Why HTML5 Tests the Limits of Automated Testing Solutions

SMARTBEAR

SMARTBEAR

Why HTML5 Tests the Limits of Automated Testing Solutions

Contents

SMARTBEAR

As Testing Complexity Increases, So Must Scalability

Websites that act like desktops, with drag and drop functions, on-page calculators and other interactive features, pose special testing challenges. Testers must deal with multiple technologies, inconsistent browser behaviors and a highly dynamic development environment.

HTML5 is intended to simplify things by incorporating within HTML functions that previously required external plugins. The result, however, is an environment that is actually less tester-friendly.

HTML5 introduces additional issues testers must navigate. The standard is still evolving. Browser support is inconsistent. And new elements require new kinds of tests.

Is HTML5 “the pin that breaks the back” of a testing solution? That depends on whether the testing model on which it is built is “complete.” In other words, is it inherently extensible, and does it provide a 360˚ view of the application from a testing, programmatic and user perspective? Finally, do those views stay in sync as things change in any of those three domains?

> *When selecting a web application test solution you can’t know every mix of technologies you may encounter. But pick the right testing model and you won’t need to know.*

You need to revisit your testing strategy.

In fact, HTML5 is the poster child for the principle that your choice of testing strategy will confer either big dividends or big penalties as new web technologies emerge. If you want to use an HTML5 extension (they keep coming), then you must learn that extension's coding rules — and the potential ripple effects that using the extension will have on other technologies present in an application. It's not just about learning (and testing) HTML5 (whatever the new technology happens to be), it's also about containing its multiplier effects on all the other evolving technologies in play.

And you need help in the form of test automation.

That's hard to do manually. It would require visually inspecting lines of code or writing a different script to test each function. Even if you knew all the relevant rules for all technologies and extensions, and how they relate to each other in every browser, you still must apply that knowledge through potentially thousands of source lines. Can you really catch each and every issue — and how long will that take? Clearly, if you want to scale testing you have to automate it.

But how can test automation help you scale your testing automation process?

Automation must improve how you test.

One way is to work with objects (abstractions of text) rather than just text. Since each interaction onscreen (e.g., filling in an order form) is with an object (e.g., a total dollar amount), that makes it possible to automate the creation of test scripts just by recording user interactions with objects. And since object classes are logical rather than physical, a tester's software need not rely on code's physical properties (like line numbers) to compare how objects are written across code iterations. And since any new class is simply a new set of rules, the complexity of handling new HTML5 objects is often reduced to adding a few new items to a dropdown.

What Makes HTML5 Testing Hard

The web attracts a broad range of developers in terms of skills. The key is finding a testing solution that serves all testers well.

Despite the fact that HTML5 extensions (see sidebar on next page) were designed to simplify RIA development, the result has been the opposite. Not only is the standard itself a moving target, and therefore inconsistent over time, the same HTML5-compliant application may need to be coded differently to behave the same on different browsers.

Chrome, for example, implements HTML5's drag and drop attribute fully, while Internet Explorer and Firefox do not. Firefox also does not support H.264 video, while Internet Explorer and Chrome do. Of the three, only Chrome supports the number input type — and so on.

The fact that a test fails, therefore, may indicate any of three possibilities:

1) The code is not HTML5-compliant.
2) It's not compliant with a particular browser.
3) It's not compliant with either HTML5 or the browser.

Examples of browser incompatibility

Furthermore, as the standard and the browsers continue to evolve, the process by which one decides between those three possibilities also evolves.

The arrival of HTML5 also doesn't necessarily make other RIA technologies, like Flash, Flex and Silverlight, disappear. With regard to HTML5 versus Flash, perhaps Fred Cavazza summed up the the situation in Forbes:

> *"Of course, there are many technical arguments for one technology over the other. But the best and most important part is that you don't have to choose between HTML5 and Flash because you can use both … depending on the experience you wish to provide, your ROI and SEO constraints, and the human resources you access."*

When developers decide to add HTML5 to their existing RIA toolkit, it means layering on an additional set of rules for testers to validate against — both for how to code in HTML5 and for how to code in the other technologies within an HTML5-compliant application.

