# 36 Days of Web Testing

**By Rob Lambert**

www.thesocialtester.co.uk

www.twitter.com/rob_lambert

# Welcome

Welcome to the "36 Days of Web Testing" eBook.

I originally posted this content out via my blog, as a series of 20 posts. I kept back 16 extra ideas for the purposes of not boring people with too many posts. Here are all 36 in one book.

I actually had around 60 ideas but whittled them down to 36 to keep this book short and succinct.

It was amazing to get such a great response from the posts on my site. Thank you to all who commented, emailed, tweeted, re-tweeted and otherwise shared the content around.

It was great to hear about people finding bugs after trying the techniques – the ultimate feedback for those of us sharing hints and tips. It was also great to hear from people running startups or small teams with no formal testing or testers; many found the posts very helpful in giving them ideas to test with.

This book is completely free and I would encourage you to share it with your friends and colleagues, but please don't change it  and please try to reference the source when possible. (www.thesocialtester.co.uk)

The ideas in this book are not in depth, or technically informative but I do hope they give you a sense of curiosity to explore the ideas further. They are starting points for your test ideas. Use them as you see fit but never stop exploring your testing, your product and your skills. Exploration and development of skills is at the heart of good testing. Enjoy.

# Cross Browser

## Why?

There are a growing number of browsers entering the market (especially for tablet and mobile devices) which means your website under test needs to be working well for your supported browsers.

The existing mainstream browsers (Internet Explorer, Firefox, Chrome, Opera, Safari) are also updating themselves very frequently and often without the end user being aware of the update.

This makes testing against this growing number of browsers essential but also increasingly more time consuming.

As with all testing though, ensure you check what is supported and what isn't by your own product/company. One of the most successful ways to reduce browser testing is to stop supporting older browsers. This is a good strategy for some companies, but not possible for all.

It obviously doesn't make these old browsers bug free but it does mean you can ignore potential problems with these older versions and focus on the version you actively support.

## How?

## Spread The Risk

One approach could be to run your everyday tests against a mixture of browser environments.

For example, you may be testing a simple web application where the user can log in, generate some reports, send the reports and then log out. The system also has a simple "management" system where sys admins or managers can view who is changing what.

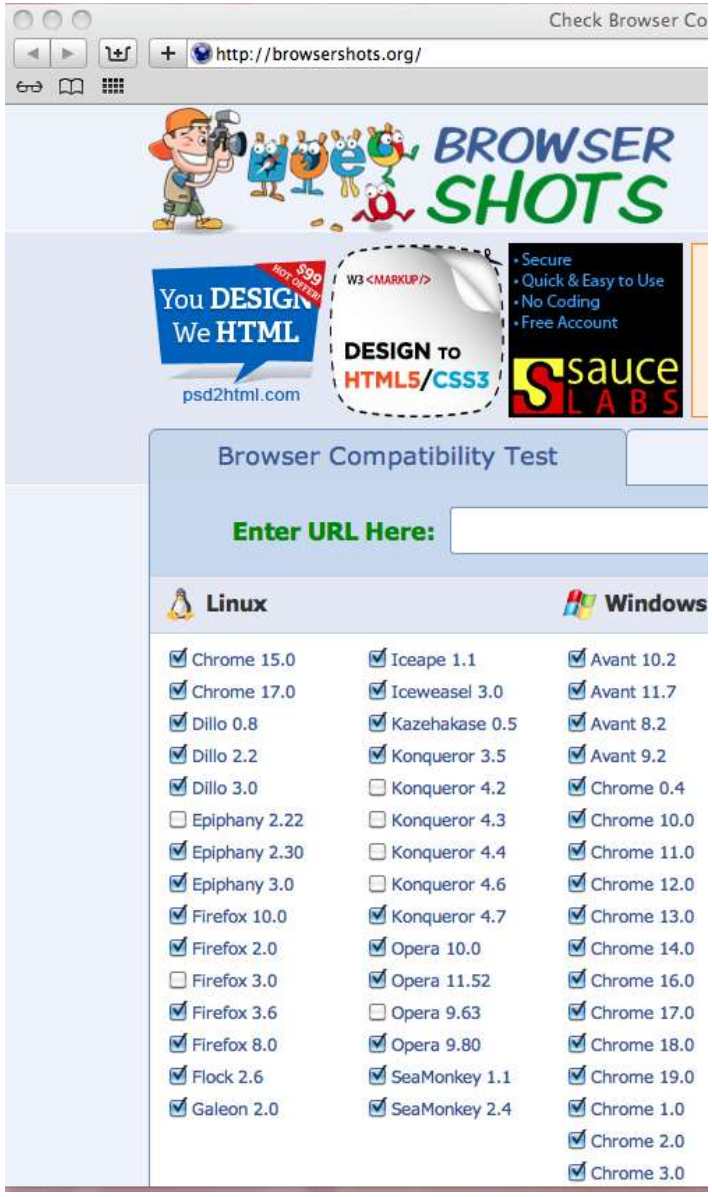
To get a wider coverage you could test your log on functionality in one browser, test the send report functionality in another browser and then the audit trail functionality using a third browser.

This is an effective way of covering different combinations of browsers at the same time as doing your day-to-day testing. The above example won't highlight a bug with the audit trail functionality in the first and second browser though. So there are plenty of gaps for bugs to slip through, however the time it saves may push your decision down this route.

## Let someone else do it

For appearance issues there are many online tools like Browser Shots (http://browsershots.org/), which will load your page in any of the supported browsers (they support a great many versions), take a screen shot and then make these screen shots available to you.

This is great for websites, but applications that require credentials or have a huge number of pages are somewhat more difficult.




*The Browser Shots homepage*

## Check against standards

You could validate your site against an HTML standards checker. Checking against HTML will give you more confidence the site works across many browsers, but there are still some browsers that will render pages differently.

A good standards checker is at the W3.org (http://validator.w3.org/) page

## Automate

You could automate your checking using something like Selenium (http://seleniumhq.org/) and then use a tool like Selenium Grid (http://selenium-grid.seleniumhq.org/) to test against several browsers.

This has infrastructure requirements and is obviously reliant on an automation suite existing (or going to be brought in to existence).

It will also only check what you have told it to check. Automated checks are great for functionality but will not tell you about layout issues between browsers.

## Fight Layout Bugs

You could write some automated tests to check the layout issues your site may face in different browsers.

Fighting Layout Bugs is an open source Java project providing an automated way of checking for potential layout bugs. As it's Open Source you can add to the project.

Fighting Layout Bugs – (http://code.google.com/p/fighting-layout-bugs/)

## Manually Test It

You could manually test against each version by installing all of the versions on different machines. To make this less of a chore it would be worthwhile using Virtual Machines.

These can then be cloned and re-used by others, or centralized for multiple users to access. This too relies on infrastructure. It also requires significant time and effort, but gives you the human analysis of cross browser compliance.

## Categorize

You could narrow the browsers down to engines.

Chrome and Safari use WebKit. Firefox uses Gecko. Internet Explorer uses the Trident rendering engine. Opera uses the Presto rendering engine.

You could therefore “assume” that testing on Chrome will cover off Safari. This may be an assumption too far for many though.

## Emulate it

You could use tools that attempt to emulate the different browsers. There are a number of tools on the market, as well as some inbuilt tools within the browsers themselves to render older versions.

A few words of caution though, they are not wholly accurate in my experience, but they *may* provide insights that are useful.

## Outsource Selenium

If you have a suite of Selenium tests but do not have the infrastructure or experience in house to get a grid up and running then you might find services like Sauce Labs (http://saucelabs.com/) and Testing Bot (http://testingbot.com/) quite useful.

I’ve not even touched on different operating systems or mobile devices but as you can see there are a number of ways to approach the cross browser issues.

Cross browser support may be your biggest pain-point, especially with the rapid rate of release that most of the browser companies are now adopting.

In my experience a nice selection of a number of the above works well.

# Useful Hint

Some browsers come with compatibility modes, which allow you to emulate different versions.

Some browsers, like Chrome, have developer tools for rendering pages in different browser and Operating Systems.

# Useful Links

Compatibility Mode page on Wikipedia - http://en.wikipedia.org/wiki/Compatibility_mode_(software)

Sauce Labs - http://saucelabs.com/

Testing Bot - http://testingbot.com/

Browsers and Rendering Engines - http://en.wikipedia.org/wiki/Web_browser_engine

Selenium Grid - http://selenium-grid.seleniumhq.org/

# Web Accessibility

## Why?

Very few companies take web accessibility very seriously but in a world that is increasingly moving online it is crucially important not to marginalise people who cannot interact with websites and applications.

Accessibility testing is very simple in one respect, but really hard in another.

The accessibility of a site is typically judged against International compliance standards set out by W3C. Their guidelines essentially give three levels of compliance A, AA and AAA, although these guidelines are under continual review.

## How?

To check basic compliance of your websites there are a number of tools on the market, which can help you get a “yes” or a “no” against the recognised compliance standards. These automated tools will be pretty quick to scan the page code for compliance.

This is the easy approach and you can get an answer in minutes, however, the answer you get may not tell you the whole truth.

In a very basic example you may have an image on the page.

The image may not have anything populated in the ***alt*** attribute.

The ***alt*** attribute is essentially to provide a text alternative to the image for when the Image is not available, or your visitor is using a screen reader.

This user will hear the text you have put for the alt attribute.

Running your site against an automated checker will give you a fail because you have no value for the ***alt*** attribute. This is easily fixed.

So you populate something in the ***alt*** attribute and re-run the scan. It passes.

However, your picture could be of a red apple and the text you entered could read “Cute Cuddly Brown Teddy Bear”.

In accessibility terms, the ***alt*** attribute in the IMG tag tells screen readers (and hence people using the screen readers) what the picture is.

So the text of “Cute Cuddly Brown Teddy Bear” is not very helpful, and is actually misleading for someone who cannot see the picture. An automated check will not tell you this.

It will also not tell you whether the flow and logic make sense to someone who cannot see the site and may not have prior knowledge of the page they are on.

In my experience there are very few people better at testing for accessibility than the very people who will use your accessible site.

It’s therefore important to consider outsourcing your testing to specialist accessibility testing companies or charities. They will offer you insights you will struggle to get through your own testing.

I once tested a site to AA compliance according to the automated checkers, yet when I handed it over to an accessibility-testing expert, I was stunned with how unusable it was even though it met the compliance. It was compliant, but it wasn't very easy or pleasing to use.

Good accessibility testing needs human judgment and thinking. It cannot simply be automated. It requires trials and experimentation. It requires feedback from your target end users.

Saying that, the online tools are incredibly good at finding the obvious errors and compliance issues. Each year they become more refined, informative and powerful.

When you run your site against these tools be prepared to be surprised at how inaccessible your site is, especially if it hasn't been developed with accessibility in mind.

Whether you are aiming for compliance or not, an accessibility check will also point out some obvious flaws in the HTML and give you insight for areas to explore and test further.

And if your company is **not** intending your site (especially public facing sites) to be compliant to even single A, as a good Tester, shouldn't you be asking "***Why Not?"***

# Useful Hint

This simplest way to check against W3C compliance would be to open up Firefox, install the accessibility checker plug in, open your site to be tested and run a quick scan.

