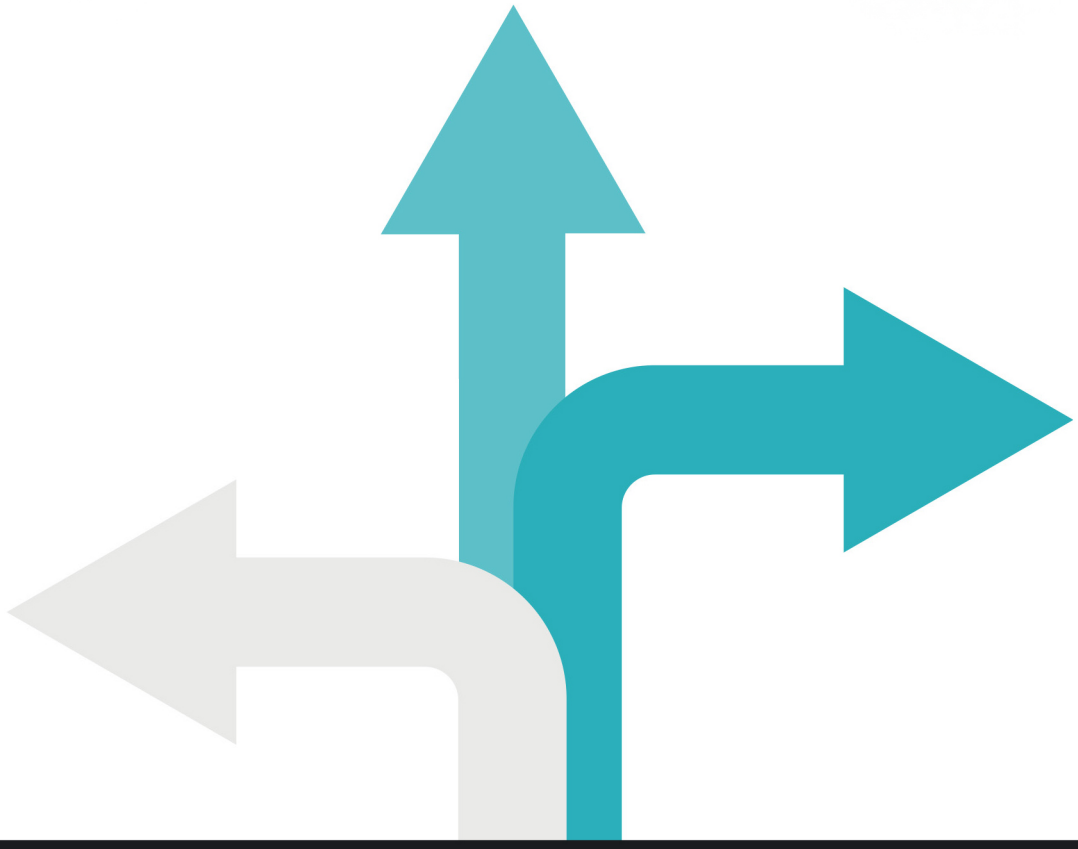
FLEX, SILVERLIGHT OR HTML5?

TIME TO DECIDE...

/ A WHITE PAPER BY COLIN EBERHARDT, TECHNOLOGY DIRECTOR



SCOTT LOGIC

ALTOGETHER SMARTER

/ OVERVIEW

Recent advances in web technologies have resulted in a complex landscape for application developers to navigate. Coupled with the recent boom in new platforms, from desktops and netbooks to smartphones and tablets, making an informed and future-proof technology choice is all the more difficult. In this paper we will set technology bias and politics aside to navigate the similarities and differences between Flex, Silverlight and HTML5 to give you the power to decide.

This whitepaper is aimed at technical decision makers who are looking to choose the correct technology for web application development. We begin by looking at the current web development landscape and the complex challenges it presents.

We consider the evolution of the Internet as a platform for application delivery, following the parallel paths of HTML and plug-ins. We examine the strengths and weaknesses of Flex, Silverlight and HTML5, which are widely accepted as the most important technologies for web application development.

The strengths and weaknesses of each technology are combined with application complexity allowing the most suitable technology to be selected for a particular application. Finally, we will look to the future of each technology and how the landscape is likely to shift over the next few years.

/ CONTENTS

INTRODUCTION

OVER THE PAST COUPLE OF DECADES THE WEB HAS BEEN TRANSFORMED FROM A LARGELY STATIC COLLECTION OF INTERCONNECTED WEB PAGES, INTO A RICH AND INTERACTIVE PLATFORM FOR DELIVERY OF CONTENT, DATA AND SERVICES.

The traditional, desktop-based, model of application distribution is heavyweight, requiring a user to install software onto each machine which they wish to use it on, often with different versions required for different operating systems. In contrast the web is much more lightweight, allowing a user to access their data and services with little more than an internet connection and computer (or smartphone).

However, for all its potential, the web presents a complex landscape, made up of a mix of standards, legacy formats, browsers and their inherent differences and incompatibilities, third-party plugins and other assorted innovations. For desktop development the technology choices are often few and simple; for web development there are many more choices to be made, and each comes with its own set of compromises.

Currently there is no widely accepted single right choice for web development, and if you skip to the conclusion of this paper, you will not find that it presents a single technology as the right answer either. What this paper does do is describe the current and emerging technologies together with their strengths and weaknesses. More importantly, the conclusions describe the suitability of each technology for the development of web based applications based on application complexity, allowing you to make a more informed choice.

