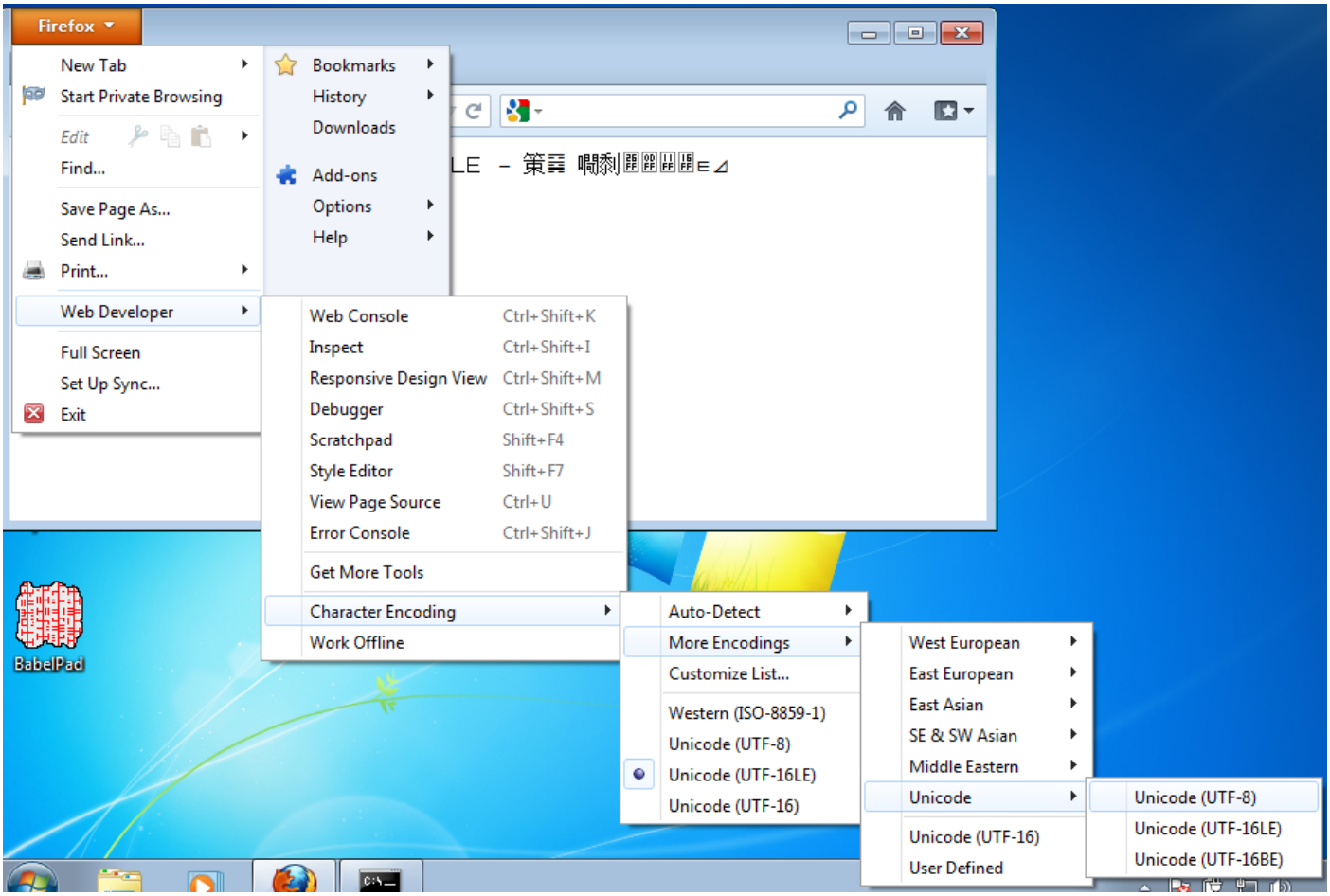
The Ishihara test is an end to end test. Anything, from lighting in the room, to the brain of the patient can influence the result. The examiner will endeavour minimise many of the control-lable factors, such as switching off the disco lights, asking the patient to remove their blue tinted sun-glasses and maybe checking they can read normal cards (e.g. your patient might be a child.).

End to end tests like this are messy, many factors can be in-play making classic pre-scripted test automation of minimal use as the burden of coding for the myriad of issues can be prohibitive. Furthermore, despite their underlying-complexity, End to End tests, are often the most valuable – they can tell you if your system can do what your customer is paying for. For example, are your data-entry-inputs making it out to the web? Are they readable by your users?
These Ishihara style tests, are a quick way of analysing that end-to-end view. This is an area I have been looking into recently, here's an example of a Unicode-encoding detection file, as rendered by default in Firefox.