# Useful Links

Firefox accessibility extension - https://addons.mozilla.org/en-US/firefox/addon/accessibility-evaluation-toolb/

Tips on how best to describe the alt attribute - http://webdesign.about.com/od/beginningtutorials/a/aa122004.htm

The W3 standards on accessibility - http://www.w3.org/standards/webdesign/accessibility

Test Partners accessibility testing - http://www.testpartners.co.uk/accessibility_testing.htm

# Is the HTML valid?

## Why?

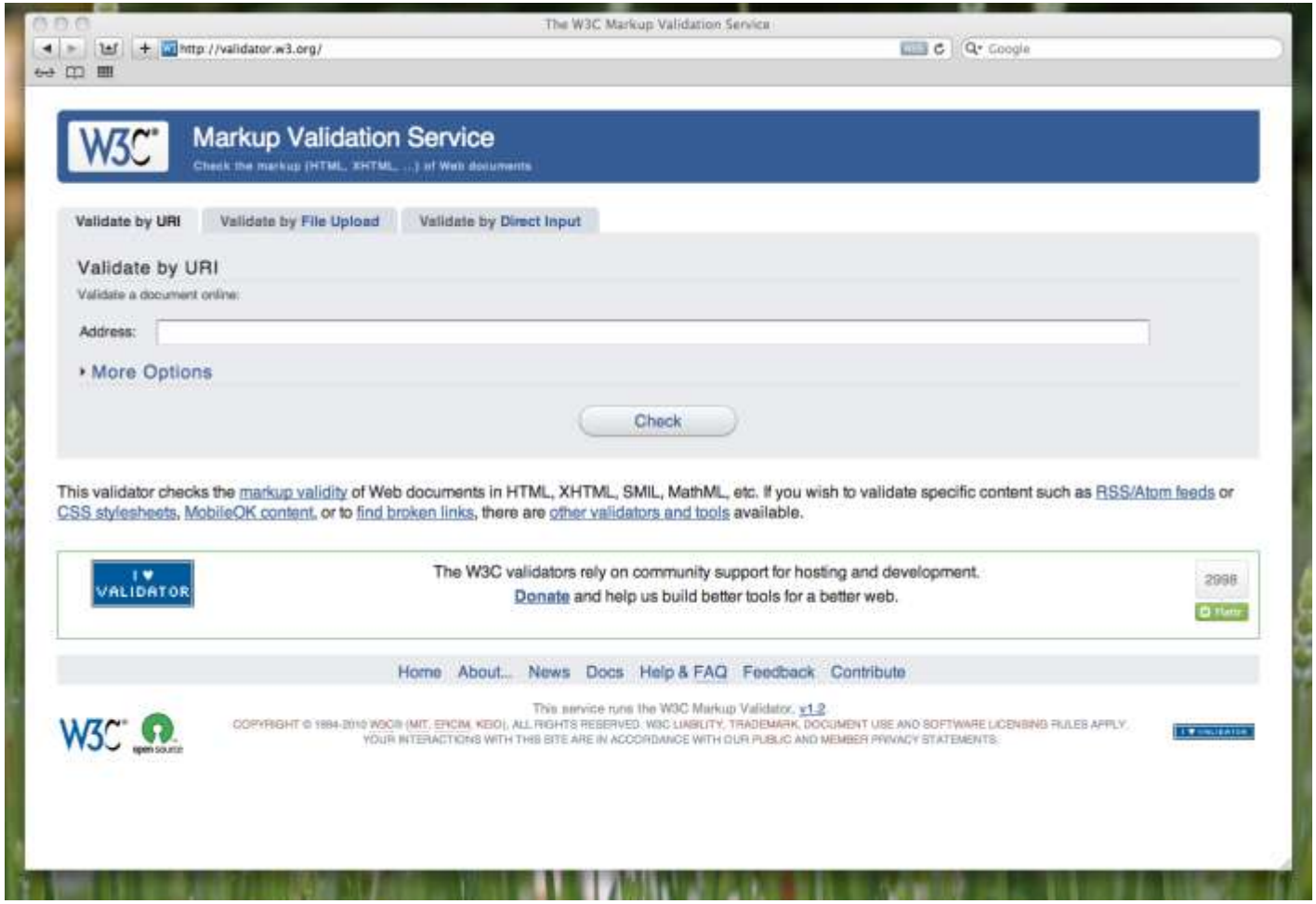
HTML is the main underlying markup language of websites and other web delivered content.

Poorly constructed HTML can lead to bugs, rendering issues, browser compatibility issues and poor accessibility compliance, so it's a good area to look for when web testing.

Adherence can also help to ensure your site is future proof too.

There are many sites that don't adhere to the typical HTML standards including some high profile sites. Reasons for non-compliance to standards are often cited as speed and cross browser support (which is ironic since HTML 5 is supposed to guarantee cross browser conformance) but that's something I'm not digging deeper in to here.




Suffice to say that it would be prudent to run a basic HTML compliance check using one of the many tools available. It will give you a starting point for discussion anyway, whether your business decide to fix the issues or not.

# How?

Tools like FireWeasel Firefox extension will help you validate as you test.

There are also several validators online that give you quick feedback on the validity of the HTML under test.

These validators will give you information on what part of the HTML is failing validation and also pointers as to what can done about it.

# Useful Hint

How about encouraging the programmers on your team to install development IDE compliance checkers if they are available. This is the ultimate early warning against HTML validation errors.

# Useful Links

Good post on why you should validate - http://www.stevefenton.co.uk/Content/Blog/Date/201108/Blog/Do-I-Need-To-Validate-My-HTML

The W3C validator - http://validator.w3.org/

FireWeasel - https://addons.mozilla.org/en-US/firefox/addon/fireweasel/

# Check for Dead-links

## Why?

Almost all websites suffer from something called Link Rot where hyperlinks are left to decay resulting in many links that no longer go anywhere, or point to out of date and unsupported pages.

The page they linked to may have been removed or they are simply no longer referencing the correct location.

There are also errors in linking which could mean some links on the pages simply don't work.

It would make sense to keep on top of the links in the first place avoiding time spent maintaining them, but this is easier said than done when faced with developing commercial sites and applications.

Dead-links can hint at other issues and areas of concern for the keen web Tester.

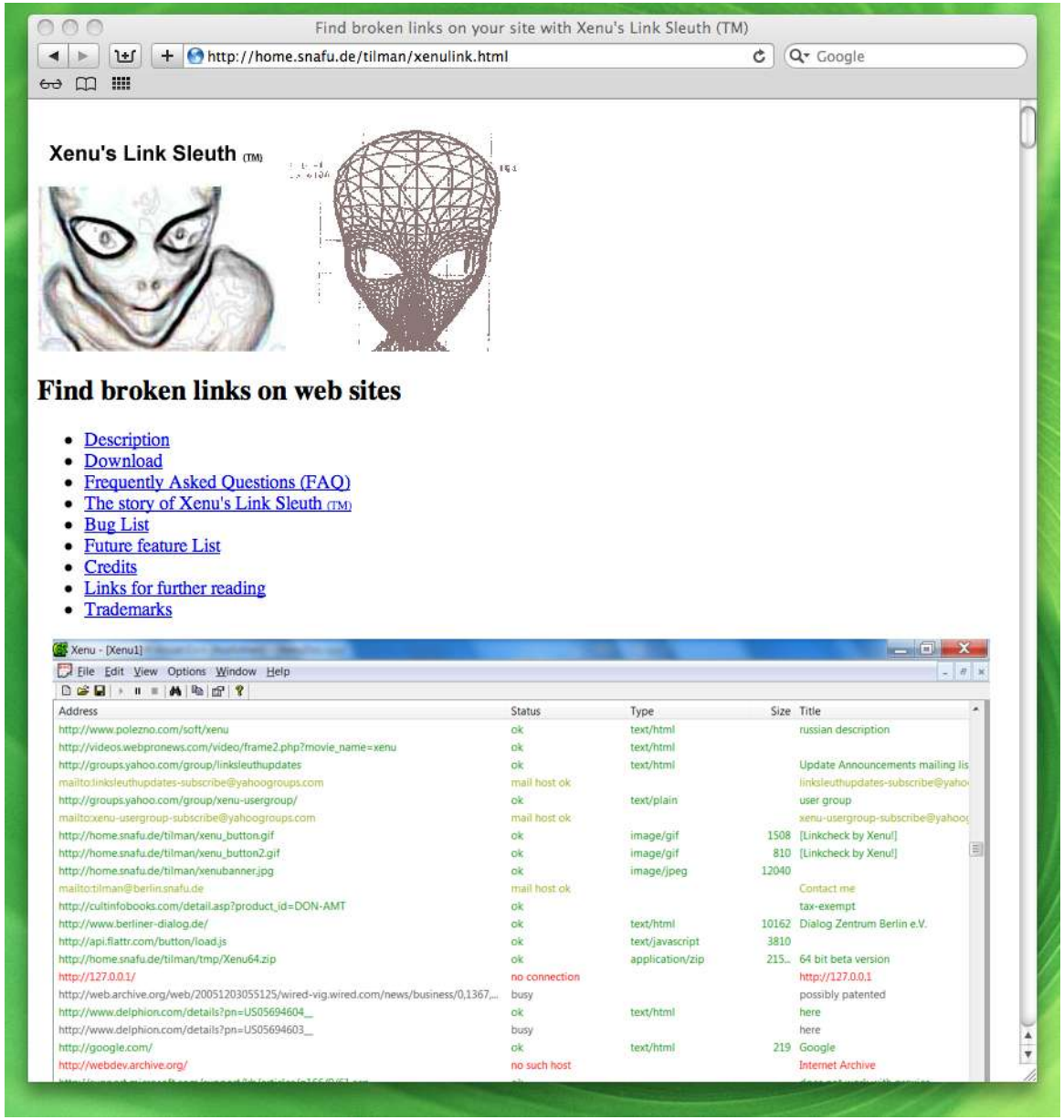
## How?

There are many tools out there that will scan your site, provide you with a list of dead-links and then suggest ways to fix the problem.

Mostly it is a case of just removing the offending link from the page but this can also come with it's own problems.

Download a link-checking tool, like Xenu, add in your URL and then go through the report.



# Useful Hint

How about running a dead-link checking tool as part of your Continuous Integration or Overnight Build Process? That way you can see dead-links after an automated test run.

# Useful Links

Xenu Link Checker - http://home.snafu.de/tilman/xenulink.html

W3C Checker - http://validator.w3.org/checklink

Broken Link Checker Tool - http://www.brokenlinkcheck.com/

# One, Two and many

## Why?

Unless you are creating a very niche site then you will probably be expecting a reasonable number of visitors to your site. As such, it would be prudent to check that it works for X number (where X is a number relevant for your context) of users.

One of the trickiest elements of testing for multi-user or even load testing your application is defining the end usage patterns and expected load.

For example, some sites may be used everyday by only a handful of people, but at the weekends visited by many thousand. There is also the added complexity of mapping out the end user usage patterns. Are all end users going to perform the same actions each and every time they visit your site?

One of the simplest methods for starting your journey to multi-user testing and then load/performance testing is to use the simple concept of One, Two and Many (or other variants of these numbers)

I personally like the One, Two and Many because it's easy to explain and it's also provided me with exceptional value when used. I've used it for testing as well as proving automated scripts.

The theory goes like this:

You test with one user to make sure the basics work at a functional level.

You then test with two concurrent users to check that there are no basic deadlock issues and that the application/site allows multi-user activity at a basic level. You'd be surprised at how many issues this test

could find. The amazing thing about this sort of test is that it's quite simple and doesn't need you to spend months building a giant automation framework to find the basics don't work.

You then test with many users. The many being whatever value you deem to be right for your goal. That could be ten, twenty or maybe even two million. The choice is yours.

The reason for starting low and then jumping to big numbers is that I've seen a few examples where expensive infrastructure and test code was prepared for load testing, only to find the product wouldn't even allow two users to log in at the same time.

# How?

I like rapid feedback so I personally perform the "One" and "Two" manually, maybe with some support by simple automation like Selenium or even ignoring the UI and diving in at the web service level.

For the bigger loads there are countless options, of which I'm not going to explore in this book.

My personal favourite at the time of writing this book is jMeter (http://jmeter.apache.org/), simply because it's free, easy to get started with and easy to extend.

Saying that, you could opt for any number of solutions, free to very expensive, locally installed or hosted; maybe even a complete managed service.

Whatever your decision never lose site of what your goals are. Performance and Load testing can become complicated, unwieldy and expensive as you explore any number of variables unless you keep focused of what you're goals and ambitions are.

# Useful Hint

Do a proof of concept or trial run with free or "free to try" tools before opting to buy any product. Tool quality and suitability can vary greatly.

# Useful Links

A list of testing tools - https://en.wikipedia.org/wiki/Load_testing

jMeter - http://jmeter.apache.org/

# Multiple tabs and windows

## Why?

Opening the application in more than one tab or window can often lead to interesting security bugs, data refresh issues or multiple cookie confusion.

An often-overlooked test is the multiple tabs/windows in the same session test. What you are doing here is essentially seeing how the application copes with the same and/or different user operating under the same session token, but on different pages or tabs.

Many sites use Cookies (a small file stored on your computer) to store session IDs and other supporting data, hence it is often possible to find bugs around confused data, sessions or security access. This can manifest itself as incorrect data display issues, security loopholes and actions not being performed as expected.

## How?

Here's a simple example to seek out security flaws.

Open Firefox and in one tab log in to your secure application.
Then right click on a page and open that new page in a new tab. Both tabs are now considered to be in the same session.

Now log out of the application in Tab 2 and try to perform some actions in Tab 1.

Has it logged you out or let you perform an action?

In most situations it would have logged you out.

In many instances it is possible to log in to two different accounts in the same browser and end up sharing data across both tabs because of cross authentication.

This is a real example.

I log in to my banking application in Tab 1 as Rob and then in Tab 2 as Dave.
Now when I switch to tab 1 and follow a new link I might get to see Dave's information.
In Tab 2, Dave might even get to see my data.
This is because of confused session data stored in a single cookie for the browser, which is shared between two tabs.

It's not just about authentication though.
What about adding things to a shopping basket in multiple tabs – do they persist in the basket? What about state changes in your application across several tabs?
Can I login across several browsers in different sessions?
Can I trick browser side validation and restrictions by performing actions across a number of tabs?

The best way to test in multiple browser tabs and sessions is to explore the application with multiple tabs open, checking what effect a change in one tab can have on the other.

As you explore around look for data, states and actions that might be confused by bad cookie management, session management and cross tab problems.

Always have some developer tools open so you can see what requests and responses are being communicated and what the content of these is.

Burpsuite security tool is especially useful if you want to start hijacking the session and manipulating stuff. Firecookie is a Firefox extension that shows you the cookies and their content.

# Useful Hint

In several of the modern tabbed browsers it is possible to open up multiple tabs and then drag a tab out of the main browser “window” to create two ‘Windows” operating under one session. This makes it easier to switch and view the two tabs whilst testing using CTRL and Tab (or CMD and Tab on Mac)

Not all browsers may treat a new “window” as the same session.