This paper is intended to help technology decision makers within the financial sector. This focus allows us to make a few generalisations which make the technology comparisons a bit easier. Features such as audio capabilities, DRM and high-definition video, whilst important in the broader consumer market, are of little importance in most financial applications, therefore these features will not be included in the comparison.

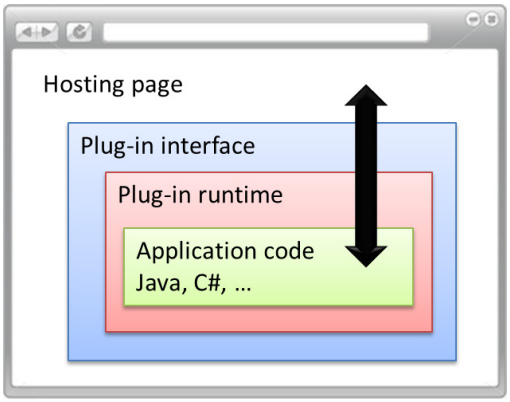
THE INTERACTIVE WEB

IN THE 1990S HTML AND JAVASCRIPT WERE IN THEIR INFANCY, LACKING THE FEATURES REQUIRED TO BUILD APPLICATIONS AND DELIVER INTERACTIVE CONTENT. EARLY WEB APPLICATIONS RELIED HEAVILY ON SERVER-SIDE CODE TO DELIVER DYNAMIC CONTENT, MAKING THE SOLUTIONS SLOW AND CUMBERSOME TO USE.

The mid-to-late 1990s saw a rise in popularity of browser plugins, downloadable browser extensions which supplement its core HTML functionality. Plugins are still used to this day to render content types not supported directly by the browser, for example PDF documents and video.

A more flexible approach to extending browser functionality than 'single function' plugins is to provide a plugin which acts as a separate runtime. Application code can be written that targets the plugin runtime providing an alternative to HTML / JavaScript technologies for web application development. This is the approach that was used to bring Java applications to the web, with a Java Virtual Machine plugin providing the runtime for Java Applets. This is also the approach used by the Flash and Silverlight plugins, each providing a runtime for their native content.

Probably the biggest advantage of the plugin model is that it gives the plugin developers a sandbox free from the browser itself. Therefore it is also free from the constraints of web standards and the associated issues of cross-browser standards support, allowing plugins to provide much more 'power' than the HTML page that hosts them. However, there are some disadvantages to the plugin model; if you develop an application that runs within a plugin you are reliant on the end user actually having the target plugin installed on their browser. Also, in the context of mixed content pages, which have a combination of static and dynamic content, you often have to work across the plugin boundaries, integrating with the HTML and JavaScript running elsewhere on the page.



BROWSER PLUGINS CAN BRING ESTABLISHED LANGUAGES TO THE WEB

/ THE PLUGIN MODEL

The plugin model enjoyed early success with Java Applets and Flash, both of which had high adoption in the late 1990s (and in Flash's case still does), with Java Applets typically being the plugin of choice for business applications, and Flash being used for games, adverts and web site 'splash screens'1. During this period HTML and JavaScript were maturing and innovations were being standardised, resulting in a much more powerful technology.

HTML and JavaScript evolved into a technology pairing which could be used for application development in early 2000. It was around this time the term AJAX was coined; AJAX stands for Asynchronous JavaScript and XML (although AJAX is more often used to fetch HTML or JSON data asynchronously). AJAX is not a specific technology or framework, rather, it describes an approach which departs from the old HTML model where user interaction results in a refresh of the entire page, i.e. navigation. With AJAX, user interactions result in an asynchronous request to the server, with the response being used to dynamically change the state of the page which the browser is currently rendering.

The ever-increasing power of HTML / JavaScript has provided developers with the tools they require to develop applications that are delivered via the browser, rather than the desktop, leading to a recent boom in Rich Internet Applications (or RIAs). However, the plugin model is still very much alive today, with Flash and Flex going strong, and the recent release of Silverlight, a newcomer to the plugin market.

The web has come a long way from its origins as an interconnected network of static pages; the technologies have become more powerful, as have the possibilities. However, choosing the right technology is still a challenge. In the next section we look at the three main players in the RIA space to see how their features stack up.

The Plugin Model
PROs • Plugins provide a predictable runtime with few cross-browser issues. • They can provide a vehicle for bringing well established technologies such as Java and .NET to the web. • The plugin runtime can be upgraded, whereas upgraded HTML support typically requires the user to install a new browser
CONs • Plugins need to be downloaded by the end user. • Plugins create 'islands' of interactivity within a HTML / JavaScript page • They have an associated load-time; HTML is much more immediate. • In portal applications, JavaScript is still needed to interact with the rest of the page

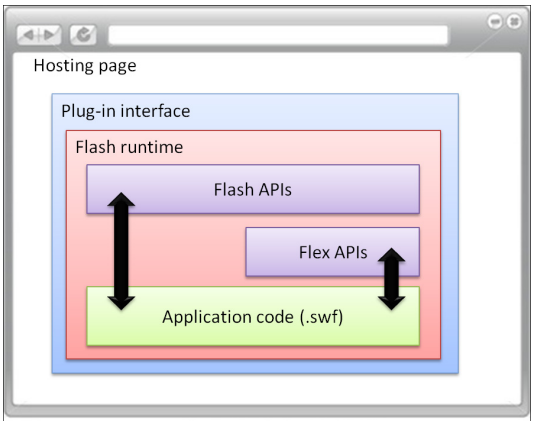
THE BIG THREE

/ FLEX

The first version of the Flash plugin was released by Macromedia in 1996, providing vector graphics and timeline-based animations. Flash-based content added vibrancy to what was quite a static web experience at the time, constrained both by the HTML technology and limited internet bandwidth. Flash found great success with its use in games, banner advertising and interactive product demonstrations; however the timeline-based nature of Flash made it difficult to use it for the development of applications.