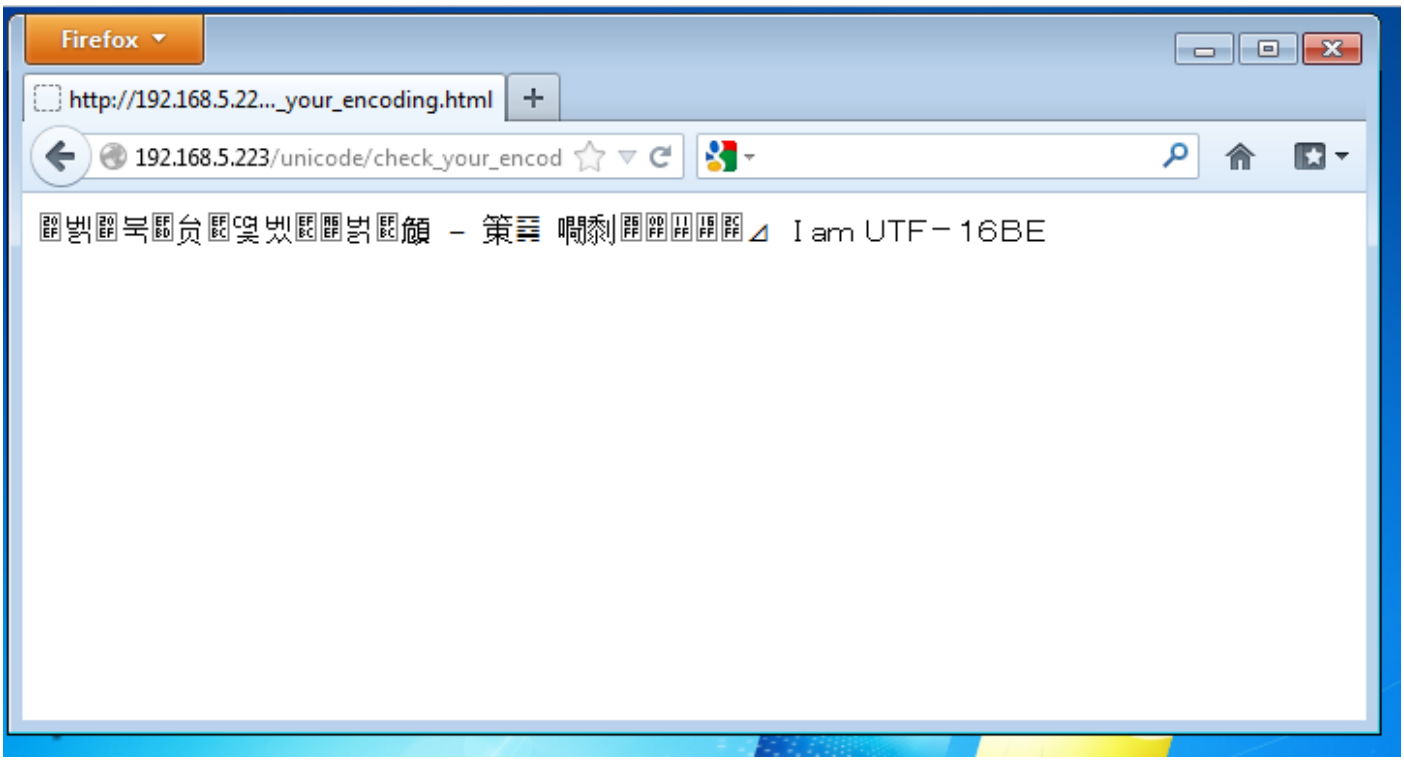


The fact that none of the text is legible, tells us that the text is not being rendered in the common Unicode formats (Known as UTF-8, UTF-16LE or UTF16BE). This type of rendering problem is known as Mojibake (http://en.wikipedia.org/wiki/Mojibake). Depending on your context, That might be expected, as by default HTTP uses an older standard text encoding standard***(labelled ISO 8859-1, which similar to ASCII)**.*

You can actually change how FireFox and Internet Explorer 'decode' the text for a page. These are menus to do it in FireFox on WIndows 7.