# Useful Links

Good site about cookies and sessions - http://www.allaboutcookies.org/cookies/session-cookies-used-for.html

Session Hijacking - http://en.wikipedia.org/wiki/Session_hijacking

Security implications of cookies - http://it.toolbox.com/blogs/securitymonkey/successful-hacking-with-xss-cookies-session-ids-11098

Burpsuite - http://portswigger.net/burp/

Firecookie - https://addons.mozilla.org/en-us/firefox/addon/firecookie/

# Http and https

## Why?

When transmitting private and confidential data over the Internet is should always be secured and encrypted so that prying eyes on the network cannot see the data being sent.

This mechanism typically uses the https protocol.

***Hypertext Transfer Protocol Secure** (**HTTPS**) is a widely-used communications protocol for secure communication over a computer network, with especially wide deployment on the Internet. Technically, it is not a protocol in itself; rather, it is the result of simply layering the Hypertext Transfer Protocol (HTTP) on top of the SSL/TLS protocol, thus adding the security capabilities of SSL/TLS to standard HTTP communications. - Wikipedia (https://en.wikipedia.org/wiki/HTTP_Secure)*

If you are transmitting data and you are using the **http** protocol then it is not secure.

Hence, anyone can intercept the message and read the contents. Perfect for a “man in the middle’ attack. And that’s where you, as a Tester, come in.

## How?

The easiest way to check whether the site you are connected to is secure is to check the browser address bar and look for **https**. Different browsers will also show a padlock to symbolize that the connection is secure.

There are also a growing number of add-ons and tools for checking secure connections. There are also many browser add-ons that cam force SSL connections got sites.

If the site is not HTTPS secured then you can use a web proxy tool to intercept the messages between the web client (browser) and the web server.

You can intercept the messages using tools such as Burpsuite, Charles Proxy or Fiddler. (and others). Messages can be intercepted between the client (browser) and the server, and also coming back the other way between the server and the client.

Once you have intercepted the messages using a proxy tool you can then do a number of different tests and attacks.

You could extract information from the message and see whether there is anything personal or private in it. It may provide minor clues or snippets of information that could be used for Social Engineering attacks.

You could delete the message and see what happens to the system. Do you lose information, lose states, break the client, break the server, handle it gracefully, or do nothing?

You could forward the message to the server with different values. There are countless examples where the price of goods to be purchased is included in the message unencrypted. A quick change and you could get goods for whatever price you want.

You could intercept the messages from the server to the client and do similar attacks and tests. There are a number of other attacks and tests you could do with messages that are intercepted.

Intercepting messages is not just about security breaches and attacks. There are loads of examples in most web systems where missing messages can cause grief. Explore and learn from each test that you do and you will soon build up experience of what works and what doesn't. I often find that each test leads to new ideas when using proxies to intercept messages.

# Useful Hint

In many “test environments” https security may not be enabled and configured so double check before generating bugs against https and SSL.  Just make sure it’s on when the site goes live ☺

# Useful Links

Good security for web checklist – http://www.techrepublic.com/blog/security/ensure-basic-web-site-security-with-this-checklist/424
Differences between http and https - http://www.virtu-software.com/ask-doug/QandA.asp?q=7
Another differences post - http://www.wisegeek.com/what-is-the-difference-between-http-and-https.htm
Burpsuite - http://portswigger.net/burp/

# Client and server watching

## Why?

There are many times as a Tester when it is useful to see what is happening behind the scenes when testing a website.

When interacting with a website there may be some logic happening in your browser (client side) and/or server. Information is flowing between the two. It's this logic and information that's so crucial to you as a Tester.

The web pages may also use CSS to make the site look and feel the way it does.

When you click on a link or perform an action the client (your browser) sends a request to the webserver, or does some client side logic (like a calculation).

Either way, actions that a user does are causing something to happen behind the scenes. Most users don't care, but Testers should.

As an example, here is a simple "log on" mapped out with specific actions.

- User: Opens a browser and navigates to the web site
- Browser: The browser client requests the log on page from the web server

- Server: The server receives the request and responds with the relevant web page.
- Browser: The browser displays the page in the browser to the user
- User: The user fills in the required details (username and password) and hit's the "log in" button
- Browser: The browser client sends these log in details to the server securely (like username, password, page requests)
- Server: The server checks the credentials and sends a response, often with a new page for the browser to go to, or display.
- Browser: The client takes the command and does some stuff (like loading a page, throwing an error message etc)

This is a simple example for demonstration purposes. Yet even a simple log in generates messages and logic in a number of places. In more complex examples there could be browser side logic taking place, or the server may be calling out to another web service somewhere too.

It makes sense then to look behind the scenes and understand how your application/site is working and what messages it is sending and receiving. Mapping this information out will give you a great starting block to explore the system. You can build up a profile, which may include potential test ideas, weaknesses, strengths and potential gaps.

Some things of interest might be:

- Is the data being sent and received correct?
- Is the layout working as expected?
- Are the response times satisfactory?
- How do timeouts affect the site?
- Are there any errors in the code?
- What happens if I pass in X?
- What happens if I remove X?
- What happens if the message is lost completely?

# How?

There are three core tools I see most seasoned Web Testers relying on. (Note, I use these three as examples, there are many more tools out there, which are just as powerful and capable)

## Charles Proxy

Charles Proxy is a great tool for sniffing web requests (i.e. watching and potentially intercepting requests.)

You can check the content against standards or specs. You can also change the request to send invalid or different data.

One of the most useful features of Charles Proxy is the ability to “Throttle” the bandwidth of the traffic between the client (your browser) and the webserver. I discuss Throttling in a later post.

## Fiddler

Despite a rather dubious name Fiddler is a must have tool for testers. In a sense it does the same sorts of actions as Charles Proxy but offers a rich interface and command line to get really technical with requests and responses.

## Firebug

I use the Firefox browser a lot for web testing because of the vast array of extensions available. One of the must have day-to-day extensions is Firebug.

Firebug does loads of things but the main uses for me have been:

- Observing the traffic taking place between client and server
- Alerting me to code and message errors
- Showing me the speeds at which the pages are loading
- Inspecting elements on the web page for automation
- Changing the CSS and seeing how it renders on the page

# Useful Hint

There is another extension called Fire Cookie that extends Firebug's capabilities to Cookie management.

# Useful Links

Charles Proxy - http://www.charlesproxy.com/

Fiddler - http://fiddler2.com/fiddler2/

Firebug - http://getfirebug.com/

Firecookie - https://addons.mozilla.org/en-US/firefox/addon/firecookie/

# Security

## Why?

Security on the web is a hot topic. It's essential that security is taken seriously and that users data and activities are secure whilst they interact with your site.

I think it's important that all testers get familiar with some of the basic concepts of security testing. As a minimum I would suggest understanding the OWASP Top Ten security vulnerabilities.

There are a number of tools and test techniques for spotting some of the more obvious security flaws. For compliance and critical systems I would always advocate the use of an external "Penetration Test" by a trusted company.

## How?

One of the easiest ways to perform security testing against your application is to download a scanning tool like Burpsuite. There are alternatives but Burpsuite is one of the more approachable tools on the market. (WebScarab is good too)

The free edition of Burpsuite has most of the features available for doing specific security tests. The "Scanner" option in Burpsuite is only available when the product is paid for. I would suggest paying for the product if you plan on doing regular security testing.

The Scanner is fully automated and will scan as you work through the application, which makes hunting for vulnerabilities somewhat easier. The real skill comes in knowing where to look, what data to pass in and what to do with gaps and vulnerabilities when you find them.

**Caution: Ensure you use the Scope feature within Burpsuite and set it to only scan the URLs you are aiming your testing at. If you were to browse around your normal everyday websites with the active scanning tool on you would be essentially performing a security test against all of these sites. This is generally not seen as a good thing and is essentially hacking. So be very careful when security testing using these sorts of tools.**

Security testing is far too big to cover here, so it's worth getting a book and reading more about building threat profiles, using the tools of the trade and how to sniff out vulnerabilities for deeper attacks. It's a really fascinating topic and there are loads of resources on it.

# Useful Hint

If your products are on the Salesforce App exchange you get a license for Burpsuite for free.

There are loads of security tools available for your browsers and mobile. A quick search in the marketplaces and add-ons libraries will highlight many good ones.

# Useful Links

Burpsuite - http://portswigger.net/burp/

Web Scarab - https://www.owasp.org/index.php/WebScarab_Getting_Started

Web Hackers book - http://www.amazon.co.uk/Web-Application-Hackers-Handbook-Discovering/dp/0470170778

OWASP Top Ten - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Plynt Security Testing Experts - http://www.plynt.com/

Salesforce Burpsuite for App Exchange Partners - http://security.force.com/webappscanner

Start with the treat profile - http://www.securityalliance.co.uk/blog/bid/40335/Web-Application-Security-Testing-Start-with-the-Threat-Profile

Firefox add-ons (search for “security”)- https://addons.mozilla.org/en-US/firefox/

# Browser Extensions

## Why?

Each of the main browsers on the market has the capability to add in extra features and functions through the form of add-ons (sometimes known as extensions or plugins).

As a Tester these add-ons are invaluable at helping you gain insight as you test. There are too many to list here, but I've included some of my favourites here.

(Note: the URLs sometimes changes and some of these extensions come and go. If the link doesn't work a quick search should find it)

### Selenium IDE

http://seleniumhq.org/projects/ide/

Selenium is fast becoming an industry standard tool. This is the IDE version. Visit http://seleniumhq.org/ for more information on other tools in the set like Selenium Grid.

### IE Tab

https://addons.mozilla.org/en-US/firefox/addon/1419
Flips your Firefox tab in to Internet Explorer mode.

### Firesizer

https://addons.mozilla.org/en-US/firefox/addon/5792
Re-sizes your window to different screen sizes

## Fire cookie

https://addons.mozilla.org/en-US/firefox/addon/6683

Lets you to see and manage cookies through the Firebug add-on.

## Delicious

https://addons.mozilla.org/en-US/firefox/addon/3615

Social bookmarking service. Never lose a site of interest again and share them with the community too. Perfect for bookmarking testing sites, information, snippets and blogs.

## Clear Cache

https://addons.mozilla.org/en-US/firefox/addon/1801

Clears out the cache within the browser at the click of a button.

## Copy Plain Text

https://addons.mozilla.org/en-US/firefox/addon/134

Copies any text from the browser in to a plain text format losing all formatting.

## Fiddler Hook

http://www.fiddler2.com/fiddler2/addons/fiddlerhook/

Firefox add-on for integrating with Fiddler (a debugging tool).

## Download Status

https://addons.mozilla.org/en-US/firefox/addon/26

Shows your downloads in a toolbar at the bottom of the browser window. Keeps it nice and tidy and easy access to your downloads.

## Xmarks

https://addons.mozilla.org/en-US/firefox/addon/2410

Allow you to save your bookmarks and sync them between browsers on different machines. Works across browsers too. This is great if you test on several different machines. It means you can save your bookmarks once and find them on other browsers you have synched.

## W3C Page Validator

https://addons.mozilla.org/en-US/firefox/addon/2250

Check your page against W3C standards and compliance.

## Pencil

https://addons.mozilla.org/en-US/firefox/addon/8487

Great for screen mocks and notes.

## SQL Injection

https://addons.mozilla.org/en-US/firefox/addon/6727

Does exactly as the name suggests

## Quick Restart

https://addons.mozilla.org/en-US/firefox/addon/3559

Restarts Firefox maintaining all tabs and open docs when it launches it again.

## Firebug

https://addons.mozilla.org/en-US/firefox/addon/1843

How can you test websites without this one?

## Regular Expressions Tester

https://addons.mozilla.org/en-US/firefox/addon/2077

Tool for testing regular expressions.

## Quick Locale Switcher

https://addons.mozilla.org/en-US/firefox/addon/1333

Good extension allowing for quick switching between browser locales.

## Http Fox

https://addons.mozilla.org/en-US/firefox/addon/6647

Good tool for analyzing the http traffic through your browser.

### TAW3

https://addons.mozilla.org/en-US/firefox/addon/1158?src=api

Accessibility checking tool. Awesome.

### Firefox Accessibility Tools

https://addons.mozilla.org/en-US/firefox/addon/5809?src=api

Accessibility compliance checking tool.

This is barely scratching the surface but you can already start to imagine how useful some of these would be for your testing. Add-ons allow you to be more efficient with your time, perform tests you might normally not be able to and allow you to let the tools do the hard work.

These are obviously for Firefox, but some of these also exist for Chrome and other browsers.

# How?

Open Firefox > Tools > Add-ons > Extensions and then search for, and download each add-on. (or copy and paste the links above)

Each tool will work in different ways but the documentation on the site is normally pretty good.