At around the time that AJAX was rising in popularity, the first version of Flex was released. Flex is a layer that sits on top of Flash, providing a programming model that is more familiar for application developers. The user interface can be defined declaratively using MXML, an XML based markup language. A suite of common user interface components such as buttons, lists, trees and grids are supplied. The interaction logic is developed in ActionScript, a strongly-typed object-oriented language which shares the same syntax as JavaScript. Development is typically undertaken within the FlashBuilder IDE, an Eclipse-based development environment. This provides a familiar environment for Java developers. Flex provides further integration with the Java via a number of different solutions, including, for example, the LiveCycleDS [1] dataservices component, which provides communications between Flex applications and server-side Java components.

Because Flex is an abstraction layer which executes against the Flash runtime, a Flex application will typically include the Flex libraries as part of their distribution, adding to the overall footprint of a Flex-based solution.



ADOBE FLEX IS A SET OF APPLICATION APIS BUILT ON FLASH

/ FLEX CONT.

Probably the biggest advantage that Flex has over Silverlight and other browser-based plugin technologies such as Java Applets is its ubiquity. Most sources of statistics indicate the presence of a recent version of the Flash plugin is around 95% of user's browsers2. Flex is also quite a mature technology, having a proven track-record since its release in 2004. It has been used to create numerous financial tools, including consumer sites such as mint [2] and Morgan Stanley's Matrix [3].

One weakness of Flex is its single threaded execution model, however, network IO requests are asynchronous and non-blocking, and hence this issue is not as significant as it first sounds; highly complex applications have comfortably been built on top of the Flex framework. Another potential problem with Flex development is the shortage of Flex developers [5] [6]. A small weakness of Flex is that the Flex libraries themselves have to be included within the application download, adding ~150Kbytes to any Flex based content.

FLASH PLAYER ADOPTION STATISTICS (MAR 2011)

Flex	
PROs	**CONs**
• Maturity / ubiquity • Predictable runtime • An Object Oriented language and familiar tools for Java developers • Data services for Java	• Single threaded • Skills availability • Heavyweight – Flex libraries required • Interaction with HTML requires JavaScript

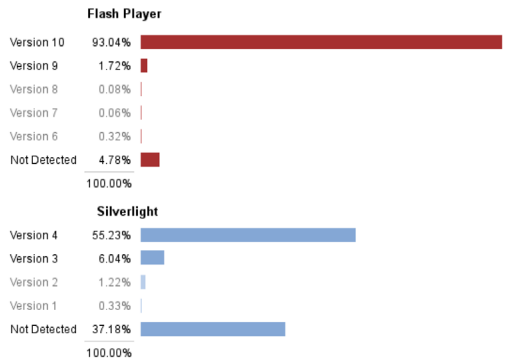
/ SILVERLIGHT

Silverlight is a framework for web application development which is based on a stripped-down version of the .NET framework. The run-time environment for Silverlight applications is a browser-based plugin, which must be downloaded and installed before Silverlight content can be viewed. Silverlight is a relatively new web technology with version 2.0, the first version to include the .NET runtime, being released in 2008. The plugin provides a Common Language Runtime (CLR), a virtual-machine and just-in-time compiler, which allows it to execute applications written in any .NET language (typically C# or VB.NET).

The Silverlight framework is based on Windows Presentation Foundation (WPF), the latest .NET UI technology. The user interface is defined in XAML, an XML based markup language, with the interaction logic written in C# or VB.NET. Various other .NET frameworks are available within Silverlight, such as Linq, WCF and RIA services. Silverlight applications are developed using Visual Studio, the same IDE which developers use for ASP.NET, WPF and other Microsoft frameworks. The XAML markup can also be edited in Expression Blend, a tool geared towards graphics designers. One of the main advantages of Silverlight is that the language, developer tools and frameworks are familiar to developers who have worked with other .NET frameworks.

Probably the single biggest disadvantage Silverlight has when compared to Flex is the current adoption. The current statistics indicate that 65% of browsers have the most recent Silverlight plugin installed, compared to 95% for Flash [4]. This means that approximately one-third of users would have to download the Silverlight plugin in order to view Silverlight content.

Silverlight is a relatively complex framework when compared to other .NET technologies such as Windows Forms and has not matured to the point of having the same level of tool support for rapid application development.



FLASH PLAYER AND SILVERLIGHT ADOPTION STATISTICS (MAR 2011) [4]

Silverlight	
PROs	**CONs**
• .NET framework / Visual Studio • Developer availability • Multithreaded • Powerful styling	• Current adoption • Maturity / longevity • Interaction with HTML requires JavaScript • Framework complexity

/ HTML5

While Silverlight and Flex are quite similar in that both use a browser plugin to execute their native code, HTML5 is very different, being the next evolutionary step of HTML, the markup used by all websites today.