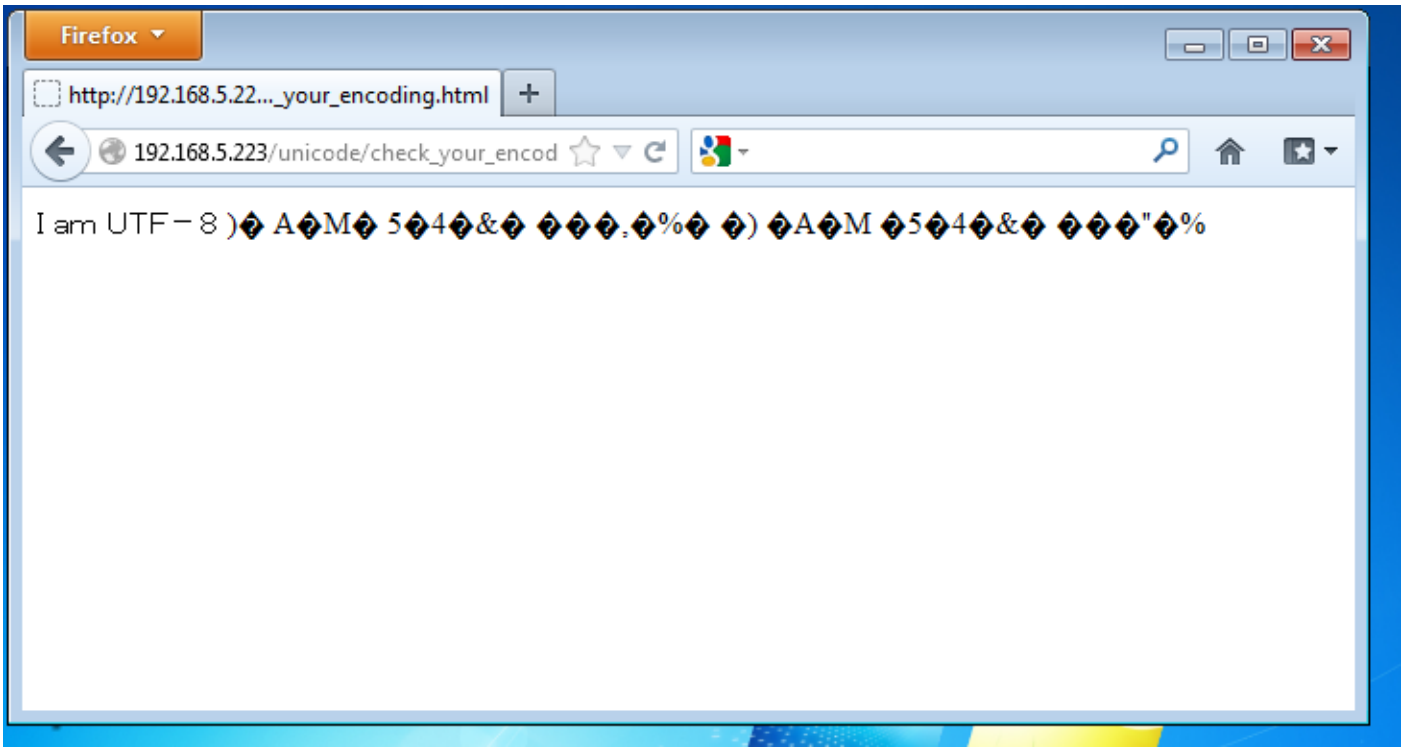


If I change FireFox to use the menu option "Unicode (UTF-16)" character encoding, This is what I see:



Notice the page tells me it is being rendered in UTF-16BE. Our special page has reverse engineered what FireFox browser means by UTF-16. There are in fact 2 types of UTF-16, BE and LE (If you are interested, you can find out more about this Big Endian / Little Endian quirk (http://en.wikipedia.org/wiki/UTF-16#Byte_order_encoding_schemes)). That's interesting, why did it use UTF-16BE? Is it using UTF-16's predecessor: UCS-2's default ordering of Big-Endian?
**(Don't worry this stuff IS ACTUALLY CONFUSING.)

If I change FireFox to use what is fast becoming the de-facto standard, UTF-8, the page tells us likewise:**



I could do other similar investigations, by checking HTTP headers. I might also examine the page-source and the encoding that has configured. But alas, its not uncommon for these to differ for a given page. So to find out which encoding is actually being used? The Ishihara tests can help.

Unlike other methods very little setup is required, the files just need to be included in the test system or its data. They are safe and simple - They don't execute any code at run time and are not prone to many of the usual programming relating maintenance issues.

When might you use Ishihara style tests? Whenever you suspect there is some medium that might be interfering with what you are seeing. For example, If you deploy a new cache in front of your website - it shouldn't change how the pages actually are encoded [should it?]. (Changes in encoding might change a page's appearance - now you have a quick way to check the actual encoding in-use.)

Remember that end-to-end view? Well if our system has multiple steps - which process or affect our text - then any one of those steps might in theory highlight an issue. So even if viewing our test file suggests it is being treated as UTF-8, this might just mean that for example our back-end content management system processed the file as UTF-8. The next step may have again changed the data to a different encoding. So while we can't always be sure what is affecting the Ishihara test text, we can at least see that something in that black box is affecting it in a visible way.

I’ve only scratched the surface here with the idea of Ishihara tests. They can provide greater resolution in issues such as character/text encoding e.g. Did that euro symbol display OK? Well we know you are not using ASCII text encoding then etc. The technique can be used elsewhere, have a try yourself. You can download the simple example above (https://sites.google.com/site/investigatingsoftware/check_your_encoding.txt).