# Useful Hint

Did you know you could make Firefox quicker to load? Do the following (only really for those with broadband):

- Type “about:config” into the address bar and press return
- Say “yes” to the “I’ll be careful” message
- Create a new Integer by right clicking and then choosing New > Integer.
    - Call it “nglayout.initialpaint.delay”

    - Set it's value to "0" (this will now tell Firefox to wait for zero seconds before acting on the information it is receiving)
- Type "network.http" in the filter field
- Set "network.http.pipelining" to "true" (Double click to switch the value)
- Set "network.http.proxy.pipelining" to "true" (Double click to switch the value)
- Set "network.http.pipelining.maxrequests" to a number. (Double click to edit) Try setting it to 20. (This number is the number of requests the browser can make at one time)

# Useful Links

Main extensions website for Firefox - https://addons.mozilla.org/en-US/firefox/extensions/

Chrome extensions - https://chrome.google.com/webstore/category/extensions

Opera extensions - https://addons.opera.com/en/addons/extensions/

Safari extensions - http://extensions.apple.com/

Cool hacks to Firefox like making it quicker - http://www.lifehack.org/articles/technology/15-coolest-firefox-tricks-ever.html

# Back to the beginning again

## Why?

The security of an application/site doesn't just concern the usual suspects of cross-site scripting, SQL injections and man in the middle attacks. Something as simple as hitting the back arrow on a browser and seeing the previous persons information is a low key but effective security breach.

Say for example that you are in an Internet Cafe and you log in to your banking application. You log out, but leave the browser window open on the home page and you leave the cafe. Yet it *could* be entirely possible that the next person to use the computer can see your details by simply clicking "back" on the browser.

Believe me this does happen.

## How?

A very simple test would be to authenticate and enter your application. Then log out and hit the back button on the browser to see what happens.

Browsers often offer the ability to skip back a number of pages in one go. Try this and see what information you can glean by skipping 3 or 4 pages back.

Here are some examples of how this bug could cause problems:

- You might be able to vote online for a candidate or topic and then hit back and vote again. And again. And again. And again. And again.
- You might be able to use a “one time only” coupon to discount a price over and over again by using the back button.
- You might be able to see someone else’s personal data by going back through their browsing session.
- You might be using a site where some client side JavaScript fires on page load even if the authentication fails. So although you have logged out, when the page reloads (and doesn’t let you back in) it might still fire some logic to perform an action.

# Useful Hint

Holding down the back button often gives you a list of several previous pages to visit.

# Useful Links

Stack Overflow question on disabling back capability (which you cannot by the way) - http://stackoverflow.com/questions/7816195/how-to-disable-back-button-in-browser-when-user-logout-in-classic-asp

IE security problem with the Back Button.
http://www.wired.com/science/discoveries/news/2002/04/51899?currentPage=all

# Change the URL

## Why?

A good test to run to find sneaky authentication bugs is to edit the URL of the site you are on to see if you can get access to stuff you shouldn't be able to see.

You might also want to change the URL and see if you can alter the data being sent or received.

## How?

### Bypass Authentication

Firstly it would make sense to understand how the site hangs together and which pages lead to which other pages.

You can then start to build up a profile that will help you organize your thoughts about how to try and trick the application.

So log in to your application and perform some actions, like submitting some details or checking some sensitive data. On each page copy the URL and paste it to notepad (or other tool).

Once you've built up a profile and a list of URLs then log out. This should now terminate that session. To make extra sure you might want to clear the cache, delete cookies and re-open the browser.

Now paste in a saved URL and see what happens. As your session is closed you shouldn't be able to gain access to anything that required authentication (like your sensitive data).

You may also want to try logging in as someone else and pasting in the same URLs. Some of the URLS may contain account/person specific information. Therefore you may need to authenticate to view the URL. See what happens when you are authenticated as someone else.

Can you see another account?

Are you asked to authenticate again?

Any clues as to whose account it is?

## Change the URL

Another trick is to look for any information in the URL that could be changed.

For example:

*www.thisisanexample.com/banktransfer/rate=12.33*

An obvious test would be to change the rate value and see what happens.
How about making the rate a negative value?
What about no value?

At the time of writing this book a large global bank is in the news over a basic URL error just like this. The URL shown to a logged in user contains an account ID for the user. By simply changing this number to another number users were able to gain access to someone else's bank account. It is such a simple test but so often over-looked. And this was a live issue for a giant bank handling millions of people's money. It happens.

In a sense, the above two test ideas are worthy of a decent Exploratory Testing session or two.

Seek out information, note down what happens, look for clues, work on potential security breach angles, but more than anything, have a go and learn about the potential for URL hijacking.

# Useful Hint

Using a recording tool like Selenium will give you each of the visited URLs in the script it creates.

It's a good way of picking out the URLs of interest.

# Useful Links

Selenium - http://seleniumhq.org/

# Automate the tedious

## Why?

Sometime the testing of websites/applications can be fairly tedious if you have to constantly navigate to certain points in a form, or create certain sets of data to get to specific pages or configurations to start testing.

In these instances I always find it incredibly useful to create a Selenium IDE (record and playback) UI test or simple SQL scripts to populate the database.

It makes getting to your actual test “point” much easier and quicker.

## How?

For Selenium UI tests add Selenium as an extension to Firefox and check out the online documentation on how to get started.

Simply record your navigation through the application and play back as necessary to get to where you need to be.

For example, I was once testing multiple users and cookie management and needed to log out and log in as different users with alarming regularity.

As such I created a simple automated script that would log one user out and then log the other in.

This took away a huge amount of tedious button clicking. I wasn't using Selenium to test the application, but I was using it to automate setup and tedious clicking.

# Useful Hint

Learning how to debug test scripts in Selenium IDE will make your script generation and playback more effective and quicker.

Apply caution though. If you find yourself building up an automated suite of tests and relying on them for regression then ask if this is really the right approach. Most of the time it won't be.

You can often use SQL scripts to populate data ready for testing also.

# Useful Links

SQLYog Tool - http://www.webyog.com/en/downloads.php

SQL Learning at W3 Schools - http://www.w3schools.com/sql/default.asp

Selenium Debugging - http://seleniumhq.org/docs/02_selenium_ide.html#debugging

Selenium - http://seleniumhq.org/

# Tab order

## Why?

There are many people using and operating websites without using the mouse.

If the tab order of the page is not logical then it makes using the keyboard a chore, illogical and in some cases impossible.

Testing tab orders seems like a simple check to do, but always consider the way your page renders, refreshes elements or uses neat tricks to hide content based on selections; these things may all affect the overall order of the tabbing.

## How?

For example, if you have a form that has 10 questions on it then the tab order should be fairly logical. When you press the tab button you should move forward through all ten fields in a logical order (typically left to right, top to bottom, but this will vary for each design).

However, if that same form reveals questions 11 and 12 if the user answers “Yes” to question 2 then ensure that this too is reflected in the tab order. In some instances questions 11 and 12 may be missing from the tab order.

## Useful Hint

Hitting SHIFT and TAB will do a reverse tab in most environments.
Reverse tab is just as important as forward tab.

# Useful Links

Tabbing in HTML guidelines - http://www.w3.org/TR/html4/interact/forms.html#h-17.11.1

Web Aim guide to changing tab orders - http://webaim.org/techniques/keyboard/tabindex

# Soap Testing

## Why?

If your application uses web services then it might be using the SOAP protocol.

*"SOAP, originally defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks" - Wikipedia - http://en.wikipedia.org/wiki/SOAP*

There is a wealth of information online about SOAP envelopes, rules and conventions but the basics are that it's a mechanism that sends information.

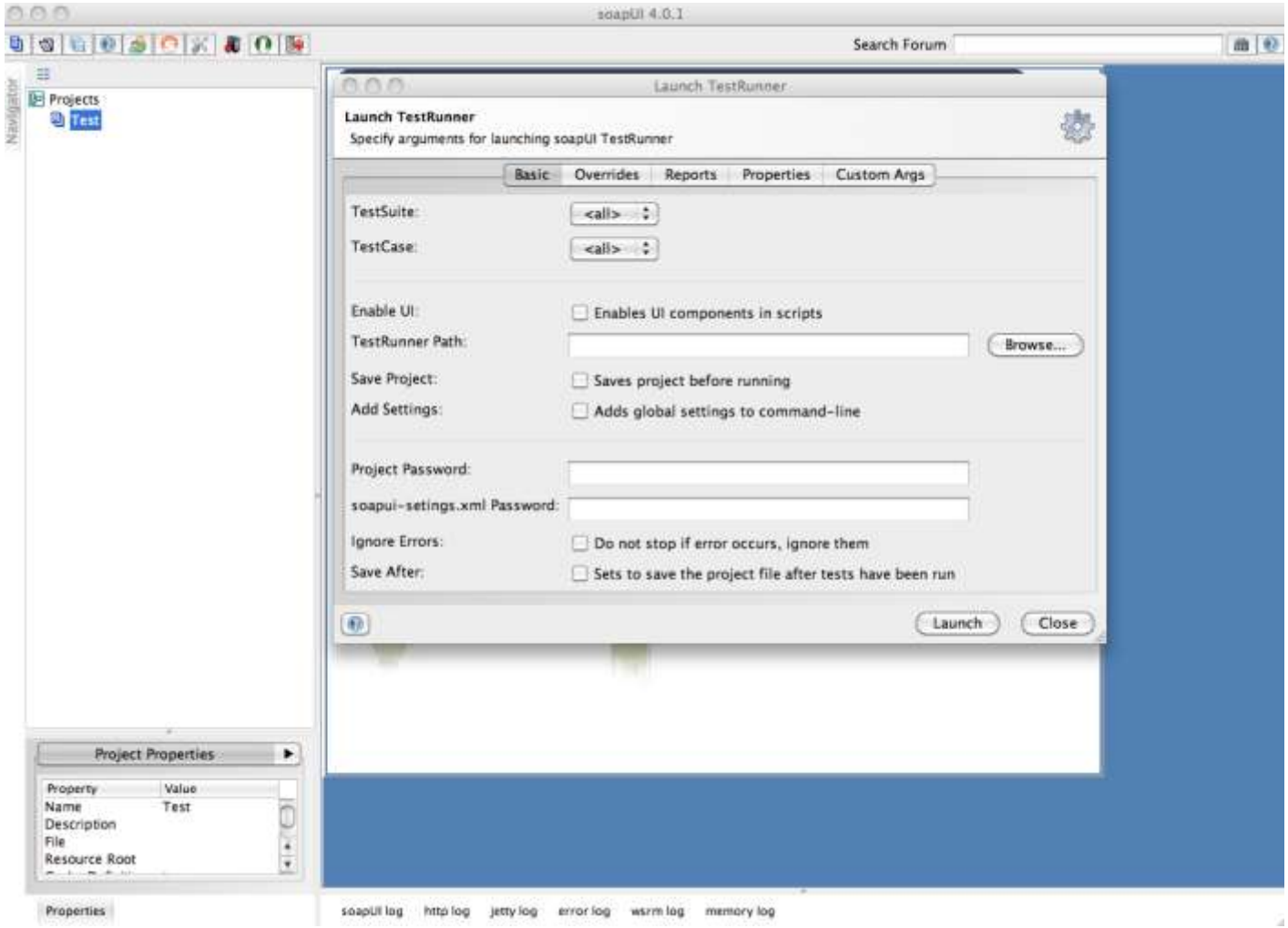
You could test at the UI level and ignore the SOAP transactions underneath, but there is much more to explore and investigate beneath the UI. It could be that you don't even have a UI at the point of testing, so you might want to test the web services directly.

You can interact with these web services and test for all sorts of things. Pass them bad data, corrupt data, invalid data, too much data or too little. You could write automated tests that could then be bundled overnight adding to your comfort levels and giving you a warm feeling inside.

## How?

The easiest way to test SOAP services is to use a tool like SOAP UI (other tools are available). It has a neat UI, lots of online support and help and is very intuitive.




There are detailed instructions on the website, but it's basically a case of connecting to the service, looking at what elements you can test and then thinking up some tests (and data). There is a sample web service available for you to test against also.

# Useful Hint

SOAP UI does RESTful and HTTP services also.

# Useful Links

SOAP UI - http://www.soapui.org

# See the source

## Why?

The source of the web page is often a great place to find bugs, areas for further exploration and potential security flaws and leakage of information.

When I talk about the source I am referring to the source code for the page you are testing. The source code is what's making the page work and as such contains snippets of information about how the application and page is put together.

There could be comments, usernames and passwords, hints referring to methods and implementations that could be useful from a security point of view. There could be rants and raves and loose comments about the company or the customers. There could be joke comments or comments alluding to partially implemented code.

There could be secret URLs and credentials in there too.

```
<body class="home blog designfolio">
  <div id="body-container">
    <div id="header-container">
      <header class="cf">
        <div id="logo-wrap">
          <h1 id="site-title">
            <span>
              <a href="http://thesocialtester.co.uk">
              The Social Tester
              </a>
            </span>
          </h1>
          <div id="site-description">
          Idle thoughts of a social tester
          </div>
        </div>
        <!-- #logo-wrap -->
        <nav class="primary-menu cf">
          <div class="menu">
            <ul>
              <li class="current_page_item">
                <a href="http://thesocialtester.co.uk/" title="Home">
                Home
                </a>
              </li>
```

# How?

When your site is open in your browser click “View” on your browser menu bar and then choose “View Source” (note: This is true for most browser but some may operate differently or call it something other than “View Source”)

The page source will open in a window. Simply reading through it may reveal some interesting areas to explore further.