Before getting into the details of HTML5 it is probably worth describing the way in which this technology evolves and is managed. Unlike Silverlight and Flex which are controlled exclusively by Microsoft and Adobe respectively, HTML is guided by the members of the World Wide Web Consortium (W3C) [7], which is comprised of browser manufacturers, web developers, academic and other interested parties.

The evolution of HTML has come as a result of the competing forces of innovation and standardisation. Whilst innovation and competition drive the technology forward, adding new capabilities to HTML and its associated technologies of JavaScript and CSS, standardisation tries to ensure that the web developer can expect the same features and APIs across various browsers. These two forces need to be delicately balanced. HTML5 emerged at a time when this balance was tipped slightly too far towards standardisation.

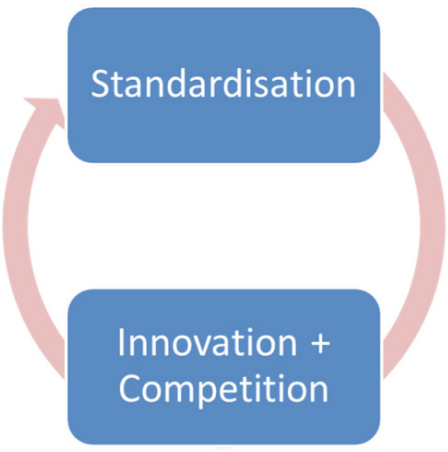
Browsers are designed to be 'forgiving', rendering non-standard (i.e. 'broken') HTML as best they can. However, this has led authors to create millions of broken web sites. As each new browser was released and new HTML standards evolved, manufacturers have had to ensure that their browsers were still compatible with these broken web pages. This has led to them investing thousands of hours of effort trying to make their product mimic the error handling of older versions and that of other browsers.

From 1999-2006 W3C were pushing for a more strict HTML structure based on XML, in an attempt to combat the issue of the millions of broken web pages on the internet. The XHTML2.0 standard they were proposing would enforce correctness, in the hope that future developers would take more care when developing web sites. However, this was not a popular move among web developers, and as a result this specification was not implemented in any major browser.

Whilst W3C were pushing for a stricter specification, a separate group, the WHAT Working Group [8], worked on an alternative standard based on backward compatibility, migration and consistent error handling. The backward compatibility made web developers happier, and the error handling specification made browser manufacturers happy. The specification grew to add a number of exciting new features to HTML and eventually became HTML5. In the end W3C dropped XHTML2 and gave HTML5 their full backing [9]. The balance between standardisation and innovation was restored. HTML5 is a collection of various new APIs and features for web application developers, tackling the gap that Flex and Silverlight fill. HTML5 briefly comprises:

- A <canvas> element, for immediate-mode 2D graphics
- Local storage
- Web Workers, a technique for running scripts as a background task, providing multi-threaded capabilities
- Web Sockets for bi-directional communication
- CSS3, including new layout concepts, opacity, gradient fills and animation
- A much faster JavaScript engine
- New semantic tags

Despite the HTML5 specification being far from complete, modern browsers are already adopting various parts of the specification, such is the evolutionary nature of HTML. However, older browsers must be upgraded to a newer version to add feature support, in contrast to the plugin model, where only the plugin runtime needs to be upgraded.



/ HTML5 CONT.

In order to use the new HTML5 features, developers need to be aware of the level of support for the particular feature they wish to use. In general terms modern browsers like Chrome and Firefox have support for many of these new features, whereas Internet Explorer does not have a good track record for feature support. Although with Microsoft starting to embrace HTML5 this looks to change with the latest version of Internet Explorer (v9) doing a much better job of HTML5 support than its predecessors [10].

In order to determine how well-supported a specific HTML5 feature is, you need to first look at browser support. Then, to convert this into real-world figures, combine this with the usage statistics of the browsers themselves. As you can see from the above table, HTML5 feature support varies considerably across browsers. There are some HTML5 features such as canvas which are quite widely supported and are being actively used today. However, it should be noted that for IE8 or below, ironically, you need to install a plugin to support canvas [12]. There are other HTML5 features, such as WebSockets, that are in the very early stages of adoption. If a feature of HTML5 is not supported by all the browsers developers wish to target, it doesn't mean they cannot use this feature at all. It is quite a simple task to detect whether a browser has support for a specific HTML5 feature, and if not, degrade gracefully by using some other technique for achieving the same task, or degrading the user experience by dropping features.

Developers rarely develop JavaScript browser applications without the addition of various frameworks and toolkits. These add a significant productivity boost by providing the developer with a more powerful abstraction layer that normalises browser incompatibilities. There are various tools and frameworks that help the developer detect support for HTML5 features and some provide abstractions that degrade by using older browser features, for example from <canvas> to SVG graphics.