> *"Testing a web page means testing the layout, the logic inside and the content being presented… multiple layers of information makes testing more difficult."*

HTML EXTENSIONS

HTML5 is intended to make the experience of using an open solution equal to that of using an RIA coded in Flash or Silverlight across all platforms, including mobile, without having to rely on proprietary plugins like Flash and Silverlight. The goal is not only to make development easier — by allowing developers to do more with fewer and more open tools — but also to help ensure applications behave more consistently across browsers and platforms so that a web page looks and feels exactly as its author intended.

Key HTML5 Extensions

- New tags for audio, video, and canvas (i.e., animation)
- New tag attributes for meaning and context (e.g., width="320" height="240")
- New tags (e.g., dragable, spellcheck, hidden screen elements)
- New input types (email, number, date / time)

So now, for example, users can drag items around on a webpage the same way as they would a file from folder to folder on a desktop. Developers can also turn on spellchecking features and show or hide screen elements via attributes. New input fields let developers automatically check whether the data users enter (an email address, for example) is valid.

Then there are all the factors that make RIAs inherently more complex to test than, say, static web pages, or even a desktop application with similar functionality.

First, functionality is distributed physically between software running in a browser (a complication all by itself), and software running in a web server. Second, functionality is implemented in different technologies.

In HTML5's case, HTML handles rich content. CSS3 handles style (e.g., custom fonts, rounded corners). And JavaScript handles logic (e.g., web workers doing computational tasks "behind the scenes").

One consequence of this complexity is that it can be difficult to determine how something in the code affects something onscreen and vice versa. Even when the connection is obvious, the relevant line numbers and screen coordinates may change when developers edit the code. That can make testing difficult by conventional methods (i.e., manual code inspection or handwritten scripts), because testers have to keep "chasing the change."

All these layers — in web standards, browsers, application architecture, and code itera-tions — have a compounding effect that makes it both harder and also more critical to validate all the values that might indicate whether an application is working properly.

HTML5 also enables developers to employ two other updated standards, CSS3 and JavaScript.

CSS3 includes support for:

- Text shadow effects, which no longer require a graphics editing program.
- Embedded fonts, so you can now display fonts not installed on a user's computer.
- Animations, which previously would have required a third-party plugin.

JavaScript works behind the scenes to make websites more interactive. HTML5 supports JavaScript features that include:

- **Offline storage**, lets users cache entire apps or websites on their computer so they can keep browsing the site even if the Internet connection is lost.
- **Web workers**, a JavaScript routine that performs intensive operations in the background without affecting a user's browsing experience.
- **Server sent events** that allow server-based applications to update values on a web page (like values in a spreadsheet) without the need to refresh the whole page.

An example might be the contents of a box that holds the result of a calculation (i.e., a property value). So the lists of things you can check for consistently across standards, browsers, architecture and iterations should be as complete as possible — including everything from property values to files contents to the response of a web service and more.