# Useful Hint

Right clicking on the web page typically provides you with a context menu to view the page source too.

# Useful Links

Andréas Prins article on Hidden Treasures - http://www.thetestingplanet.com/2010/12/hidden-treasures-for-everyone/

Firefox Extension to visualise page source - https://addons.mozilla.org/en-US/firefox/addon/view-source-chart/

# Explore the competition

## Why?

A great way to generate test ideas, check compliance/claims and to see what systems your users may have previously used is to check out the competition.

Have a look at what your competitor's products/services offer and bring some of that learning back to your testing.

There are heated discussions in the testing world about whether a tester should "recommend" enhancements or suggestions to the site under test.

My answer is "**Yes. Absolutely**".

I think any tester not contributing their knowledge and insight (assuming it's welcome) to making the product better is missing out on adding great insight.

Testing isn't about happy path checking, ticking boxes and not stepping off the pre-defined path.
Testing is about adding value, insight, experience and knowledge.

These insights, if they could improve the product, should very much be aired and communicated.

If there is a chance to improve the software, make it more usable, compete in the market place and generally make a more successful product then isn't that adding value?

Don't always assume that someone else is checking out the competition or has the customer experience at the front of their mind.

Information you glean from comparing products could generate new test ideas, ways to improve features or ways to make your product stand out from the crowd. It's therefore a good testing technique to aid you.

Matching competitors on a like for like basis is not always a good business strategy though, sometimes going around/under/over them is better.

However, you could start to glean some interesting use cases from their product information or think of your own product in new ways.

It's also a good way of understanding more about the domain you are working in.

# How?

A quick Internet search should bring you back some of your competitor's sites where you should be able to read the marketing material, maybe sign up for a free trial and generally get to know a little more about the product/domain.

# Useful Hint

You may find your product/site listed on 'alternative' sites such as www.alternativeto.net. Sites like this show you alternatives to products so it might be a good way to see what the competition are doing and what people think of their products.

# Useful Links

www.alternativeto.net

# Compliance and Claims

## Why?

Often a product has a release process where some collateral is put together to explain why the product is so good and what features it has. This material (in any format or medium) may contain some claims about features, performance or adherence to compliance.

As a tester it's a good idea to find out, as early as possible, what this material is claiming. You may be lucky enough to have these claims and compliance statements represented in user stories or specifications, often though they are not aligned directly with the product teams work.

Once you have this information you can start to test against these claims.

For example:

- Our product is compliant with X standard ← Really?
- Our product is the fastest on the market ← Really? How is that proven? Where is the data?
- Our product is incredibly secure ← Really?

# How?

Take each claim and compliance and jot down any questions that spring to mind.

It might be as simple as speaking to the product/marketing/management team, or researching each compliance regulation.

Ask questions, seek out more information and always discuss these findings with product/development (the person in charge of releasing stuff).

It could be that all of your questions and concerns fall on deaf ears, in which case give the product a test and find evidence for/against each claim.

Present this to the people that matter, but obviously use judgment on where you spend your time most valuably.

# Useful Hint

If you are testing against standards and laws it might be worth popping a question on user forums or social channels like Twitter (use the #softwaretesting or #testing hashtags).

There will no doubt be countless others working in the same domain who might have some useful insights.

# Useful Links

http://curioustester.blogspot.co.uk/2009/12/claims-testing.html
http://www.bettertesting.co.uk/content/?p=613

# UX

## Why?

UX is becoming a bit of a buzzword recently and is typically associated with usability and design.

In my mind, UX is bigger than that. UX is, as the name suggests, about User Experience. For me, that's about the bigger picture and involves more than just the product. Customer Support, release notes, documentation, supporting docs/sites/help, performance, security, ease of use, usability, accessibility, fit for purpose etc. It's a big topic.

## How?

On the surface there are simple things we can think about regarding UX as Testers. Not all of the elements of UX will be applicable to your context but there is no harm in thinking about some or all of the following:

### Customer Support

Is it available?
Do your support team know the details around the product?
Is there sufficient support/help available to them?
Do they need it?
Are the support contact details correct?
Is there online help?
Is the help correct, accurate and accessible?

### Release Notes

Are there any release notes?

Are they accurate?
Are they complete?
Are they written for the right audience (too technical, not technical enough)?

## Error Tolerant

Is the system error tolerant?
Are the error messages relevant?
Does the system help the user recover from any errors?

## Documentation

Is it there?
Is it needed?
Is it accurate?

## Performance

Is the site performing as per requirements/needs of the customer?
Do we even know this information?

## Security

Is the customer's data safe?
Do they know it is?
Are they re-assured that it is safe?
Can we prove it is safe?

## Usability and Ease of Use

Is the product easy to use?
Can people find what they want quickly?
Is it consistent across all screens/pages?
Is it consistent with expectations?
Is the site behaviour typical based on other systems/sites/applications?

## Accessibility

Is the site accessible by people with accessibility requirements?
Does it need to be?

# Useful Hint

There are a number of resources about design and user behaviour on the web from designers and others working in this exciting domain. They can offer some amazing insights in to how the design of the product can encourage actions and outcomes. As a Tester, don't be afraid to seek out information from sources that aren't considered "testing" specific.

# Useful Links

UX on Wikipedia - http://en.wikipedia.org/wiki/User_experience_design
UX Booth - http://www.uxbooth.com/
UX Mag - http://uxmag.com/

# Change the locale

## Why?

The locale of the browser can play a large part in the rendering of the product and the way that it functions.

It's a simple test idea that could highlight potential multi-language issues, especially if your site actively supports multiple languages.

## How?

Changing your browser locale is different in each browser but a quick web search should give you the information you need.

For example, in Firefox on MAC:

*Firefox > Preferences > Content > Languages*

This will give you a dialogue where you can change the language locale of the browser.

Your browser locale (language) settings will steer your browser to look for "locale specific" content in the sites it loads.

So if your site is intended to be used in English and French markets and will change the language of the home page depending on the browser locale detected, then try removing all locales and see what happens. How about adding en-us instead of en-gb – does it still render in English?

You can quickly start to build up a number of tests and further areas to explore.

# Useful Hint(s)

Try using a Browser Extension to make changing locale easier.
The locale of the browser and the language of the Operating System are two different things.

# Useful Links

Quick Locale switching extension - https://addons.mozilla.org/en-US/firefox/addon/quick-locale-switcher/
Changing Windows language - http://msdn.microsoft.com/en-us/library/ms144258.aspx
Changing MAC Language - http://support.apple.com/kb/HT2490#l6

# Resize the windows and resolution

## Why?

Not all monitors and displays will be the same size and resolution hence your site/application will render differently and potentially behave slightly differently (i.e. buttons moving, flow being interrupted)

With the proliferation of mobile devices like phones and tablets then there is an ever-growing need to ensure your application works on small screens. It may be you have to create mobile specific versions.

## How?

A really simple test would be to change the size of the browser window by simply dragging the edges to shrink the page. Always consider that there may be a minimum size before the layout goes pear shaped. This is to be expected for most non-adaptive designs (and some adaptive ones too) and the point at which this happens will be purely a judgment call on your part.

Another good test would be to change the resolution of your own monitor and see how the site renders.

Try the site and application on a mobile device or tablet. There are emulators out there that can help you out with this.

There are browser add-ons like Firesizer that can help you change the resolution of the page with ease.

## Useful Hint

You could use a Firefox extension to help you resize windows.

## Useful Links

A monster list of emulators - http://www.mobilexweb.com/emulators
Firefox window resizer extension - https://addons.mozilla.org/en-US/firefox/addon/window-resizer/

# Block pop-ups

## Why?

A growing number of sites and applications make use of Pop-Ups to manage form entry, alerts and any number of other processes needed for working through the site/application. They are also used heavily for advertising and marketing.

## How?

Try turning off pop-ups in our browser and see what happens.

Once you have turned them off in the browser, you should get a warning when your site/application tries to open a pop-up.

Can the system still be used without the pop-up?

If you disable pop-ups has it affected the state transition or flow? For example, when a user clicks a button it may open a form in a pop-up and it may make a state change somewhere else in the system, but the form may not show (because its blocked). Can they recover from this? Can they get to the form? Can they continue to function?

## Useful Hint

Try using a browser extension to help you manage pop-ups.

# Useful Links

Chrome Pop-up blocker https://chrome.google.com/webstore/detail/nmpeeekfhbmikbdhlpjbfmnpgcbeggic
Ad Block blocker - https://addons.mozilla.org/en-US/firefox/addon/adblock-plus/

# Disable CSS

## Why?

CSS, or Cascading Style Sheets is a mechanism for adding layout and style to a web page. It should be possible to change the CSS and change the whole look and feel of the website, assuming everything within the application has been coded with this in mind.

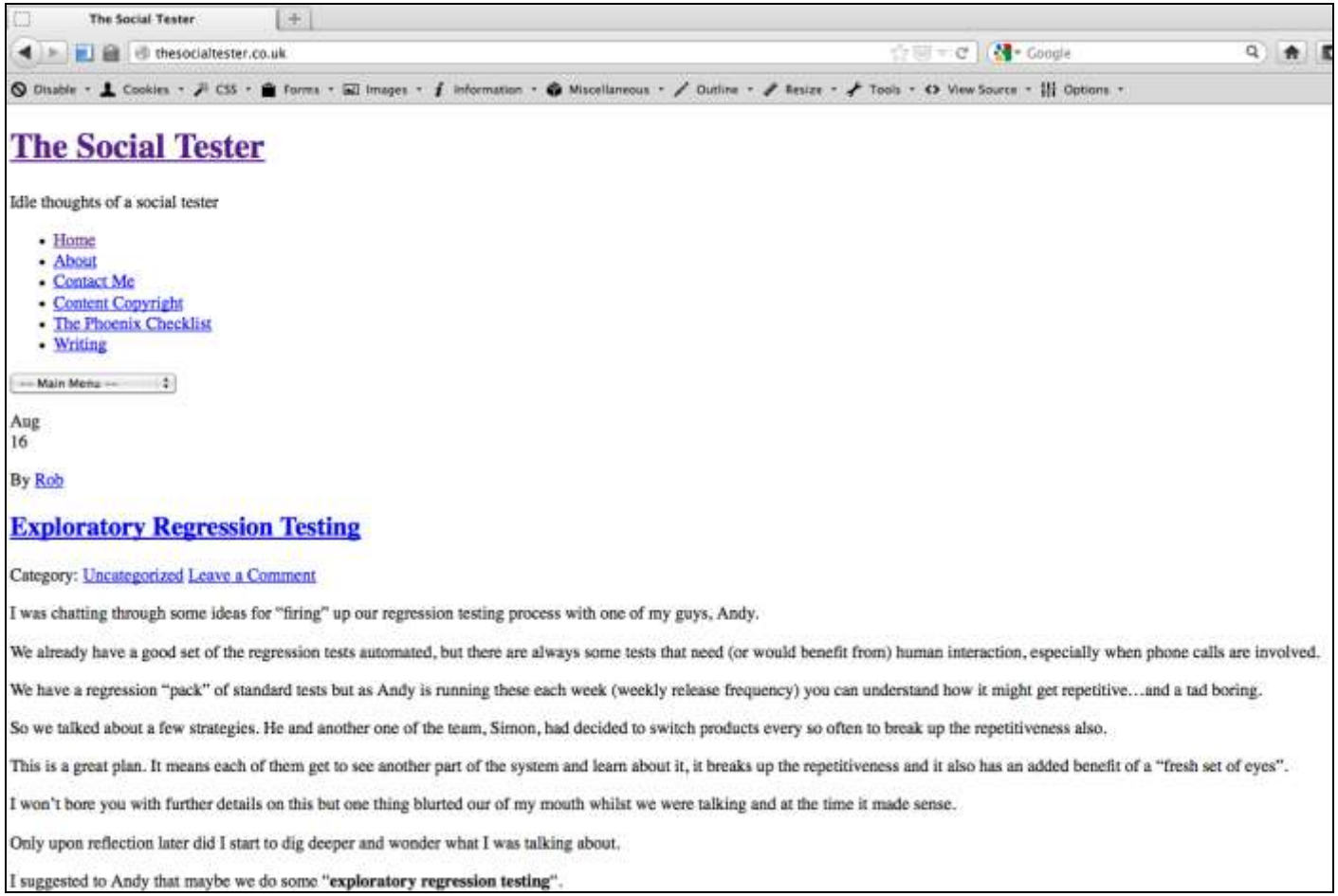
Yet some people, through either choice or necessity, use browsers that disable the CSS. It keeps the site simple and ultimately more accessible for screen reading tools.

As such it's important to test the site with CSS removed. You may be surprised about the flow and order of the site when the CSS is removed.