	IE	Firefox	Safari	Chrome	Opera	iPhone	Android	Reach
Canvas	6.0[3]	3.0	3.0	3.0	10.0	1.0	1.0	97%
local storage	8.0	3.5	4.0	4.0	10.5	2.0	2.0	81%
Video H.264			3.0	5.0		3.0	2.0	20%
Geolocation		3.5	5.0	5.0		3.0	2.0	49%
Form features		3.7	4.0	4.0	10.0			23%
WebWorkers		3.5	4.0	3.0	10.6			50%
WebSockets		4.0 (beta)	5.0.2	4.0	11.0			24%

THE SUPPORT FOR VARIOUS HTML5 FEATURES ACROSS BROWSER VERSIONS [11] WITH 'REACH' DETERMINED BY COMBINING THIS WITH BROWSER VERSION ADOPTION [4]

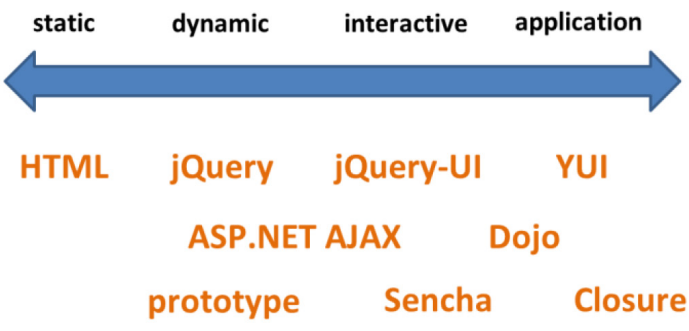
As you can see from the above table, HTML5 feature support varies considerably across browsers. There are some HTML5 features such as canvas which are quite widely supported and are being actively used today. However, it should be noted that for IE8 or below, ironically, you need to install a plugin to support canvas [12]. There are other HTML5 features, such as WebSockets, that are in the very early stages of adoption.

If a feature of HTML5 is not supported by all the browsers developers wish to target, it doesn't mean they cannot use this feature at all. It is quite a simple task to detect whether a browser has support for a specific HTML5 feature, and if not, degrade gracefully by using some other technique for achieving the same task, or degrading the user experience by dropping features.

Developers rarely develop JavaScript browser applications without the addition of various frameworks and toolkits. These add a significant productivity boost by providing the developer with a more powerful abstraction layer that normalises browser incompatibilities. There are various tools and frameworks that help the developer detect support for HTML5 features and some provide abstractions that degrade by using older browser features, for example from <canvas> to SVG graphics.

One of the biggest advantages that HTML5 has over Silverlight and Flex is that it does not require a plugin. This means that the user does not have to install any other software to view a HTML5 page, and also results in faster load times.

Out of all three technologies, HTML has the farthest potential reach, with newer form factors like mobile and tablet quickly adopting the latest HTML5 features. With billions of websites already online, all of which use HTML, the future of HTML itself is quite secure. A webpage written in HTML is far more likely to still be useable in a decade, than one written in Flex or Silverlight.



HTML5 / JAVASCRIPT ABSTRACTION LAYERS AND FRAMEWORKS

/ HTML5 CONT.

For mixed-content sites with embedded 'widgets' in an otherwise static page, widgets developed in Flex and Silverlight need to bridge the technology divide to communicate with the JavaScript / HTML that forms the rest of the page. With HTML5 the application does not sit within the page, it is the page.

Some of the disadvantages of HTML5 are that a user needs a modern browser to experience the majority of the features this technology offers. An application can detect support and degrade gracefully, but this is inferior to Flex / Silverlight where a predictable environment is always provided. Also, the JavaScript language is not Object Oriented and doesn't have the same formal structure that Java and C# impose.

This and the lack of similar tool support to Silverlight / Flex, means that developers need to be more skilled to develop quality HTML5 applications. Ultimately this pushes up the cost of development for a HTML5 application, compared to an equivalent created with Silverlight or Flex.

HTML5	
PROs	**CONs**
• No plugin = lightweight • Future proof • Maximum reach (browser / OS / platform) • Multithreaded • CSS / HTML are designer friendly	• JavaScript language • Features not present in old browsers • Developer tools not as advanced as Flex and Silverlight • Lack of maturity • Skills availability

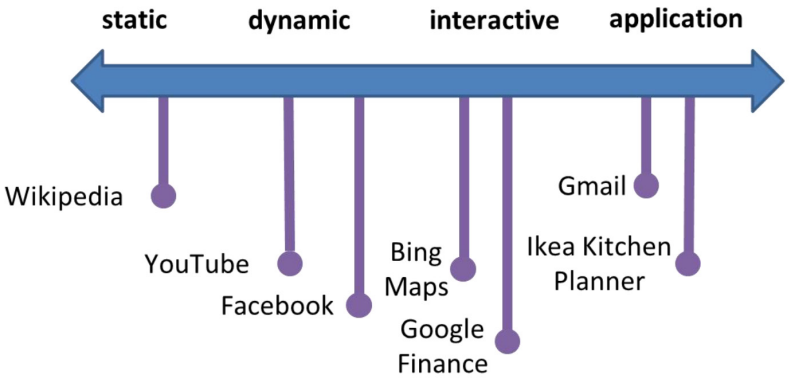
TIME TO DECIDE

SO FAR WE HAVE LOOKED IN SOME DETAIL AT THE THREE TECHNOLOGIES, FLEX, HTML5 AND SILVERLIGHT, HIGHLIGHTING THE MAJOR STRENGTHS AND WEAKNESSES OF EACH.