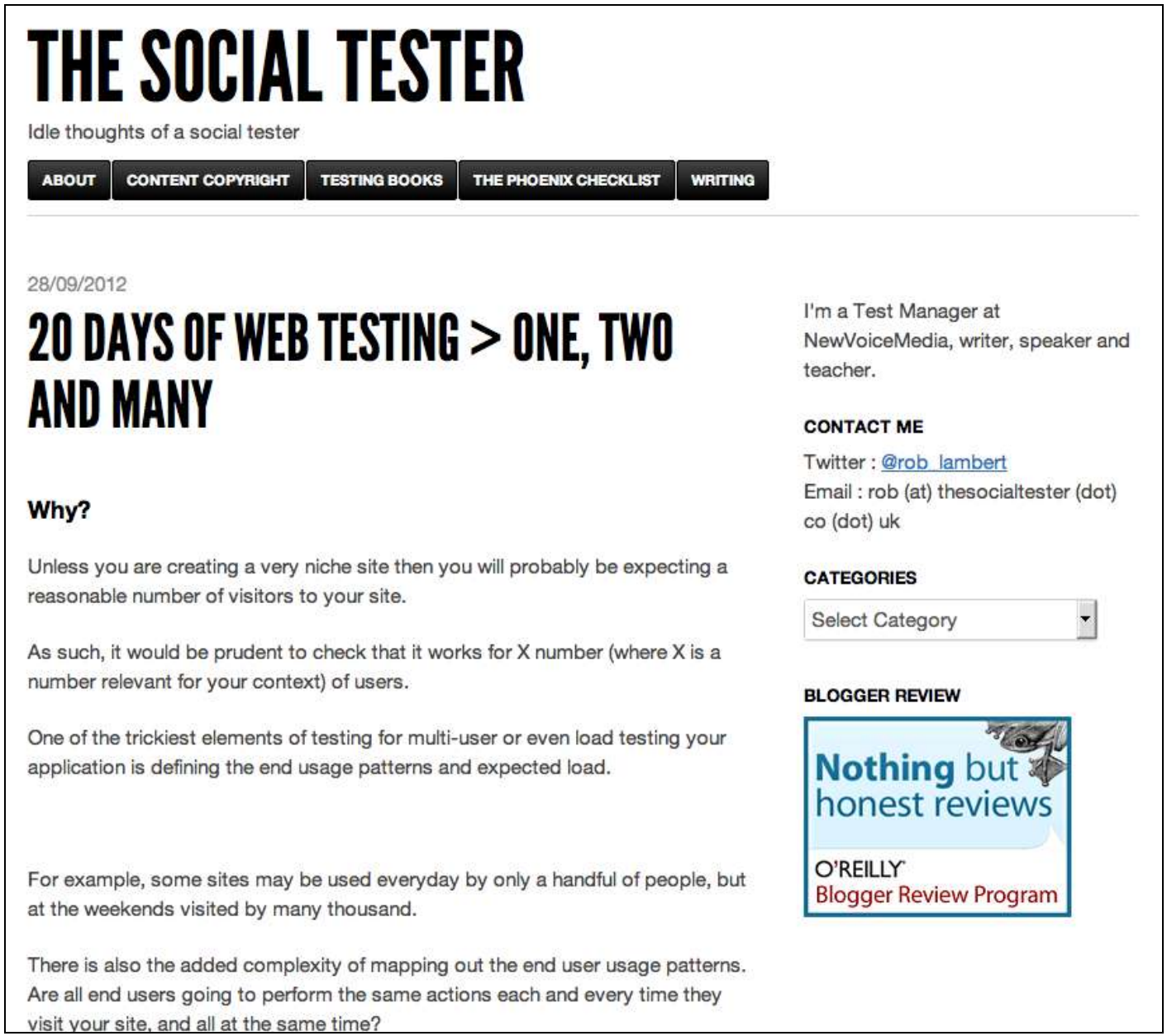
## How?

The simplest way I have found is to open the site in Firefox with the web developer add-on installed. The add-on has an option in the menu to disable CSS. The site will then be rendered with no styling. You could also use Firebug or a number of other tools. With the CSS removed you will typically have a very simple looking site, like the one pictured below:




With the CSS enabled, it looks like this:



# THE SOCIAL TESTER

Idle thoughts of a social tester

ABOUT | CONTENT COPYRIGHT | TESTING BOOKS | THE PHOENIX CHECKLIST | WRITING

28/09/2012

## 20 DAYS OF WEB TESTING > ONE, TWO AND MANY

### Why?

Unless you are creating a very niche site then you will probably be expecting a reasonable number of visitors to your site.

As such, it would be prudent to check that it works for X number (where X is a number relevant for your context) of users.

One of the trickiest elements of testing for multi-user or even load testing your application is defining the end usage patterns and expected load.

For example, some sites may be used everyday by only a handful of people, but at the weekends visited by many thousand.

There is also the added complexity of mapping out the end user usage patterns. Are all end users going to perform the same actions each and every time they visit your site, and all at the same time?

I'm a Test Manager at NewVoiceMedia, writer, speaker and teacher.

**CONTACT ME**

Twitter : @rob_lambert
Email : rob (at) thesocialtester (dot) co (dot) uk

**CATEGORIES**

Select Category

**BLOGGER REVIEW**

Nothing but honest reviews
O'REILLY
Blogger Review Program

# Useful Hint

Firebug has a CSS tab which shows you the CSS for the site. What happens when you change some of the CSS markup?

# Useful Links

W3C standards and advice for CSS - http://www.w3.org/Style/CSS/
Firebug Firefox Extension - http://getfirebug.com/
Web Developer Add-One - https://addons.mozilla.org/en-us/firefox/addon/web-developer/

# Text only

## Why?

As a basic rule it would be good to ensure that the site under test works in text only mode. This is particularly important from an accessibility point of view, but is could be a good goal to aim for from a simplicity point of view too.

It's hard to find sites that work in purely text only mode. Text only mode sites are often external to the main landing page. (i.e. the main site offers a text only parallel site).

There are still mixed views on the benefits of providing a text only version of the site.

## How?

The easiest way to achieve a text only version of a site is to enable the Developer Tools on your browser. For simplicity reasons I have chosen to use the "Developer Tool Bar" for Firefox.

Once installed you need to seek out the option that allows you to turn text only mode on.

In the "Developer Tool Bar" it is found here:

Images > Disable All Images
Then select Disable > Disable JavaScript
Then select CSS > Disable Styles > All Styles

This will result in a completely text only version to test against. You may find you lose some features you need, or the flow is wrong, or you really need the visual symbolism a picture displays to make sense of the page.

# Useful Hint

There are plenty of other options in the Web Developer Tool Bar in Firefox that will give you insights in to how your site might perform on devices that don't support certain features, or for users who are seeking accessible pages.

# Useful Links

Information on Text Only - http://www.webcredible.co.uk/user-friendly-resources/web-accessibility/text-only.shtml
The Developer Tool Bar extension for Firefox - http://chrispederick.com/work/web-developer/

# Disable java script

## Why?

There are often very good reasons to use JavaScript (a client side scripting language which runs in the browser), but sometimes the overuse (or mis-use) of JavaScript can lead to some very interesting bugs.

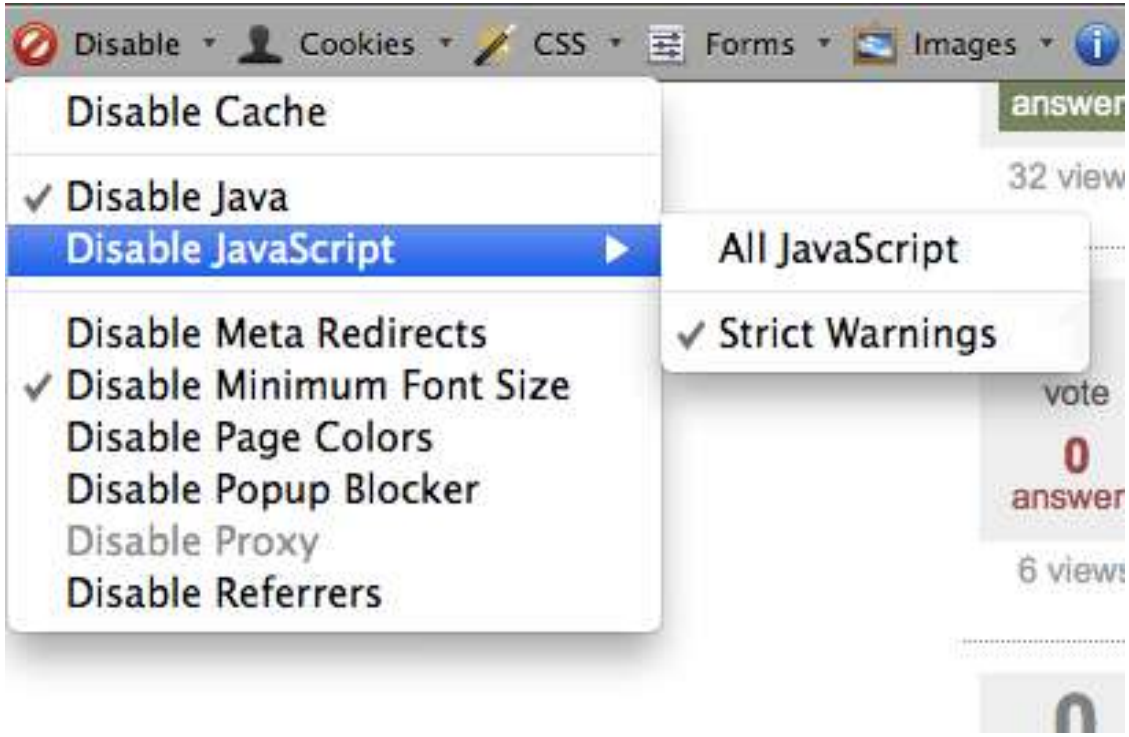
For example, not all browsers accessing your site will have JavaScript enabled which may mean specific parts of your site simply do not work, or certain logic means the product is not running as intended.

JavaScript is sometimes used to collect data, show information, do funky visual stuff whilst browsing (like *mouseover*), perform logic and post form/user data. As such there are a number of features and/or capabilities that could be lost if the user disables JavaScript.

## How?

The simplest test is to Disable JavaScript in your browser using Developer Tools Extension (or similar) and see what happens to your site.




There are other tests you might want to consider such as fiddling with the JavaScript and seeing what affect this has.

# Useful Hint

There are a number of Unit Testing frameworks available for JavaScript.

# Useful Links

JSLint - http://www.jslint.com/lint.html
JavaScript information - http://coding.smashingmagazine.com/2011/10/27/lessons-from-a-review-of-javascript-code/

# Flash free

## Why?

Not all browsers support or are configured to allow Flash, therefore providing a Flash free alternative could be considered a "Must". This is especially so now as Adobe themselves have announced the end of Flash on many mobile platforms.

There's the obvious newsworthy item that the iPad doesn't support Flash but there are also plugins and settings that will actively block Flash in the browser. That's bad news if your entire site is developed in Flash.

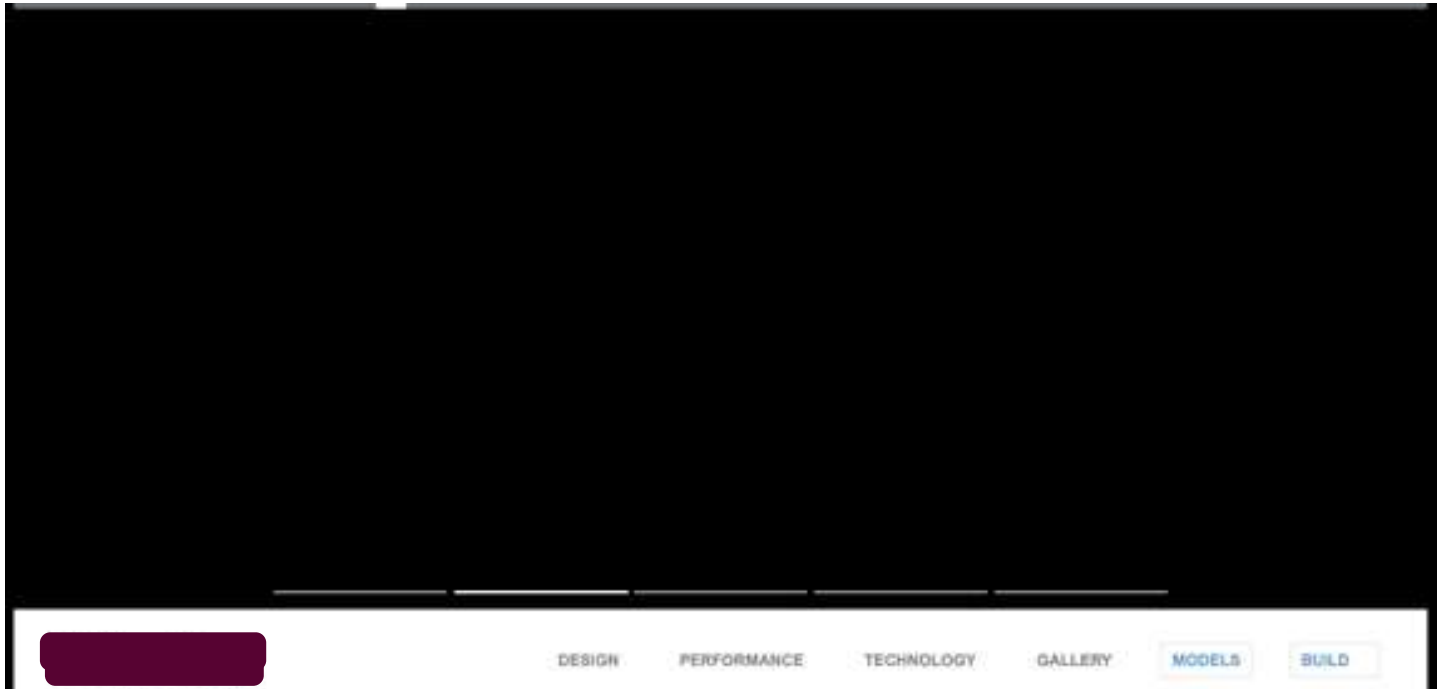
HTML5 is now on the block and very capable of doing the things that Flash became famous for.