/ WEB APPLICATION DEVELOPMENT

As an initial broad-brush comparison it is fair to say that Flex and Silverlight are similar technologies, while HTML5 is something quite different, but more on this later. In the introduction it was mentioned that there was no clear winner and no single right choice, however it does not mean that this comparison is in vain. Fortunately every application is different, and this has an impact on the suitability of each technology for that application. Furthermore, every development team is different, with a mix of experience, skills and competencies, and again this has an important impact on the technology choice.

In order to differentiate between different web sites and applications we will start by looking at a single metric: complexity. The most basic, least complex web sites are comprised of largely static content, they may have complex server-side logic generating and maintaining this content, but once it reaches the browser it does not change, resembling the pages of a book. As the complexity increases, web sites become more dynamic, with user interactions resulting in small changes to the page currently being displayed. Further increases in complexity result in truly interactive, or immersive web sites where there is no longer the concept of a page. Finally, at the most complex end of the spectrum we have applications, which perform complex and useful business functions and often feel somewhat disconnected from the web.



Starting at the simplest end of this spectrum, the static website, this is where the internet began. Static websites are still predominantly written in HTML and its associated technologies. Will HTML still dominate in the future? Firstly, it is worth looking at whether Silverlight or Flex have anything to offer the static website. The answer is a definite 'no'. Flex and Silverlight do not have any innovations that would add significant value to static websites, but more importantly, the plugin model adds an extra load time to the page, disrupting the browsing experience. Having discounted the two plugin technologies, this leaves HTML5. Will it make an impact here? In this case, the answer is a definite 'yes'. HTML5 adds a lot of value to static web pages with improved accessibility and more powerful CSS techniques.

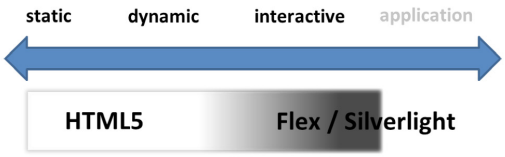
Ramping up the complexity; dynamic websites are similar to static web pages in that they still follow the standard navigation model, with hyperlinks loading a new page from the server. However, the individual pages are more interactive; for example, the user can post comments, sort tables, page through data, without a complete page refresh.

This type of website gave rise to the classic AJAX patterns. Again, HTML5 offers a natural evolution for developers of these pages. There are well established toolkits that provide asynchronous requests, sorted tables etc… which are readily available. Flex and Silverlight are also well-equipped to deliver dynamic content, with controls that are dynamic and interactive 'out of the box'. Despite all three technologies supporting dynamic UIs, HTML5 would most likely emerge as the winner due to its superior static document layout and lightweight approach.

With interactive / immersive web pages, we start to lose the concept of a page altogether, as the website starts to look more like an application. These sites might include more significant business logic and validation, charting and diagrams. These higher level UI constructs are not present in HTML5, so must be built from scratch or provided as part of a framework, and there are indeed a range of frameworks available which service this need.

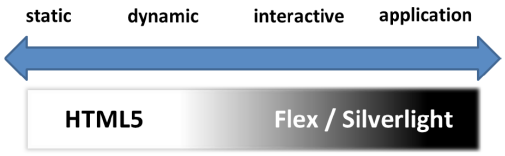
Both Flex and Silverlight were designed with these higher level UI constructs in mind, with suites of controls available to build your application, and standard patterns and practices for validation and separation of concerns. In terms of what can ultimately be achieved with each technology it would seem that all three are equally matched, however, in practice the development of an equivalent interactive website with HTML5 would typically take longer than development of an equivalent with Silverlight / Flex. This is mostly due to the differences between the plugin approach, which gives a predictable and consistent runtime environment, whereas HTML5 applications require more attention to cross-browser support of features and capabilities.

However, this does not put HTML5 out of the picture. The added 'reach' of the standards-based HTML5 does give it some advantages over the two plugin technologies, especially if the application is intended for the general consumer.



Finally, at the most complex end of the spectrum, fully-fledged web applications, the benefits of Flex / Silverlight start to pay dividends. The combination of a mature language and development environment, third party controls and UI components, modular application frameworks coupled with a predictable runtime make it a much easier task to develop highly complex applications with Flex / Silverlight. However, again, this does not mean that it is not possible to develop complex applications with HTML5. As an example, Google Documents demonstrates just how far HTML has come from its beginning as a static document markup.

With the full spectrum of web site / application complexity covered, what does the overall landscape look like? Unfortunately it is a bit messy, with HTML at one end of the spectrum and Flex / Silverlight at the other end, with a large grey area in between.



Whilst it is possible to use any of the three technologies to produce a web site / application at any point on the above spectrum, what we find is that each has their own 'comfort zone'. If a technology is used outside of its 'comfort zone' either the development becomes harder and more costly or the application delivered compromises on features. For example a Silverlight static website might look fine, but the load times would frustrate users, whereas a HTML5 3D kitchen planner would likely be costly to develop and only work satisfactorily on a small subset of browsers.

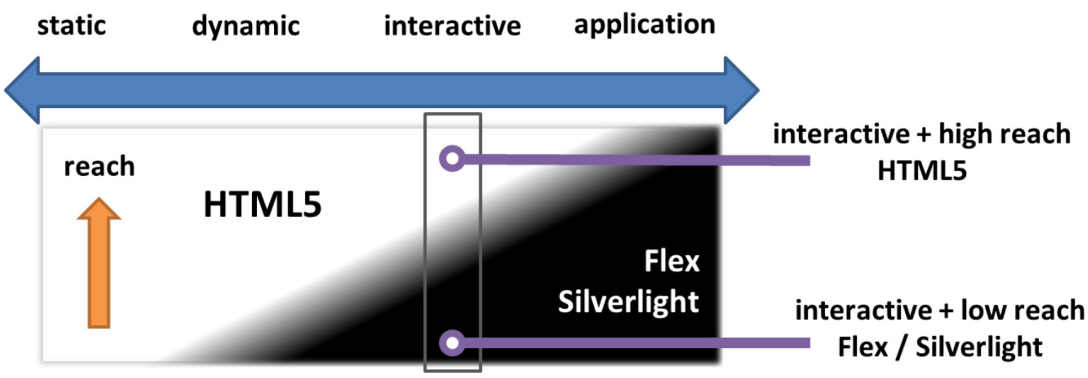
/ FINE TUNING OUR DECISION

If complexity alone still leaves a large grey area for technology choice, further factors need to be considered which affect the 'comfort zone' for a specific development. As stated previously, fortunately each application and each development team are different, and this will influence the decision.

If we start to consider these other factors and add a second axis, we force a wedge through the grey area.

If the overall reach of an application is important, as it often is with consumer applications, this tips the balance towards HTML5, which has (or at least will have in the near future) the greatest reach.

In the scenario illustrated by the diagram above, an application of moderate, or 'interactive' complexity is being developed. If reach is not an issue (for example the application might be deployed on a company intranet) this pushes us towards Flex / Silverlight where the streamlined development experience will reduce costs. If high reach is desired, this pushes us more towards HTML5.
The above exercise can be repeated with other key factors relating to the development of a specific application. For example, if development cost is important, this tips the balance towards Flex / Silverlight.



/ FLEX VS. SILVERLIGHT

In the previous section Flex and Silverlight have been considered together due to their similarities with each other, and great differences compared to HTML5. If, for a particular development, the 'comfort zone' is within the Flex / Silverlight region, the next decision is which of the two plugin technologies to use.

Something to bear in mind is that from an end-user perspective, the two technologies are very similar, sharing the same sort of capabilities and experiences. Important factors that influence the choice between the two relate to productivity and the technologies which the team or business currently use.

- **Silverlight is probably an obvious choice for companies which are already heavily invested in Microsoft technologies, where the familiarity with the tools and languages make moving to Silverlight an easy transition.**

- **The synergy between Flex and Java, due to the similar IDEs and Java-Flex connectivity frameworks, means that Flex is a good choice for companies that already have Java software.**

However, there are other non-productivity related factors to consider:

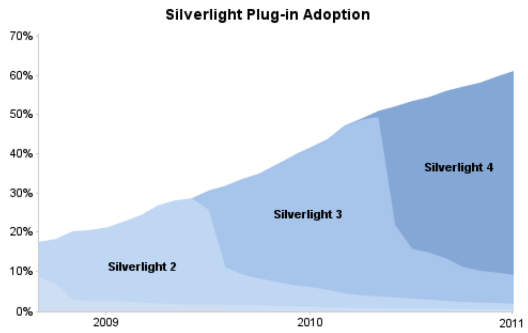
- **The adoption of Flash / Flex is currently very high, whereas Silverlight is much lower. For applications for the general consumer, the current level of Silverlight adoption would probably be considered quite a significant issue.**

- **Flex, because it is built on top of the ubiquitous Flash plugin, is at the moment more future- proof than Silverlight, which has yet to become really 'embedded' in the web.**

/ CRYSTAL BALL GAZING

If complexThe internet and its associated technologies are constantly changing, with browser usage and technology adoption statistics looking very different from one year to the next. From the previous section the 'comfort zone' of each technology was defined based primarily on application / website complexity.

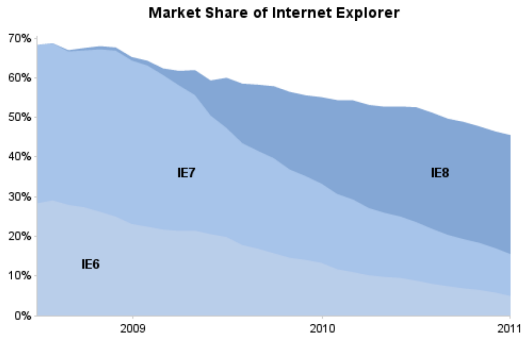
Silverlight is a relatively new technology, and is currently being hampered by the current level of adoption, although the current statistics indicate that this is increasing at a steady rate with a predicted adoption figure of 76% by the end of 2011 [13]. Silverlight has the slight edge on Flex in terms of capabilities, and this, coupled with the popularity of the .NET framework, will likely lead to Silverlight gradually acquiring some of Flex's market share in the application space. Flex is not 'sitting on its laurels', with Adobe recently announcing that AIR, their cross-platform runtime for Flex, now supports mobile devices [14].