## How?

An easy way to test with no Flash is to download one of the Firefox plugins, like Flash Block, and then use the site/application.

The image below shows a car website with flash blocked. In this example it’s blocked the interactive view of the car, pretty much the core reason for the site. In fact, there is very little reason to hang around on this site. Is this the result you want for your end users?




I always work on the simple assumption that I should still be able to perform the main capabilities intended by the site with Flash blocked. (Unless of course the goal of the site is to showcase how flashy your flash is)

If your site MUST be Flash for reasons other than to showcase your Flash skills then a question a Tester must ask is “have we chosen the wrong technology?”

Have you essentially ignored an entire section of your potential market by choosing to use Flash on your site?

# Useful Hint

If you use the Flash block extension for Firefox then visit the preference page to change the settings as it now also supports the blocking of Silverlight, Microsoft’s framework for rich media experience (i.e. – alternative to Flash).

# Useful Links

Flash Block - https://addons.mozilla.org/en-US/firefox/addon/flashblock/
Flash and why you shouldn't use it - http://dynamicwebs.com.au/weblog/?p=33
Flash 99% Bad - http://www.useit.com/alertbox/20001029.html
When and where to use Flash - http://www.businessknowhow.com/internet/flash.htm

# User Acceptance Testing

## Why?

Most people building the product (programmers, testers, product), in my experience, should be considered as “Power Users”. I believe this because in my experience they are competent in using computers, they posses domain knowledge, they are project stakeholders and they typically have insights in to the project and product.

They are often computer savvy and hold significant insight in to the product being built. This would essentially mean they aren't "greenfield" users. They may be biased in many ways towards what the product should do and why they are building it. After all, they are the ones building, testing, marketing and supporting the product.

Greenfield users are users who are coming to your site for the very first time, with little or no prior knowledge of the technology, the design and want to complete certain objectives (i.e. solve a problem or complete an action.)

Sometimes, as Testers, we ignore the end users of our products and assume they possess similar levels of experience when interacting with websites. This is a dangerous assumption.

## How?

The easiest way to get feedback from people outside of the development team is to perform a regular user acceptance program. These can be as large or as small as you see fit, but the more frequent they are in the development process the better. Agile development partly takes care of user acceptance testing assuming that you have a customer available and are releasing regularly.

Long term projects running waterfall, V or other longer releasing methodologies would certainly benefit from more frequent user acceptance testing.

User Acceptance, as stated in that ever-so-contentious ISTQB certification is:

*"Acceptance testing is often the responsibility of the customers or users of a system; other stakeholders may be involved as well.*
*The goal in acceptance testing is to establish confidence in the system, parts of the system or specific non-functional characteristics of the system. Finding defects is not the main focus in acceptance testing."*

There is much to argue with in this statement, but I'll save that for another time. I think we should look differently at this. I think User Acceptance testing can be done at any point in the project.

I think it should be the responsibility of the Development Team (as it aids you in delivering the right product) and I think we should encourage bug finding (depending on what you classify a bug as; enhancement, defect, fault, missing compliance, usability, performance). It is feedback on whether the product you are building is on point, working as expected and relevant to your users. Why would you wait until near completion to get that feedback?

Getting feedback from your end users is critical to your success. Try to get that feedback as early as possible.

# Useful Hint

A UAT phase every few weeks, or months is a great strategy. Get your product in front of the end users as quickly and as often as possible. That way you get feedback early and can include this feedback in your product.

# Useful Links

ISTQB online - http://istqb.org/display/ISTQB/Home

# Mobile friendly?

## Why?

There are a number of stats flying around about mobile usage and mobile device uptake.

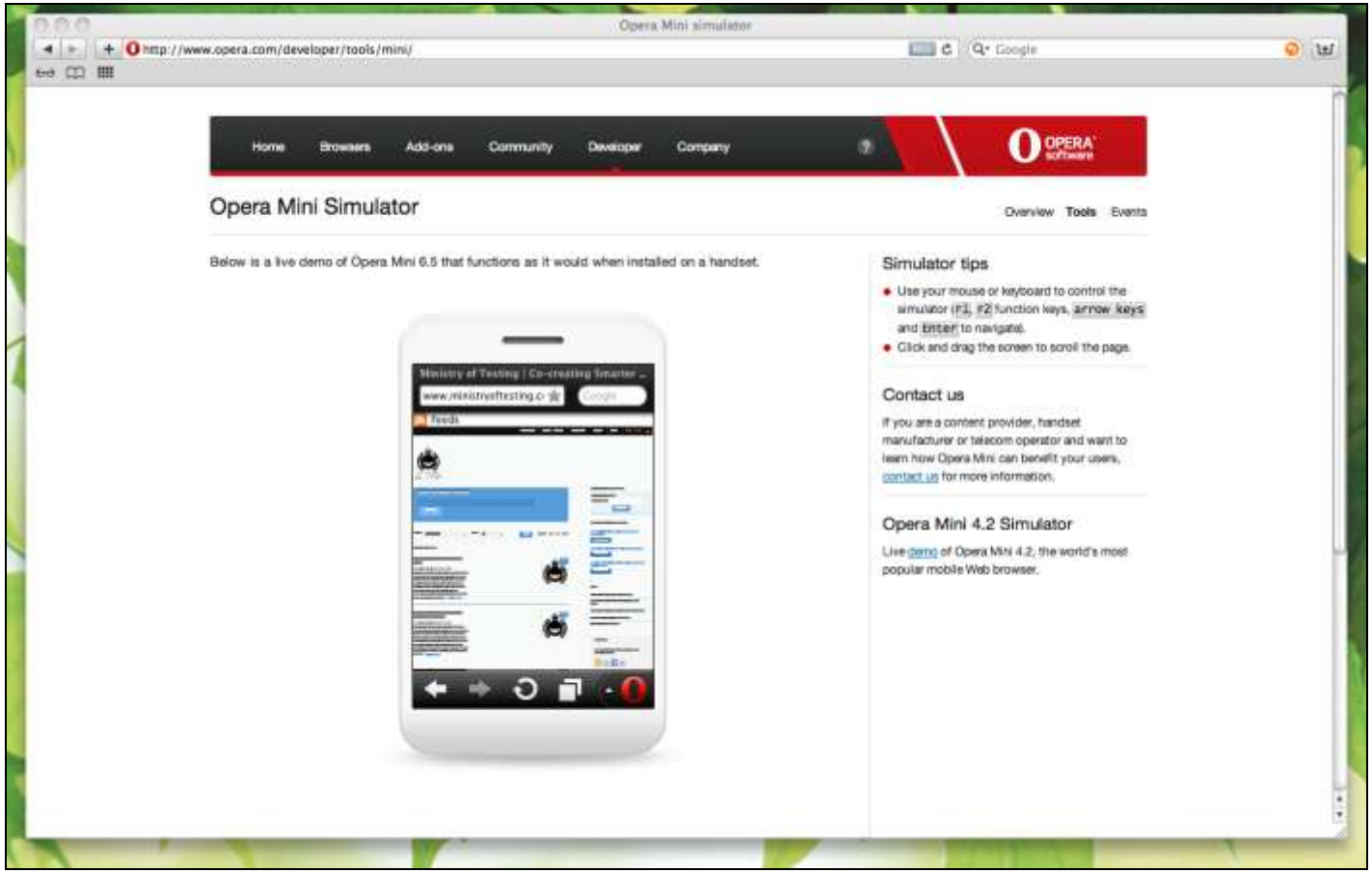
No matter which stat you believe (try a search of “mobile device usage statistics”) it is clear that for many industries, mobile is very much a future we should be looking to embrace.

As a tester it would probably be an idea to try your site or application on a variety of mobile devices and see how well it performs. It could be that your company doesn’t deem mobile a supported platform, but the future looks like it’s going to become very mobile indeed. Who is taking ownership of your mobile support?

## How?

If you have a mobile device yourself then try loading your site/application and see how it performs. There are loads of mobile emulators for you to try your site in too.




It's a good idea to run a stock set of tests against a number of operating systems (mobile and tablet) and build up a profile of what works in which one.

Try right clicks, page updates (like forms that open up further sections of a form), multiple tabs and data selection (drop downs, radio buttons). These are where I have found issues in the past when looking at mobile applications, but each product operates in a different contexts, so tailor it to be specific to what you're working on.

# Useful Hint

Need data on your mobile for testing but don't fancy typing it in all the time? Try sending an SMS or Email with the data in to the phone, then you can copy and paste on the phone itself.

If you're on a smartphone, why not try using Evernote or Dropbox for synching data between?

# Useful Links

A monster list of emulators - http://www.mobilexweb.com/emulators
Micro emulator - http://www.microemu.org/
Android Emulator - http://developer.android.com/guide/developing/tools/emulator.html
Nokia Toolkit - http://www.developer.nokia.com/info/sw.nokia.com/id/d57da811-c7cf-48c8-995f-feb3bea36d11/Nokia_Mobile_Internet_Toolkit_4.1.html
Opera Mobile Emulator - http://www.opera.com/developer/tools/mobile/
Windows Mobile Emulators - http://msdn.microsoft.com/en-us/windowsmobile/bb250560.aspx
iPhone Emulator - http://www.marketcircle.com/iphoney/
Evernote - www.evernote.com

# Race Conditions with Selenium

## Why?

Selenium is a good tool for automating at the UI, but it also has a number of other usages that often go overlooked. Looking for Race Conditions is one of these usages.

Using a tool like Selenium IDE allows you to enter data, click buttons and navigate the site at speeds a human utilising the browser wouldn't be able to do. Sometimes going too fast through a process or application can change the order events/messages should happen, or trigger results at stages they shouldn't be triggering. These are race conditions.

*A race condition or race hazard is a flaw in an electronic system or process whereby the output or result of the process is unexpectedly and critically dependent on the sequence or timing of other events. The term originates with the idea of two signals racing each other to influence the output first*. – Wikipedia - http://en.wikipedia.org/wiki/Race_condition

It's often easy to find race conditions using a Selenium UI test. I've used Selenium to recreate issues at a pace I've not been able to manage through the UI myself.

## How?

Use a Selenium script to play back a sequence of activities at a faster pace than normal.

For example, if you are completing a form and when you choose “yes” to question 3 a further two mandatory questions become available at the end of a form - then you have a good potential for race conditions. The questions may be shown after a quick page re-load (there are other ways of implementing this).

If you work through the form at your own pace and provide a “yes” to question 3 then you see the further questions revealed, and are not able to continue because the two further questions are mandatory. The form validation fires and you receive a warning message to complete the two further questions.

However, if you record the form entry and play it back much faster using Selenium, you may be able to provide the answer “yes” and click continue before the page load happens.

This may mean you are able to submit a form that technically is invalid (you didn’t answer the two further compulsory questions).

I found a bug like this very one in a real system a few years back. It was not reproducible by me clicking all of the time, but Selenium would catch it every time.

There is always an interesting debate regarding whether an end-user would see these types of issues as they aren’t working at the same speeds, but sometimes these Race Conditions point to underlying code issues or process flow problems. They are worth starting a discussion about at least.

Each team will treat these sorts of bugs differently. Some automation frameworks use the concept of “think time” to emulate real user response times. This can be effective for on-going regular automation, but if you’re specifically looking for race conditions then fast automation through the UI can help you.

Some users work faster than others so there always needs to be some judgment applied, but race conditions can occur at any speed, it just so happens Selenium could help you find more of them.

# Useful Hint

Mastering the IDE test controls for Selenium will mean you can move quickly to edit and run tests.

# Useful Links

Race Conditions (Wikipedia) - http://en.wikipedia.org/wiki/Race_condition
Selenium - http://seleniumhq.org/

# Blink Testing

## Why?

I was trying to write an introduction to Blink Testing that was different to James Bach's, but in actual fact his description was always much better. Therefore here it is:

*"What you do in blink testing is plunge yourself into an ocean of data– far too much data to comprehend. And then you comprehend it. Don't know how to do that? Yes you do. But you may not realize that you know how."* *– James Bach – Satisfice -* http://www.satisfice.com/blog/archives/33

## How?

So for example you could work through a form at a leisurely pace (recording in Selenium or a screen capture video tool) filling in each field and then play the script/video back. You may spot layout issues, inconsistencies in labeling, errors on the page or interesting patterns to explore that you hadn't spotted before.

You could record a video whilst you hold down buttons that generate values (like in the video on James Bach's article - http://www.satisfice.com/blog/archives/33, or use it to record an entire exploratory testing session which could be played back at a faster pace afterwards. You may spot things you missed first time around.