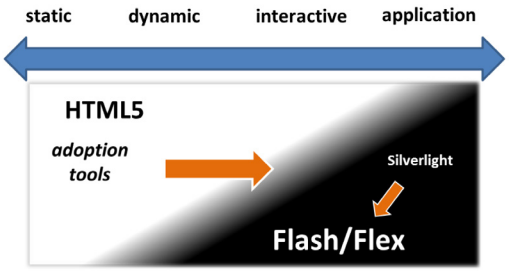
Just as Flex has been built upon the rendering and media stack that Flash provides, HTML5 provides a rendering and media stack that can be built upon by frameworks and abstraction layers. As the HTML5 abstraction layers become more advanced, this will make it easier to deliver more complex applications using this technology.

Recognising this potential, Adobe has released 'Wallaby' an experimental technology that converts Flash animations to HTML5 [15].

One of the obstacles currently faced by developers of HTML5 applications is the lack of feature support in older browsers (most notably Internet Explorer). This means that the 'reach' for some of the HTML5 features are quite low, for example WebWorkers currently have a reach of 50%. The poor HTML5 support of Internet Explorer, coupled with its large market share is certainly holding this technology back. Internet Explorer version 9 has good HTML5 support [10] and with historic trends indicating that users do upgrade to the latest version of Internet Explorer, it is likely that the reach of HTML5 will be much higher in a few years' time.



If we revisit our earlier diagram which illustrates the comfort zones of each technology, HTML5's rising adoption, improved tooling and abstraction will allow this technology to push more towards the right-hand edge of the spectrum. The rise in adoption and a solid toolset gives Silverlight the potential to eat away at the Flash / Flex market-share. In summary, Flex is going to have a fight on its hands from Silverlight, its direct competitor, and both are being threatened by HTML5.



It is very difficult to predict where we will be in a few years' time and whether there will be some technology casualties along the way. Can all three technologies co-exist? Or will one or more be relegated to the internet history books like Java Applets and ActiveX?

CONCLUSIONS

THIS PAPER HAS LOOKED AT THE FEATURES OF FLEX, SILVERLIGHT AND HTML5 IN DETAIL. APPLICATION COMPLEXITY HAS BEEN IDENTIFIED AS A KEY METRIC FOR DIFFERENTIATING THESE TECHNOLOGIES, AND THE COMFORT ZONE OF EACH HAS BEEN IDENTIFIED. THIS CAN BE USED AS A GUIDE TO SELECTING AN APPROPRIATE TECHNOLOGY FOR WEB APPLICATION DEVELOPMENT.

/ ACKNOWLEDGEMENTS

Thanks to my colleagues Graham Odds, Chris Price and David Pentney for their ideas, feedback and assistance in writing this paper.

BIBLIOGRAPHY

1. Adobe LiveCycle Data Services ES2. [Internet]. Available from:
http://www.adobe.com/products/livecycle/dataservices/.

2. mint.com. [Internet]. Available from:
http://www.mint.com/.

3. Morgan Stanley Matrix. [Internet]. Available from:
http://www.morganstanley.com/matrixinfo/.

4. StatOwl.com - Statistical analysis and market research of Internet usage trends. [Internet]. Available from:
http://www.statowl.com/.

5. In 2009 The Demand for Flex Developers Heats up!. [Internet]. Available from:
http://www.flexden.net/blog/2009-demand-flex-developers-heats.

6. How to find Flex developers! [Internet]. Available from:
http://ted.onflash.org/2007/07/how-to-find-flex-developers.php.

7. World Wide Web Consortium (W3C). [Internet]. Available from:
http://www.w3.org/.

8. Web Hypertext Application Technology Working Group. [Internet]. Available from:
http://www.whatwg.org/.

9. Berners-Lee T. Reinventing HTML. [Internet]. Available from:
http://dig.csail.mit.edu/breadcrumbs/node/166.

10. Microsoft previews Internet Explorer 9 with HTML 5 support. [Internet]. Available from:
http://www.zdnet.co.uk/news/desktop-apps/2010/03/17/microsoft-previews-internet-explorer-9-with-html-5-support-40088334/.

11. Pilgrim M. HTML5: Up and Running. 2010.

12. Mozilla drags IE into the future with Canvas element plugin. [Internet]. Available from:
http://arstechnica.com/software/news/2008/08/mozilla-drags-ie-into-the-future-with-canvas-element-plugin.ars.

13. Silverlight 5 Adoption Predictions. [Internet]. Available from:
http://www.scottlogic.co.uk/blog/colin/2010/12/silverlight-5-adoption-predictions/.

14. What's new in Adobe AIR 2.5. [Internet]. Available from:
http://www.adobe.com/devnet/air/articles/air25_whatsnew.html/.

15. http://labs.adobe.com/technologies/wallaby/. [Internet]. Available from:
http://labs.adobe.com/technologies/wallaby/.

Scott Logic is a leading software consultancy with enterprise level clients in finance, energy, healthcare and the public sector. Our services include software development & delivery, user experience design and an independent consultancy advising on strategic software selection and deployment. Our consultants, recruited on their exceptional qualifications, skill sets and knowledge, are central to continued project success and complete client satisfaction.

enquiries@scottlogic.com

SCOTT LOGIC / ALTOGETHER SMARTER

3rd Floor, 1 St James' Gate
Newcastle upon Tyne
NE1 4AD

+44 333 101 0020

48 Warwick Street
London
W1B 5AW

+44 333 101 0020

25 King Street
Bristol
BS1 4PB

+44 333 101 0020

1st Floor, 80 George Street
Edinburgh
EH2 3BU

+44 131 202 8625

www.scottlogic.com