When you see the pages and/or data flashing by you will start to observe differences that weren't obvious during your "regular" testing. Selenium is a perfect partner for this, as are screen-recording tools and other automation technologies.

# Useful Hint

When testing audio it can be very effective to play the audio at twice speed. You start to notice patterns or interesting elements that you might not normally have noticed.

# Useful Link

James Bach's article on Blink Testing - http://www.satisfice.com/blog/archives/33
Screen recording for your browser - http://www.screenr.com/
Desktop screen reading software - http://camstudio.org/

# Test in situ

## Why?

I'm a huge fan of doing in-situ testing; which basically means test in the environment your site/application will be used in. Defining the "in-situ" environment could be a challenge for some people as a growing demographic of people, in a multitude of environments, may use their products.

If you are testing public kiosk software - does it work as expected in a crowded, noisy and bright public area? It may work in the test lab but out on the streets are you seeing the same thing? Does your mobile app work when the person using the phone is stood at a bus stop, late at night, with rain coming in sideways at them?

## How?

Try to define some common "in-situ" environments and test your app in them. Planning a series of scenarios can be a good idea. In these scenarios describe the context the users would find themselves in, such as “busy office”, “loud and busy railway station”, “quiet room” etc. These scenarios will help you to understand where and why people are using your products.

I test Call Centre software but there is nothing more valuable than getting on site with some end users (call centre agents and supervisors) and seeing how they use the application. It changes your perspective on a number of elements.

I have a friend who is working on some cool marine navigation tools. He tests on a boat in the ocean when possible; there’s no substitute for the real environment.

Another friend tests medical equipment, they stage fake operations complete with blood, noise and full medical attire. Gives him a whole new perspective on everything when he’s back in the office.

# Useful Hint

Defining personas isn't just the preserve of Designers and Usability experts; it's also a fabulous idea for generating test ideas too.

# Useful Links

In Situ on Wikipedia - http://en.wikipedia.org/wiki/In_situ
Blatant self-promotion link to Social Tester site - http://thesocialtester.co.uk/theres-method-in-the-madness/

# Print it out

## Why?

If your website has forms, applications or offers other user centric data such as receipts, access codes, confirmation codes, then it could be a good idea to print these out as part of your testing.

End users do print out receipts and data and as such, if you found your beautifully functional site offered a terrible print out, then it might spoil the overall customer experience.

For example, when I receive an e-receipt for conference travel I often have to print it out to reclaim expenses, yet you would be amazed at how awful the print formatting can often be.

## How?

Click Print. Check it looks ok. ☺

## Useful Hint

Most browsers offer a print preview. It is often enough to just do this, that way you are not wasting paper.

## Useful Links

Insights in to how to make a print ready CSS - http://answers.oreilly.com/topic/1018-how-to-make-a-web-form-print-ready-with-css/

# Too many extensions

## Why?

Many modern browsers allow you to add add-ons, extensions, or plug-ins which give you increased functionality or allow you to perform actions and activities from within your browser.

A tester's main Firefox add-on is probably Firebug, which allows you to see the traffic between you and the webserver and also do some cool inspection of the page, amongst lots of other things.

There are literally thousands of these add-ons for all of the major browsers. I've discussed many of them in this series.

However it could be that these add-ons also interfere with the product under test causing you to spend time investigating a bug that might not be important to your target end user.

## How?

I had this recently where one of my add-ons was stopping some JavaScript functionality from firing even though there were no settings at all that **should** have prevented it.

This bug was un-reproducible on all other browsers except mine. After exploring all avenues of code we concluded that it might be a Firefox add-on. I had another virtual machine with the same version of Firefox and no add-ons and it worked ok, so the reasoning was sound.

I removed two add-ons at a time, as I typically run with about twenty+ add-ons to aid my testing. Two at a time seemed like a reasonable amount to remove to narrow down which add-on was the problem.

As it turned out it was the last two that I removed that resolved the issue.

I installed them back in the reverse order and the issue re-appeared with the first two add-ons (i.e. – the last two I removed). I narrowed this down to the exact add-ons and did some Internet searching. As it happens it was a minor known issue with that add-ons and some JavaScript implementations.

Add-ons are immensely powerful but they can interfere with functionality.

# Useful Hint

Create a test machine (virtual or real) and install an absolutely clean (i.e. no add-ons or un-needed plug ins) browser and compare the two if you find it tricky to replicate issues. It could be a browser extension problem.

# Useful Links

How to manage add-ons across multiple machines - http://lifehacker.com/272113/sync-your-firefox-extensions-and-profiles-across-computers

# Refresh during page loads

## Why?

If you are testing an application or site that has specific state or transaction model that is updated when you, for example, click on a button then try refreshing the page whilst it's loading these different states.

You'd be amazed at how many potential problems you might find doing this. As one page is loading and sending/retrieving a new set of data the refresh is also then grabbing data, potentially from an old state. The data it grabs might not be the most up-to-date data set.

## How?

Whilst performing actions (like submit form) hit the refresh button. Depending on how fast certain components interact you could find that there are a wealth of issues surrounding states, data and user activities.

## Useful Hint

Find your shortcut key for your operating system to make refreshing quicker.

## Useful Links

Windows shortcut keys - http://support.microsoft.com/kb/126449
MAC shortcut keys - http://support.apple.com/kb/HT1343
Linux shortcut keys - http://linux.about.com/od/linux101/l/blnewbie5_1.htm

# Check for SEO

## Why?

In today's ever growing socially aware society we often need to get hits in to get sales or market share. Getting hits in isn't just a case of providing the right content and hoping marketing gets their campaign right. It's also about creating a site and framework/design that is Search Engine Optimised (SEO).

*"Search engine optimization (SEO) is the process of improving the visibility of a website or a web page in search engines via the "natural," or un-paid ("organic" or "algorithmic"), search results."* – Wikipedia - http://en.wikipedia.org/wiki/SEO

For this you need to do some research on what to do and what not to do with regards to SEO.

There are useful guides, not so useful guides and some in between. There is also disagreement as to the best way to improve SEO and whether it is important or not.

One of the often over-looked elements of making a site good for SEO is that you also make it good for accessibility too. See this slide deck for some interesting comparisons: http://www.slideshare.net/ConeTrees/seo-through-accessibility

## How?

I've included some useful links below but it's always best to find a reliable source on the web, subscribe to it and keep up to date with this fast changing domain.

# Useful Hint

There are countless SEO checker add-ons for your browsers.

# Useful Links

Marketing and Sales look at SEO - http://www.promotionworld.com/articles/se/articles/Internet_Marketing_Strategy/SEO/071007EssentialGuide.html
SEO through accessibility - http://www.slideshare.net/ConeTrees/seo-through-accessibility
SEO at Wikipedia - http://en.wikipedia.org/wiki/SEO

# Five second test

## Why?

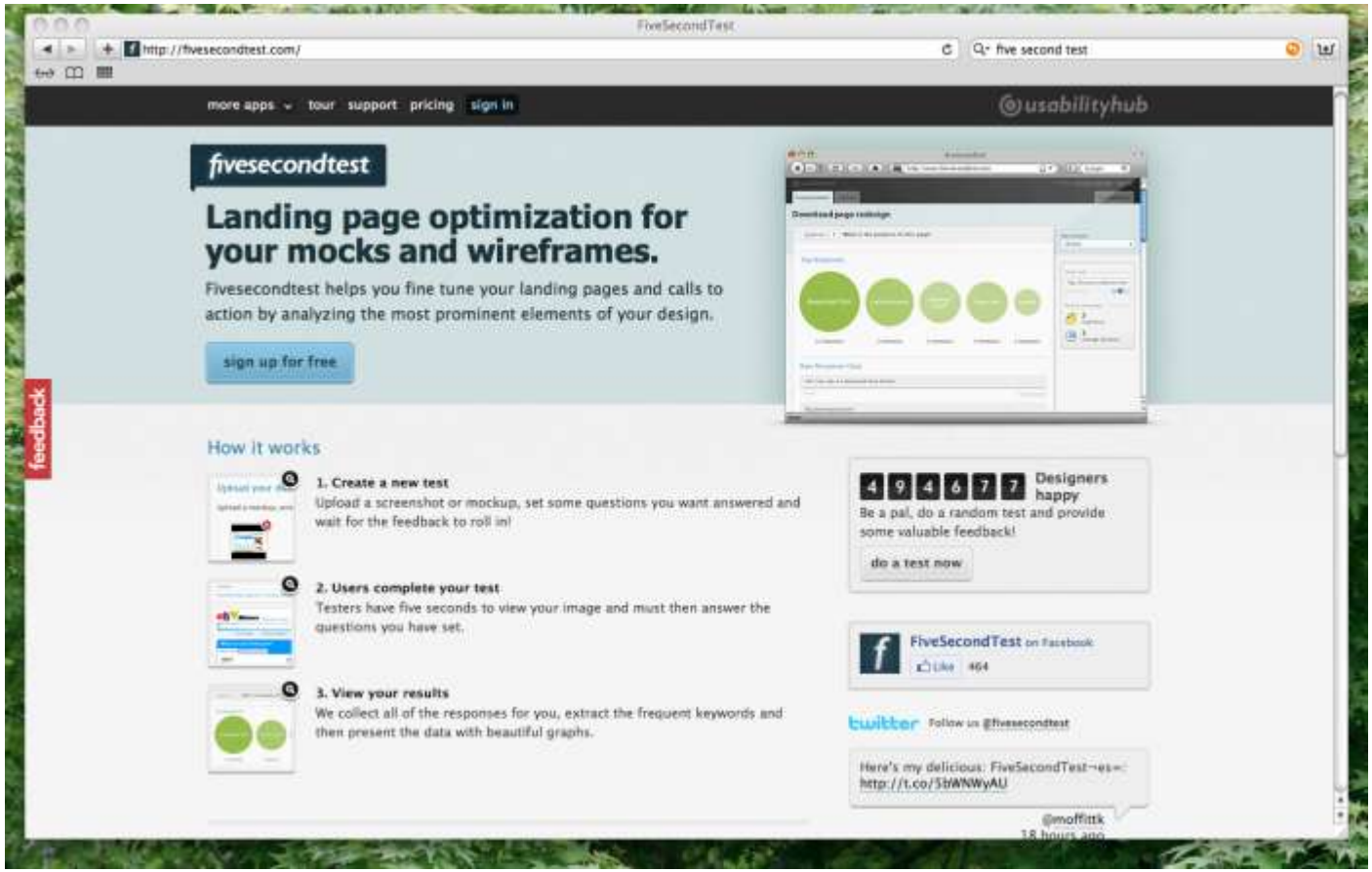
When someone visits your site they often make snap judgments about the site, where to go and whether they like the design (and content) or not.

There is a great service called Five Second Test (http://fivesecondtest.com/) who has users/testers who will visit your site, then answer some questions about the site and give you feedback.

It can help to drive out where your users are focusing their attention when landing on the site.

*"By finding out what a person recalls about your design in just 5 seconds you can ensure that your message is being communicated as effectively as possible." - From Five Second Test Website*




# How?

Simply sign up to the service, upload your site wireframes or mocks and let the feedback commence.

# Useful Hint

Getting your feedback from users early is the best way to ensure you remain focused on building the right application in the first place. Too often I see usability testing done late in the day when it becomes harder to change the code and too much “time” has been invested in a poor design.

# Useful Links

Five Second Test - http://fivesecondtest.com/
24 Usability Tools - http://www.usefulusability.com/24-usability-testing-tools/

# Throttle It

## Why?

When we test applications we are often privy to a fast Internet connections, which means our products under test are tested under pretty good speed and connectivity conditions.

Are your end users all accessing your site through a fast connection?

It's important not to assume too much about your end users. Cold hard facts are great. Assumption however, can lead to designs and implementations that don't suit the purpose, audience or context.

## How?

There are many tools on the market but the one I find most useful is Charles Proxy.

It has a neat "Throttling" feature that is quite straightforward to get running. It allows you to throttle the connection at pre-set speeds as well as creating customised speeds.

Instructions on how to throttle the site are here:
http://www.charlesproxy.com/documentation/proxying/throttling/

Testing with Charles Proxy gives you some indication of what it would be like to interact with your site/application under different connection speeds.

## Useful Hint

Your product specifications/documentation often lead you to the minimum specification required which is useful for customer documentation and support. This is a good starting point for your boundaries.

# Useful Links

Charles Proxy - http://www.charlesproxy.com/

# Thank You

I would like to thank you all for downloading and reading this book.

I expect there will be bugs, typos and general queries about the book so please fire these to me at rob@thesocialtester.co.uk. I will try and respond to all queries.

Please feel free to share this content around but please do not change it and please remember to reference the original source (www.thesocialtester.co.uk)

I was in two minds as to whether to include a link to the ISTQB site. I was certainly in two minds as to whether it constituted a “useful link”. It is included for completion. ☺

I will be around at a few conferences in 2012 and 2013 so please say “hi” if you see me, but be warned – I’m still researching how people take notes (so I may start asking you questions about note-taking) and I am starting a new series of podcasts next year also (so you may get dragged in to a short podcast question and answer session).

Thanks again for your time. It’s appreciated.