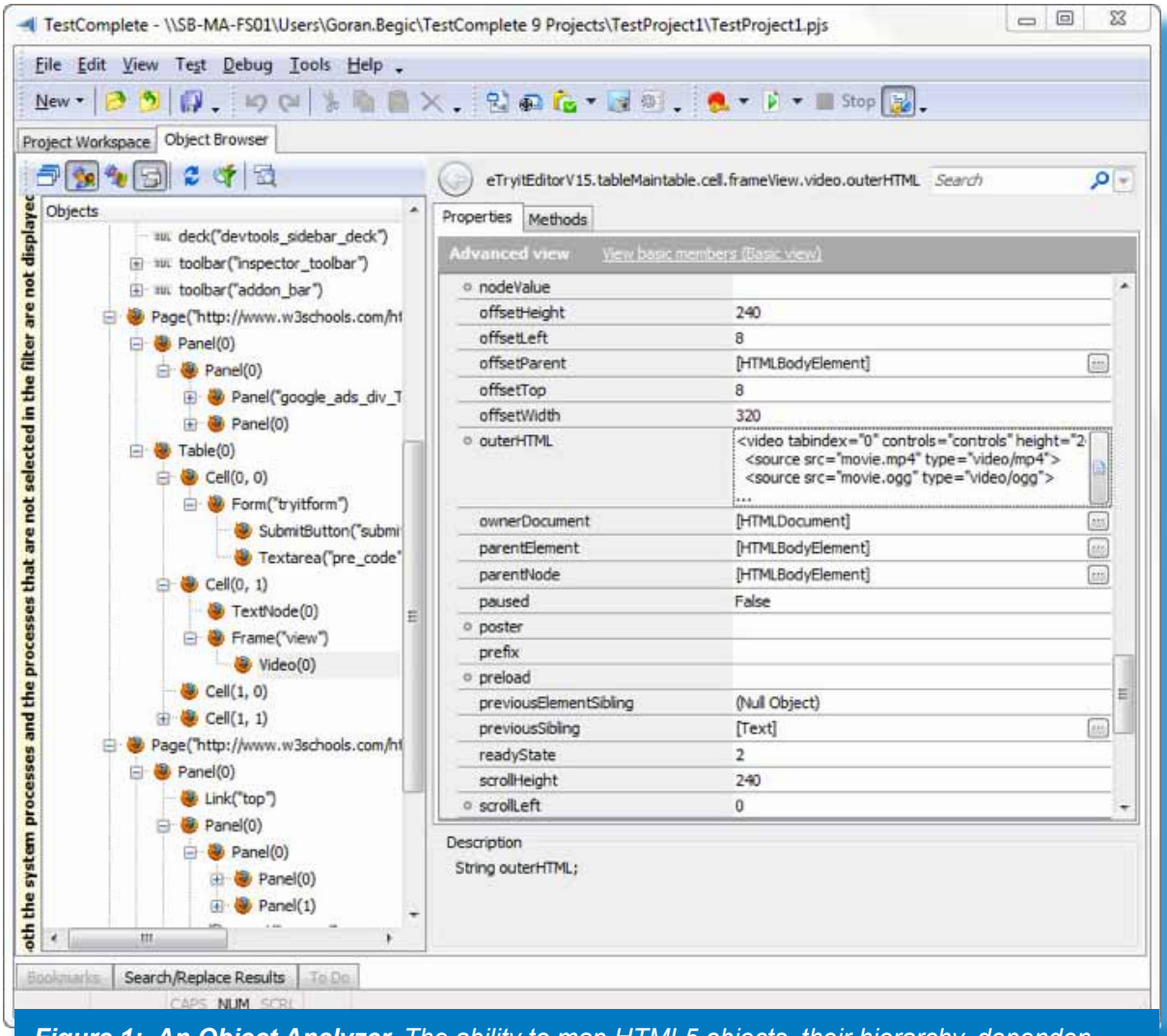


Figure 1: An Object Analyzer. *The ability to map HTML5 objects, their hierarchy, dependencies, properties and methods provides a technology-independent view into how an application works (or doesn't).*

Are you ready for success?

Also not to be overlooked is how the web attracts a broad range of developers in terms of skills. Just as some singers can read a piece of sheet music and hear the melody in their head, some developers can look at a piece of code and visualize what happens onscreen in the web browser. Others can't. So that's one final complication — finding a testing solution that serves all testers well, even those who need help "seeing" how changes in the code will actually play out when users run the application.

Checkpoints for a Scalable Testing Solution

The completeness of an HTML5 testing solution clearly correlates to the organization's ability to scale testing despite all the complexity involved.

A solution that is more complete has answers to more questions like:

? *Can you verify all three components of your HTML5 solution, including rich content, style and logic?*

Defects can originate in any of these three aspects of rich Internet applications and missing anyone of them can impact quality of the end product.

? *Can you automatically build test scripts just by recording interactions with the application like a typical user?*

Test recording capability dramatically reduces test development time allowing you to easily capture the use case even if you decide to improve the recording by editing the test later.

? *Can you add custom logic to the scripts, such as if-then-else statements that test whether an application's own condition logic enables it to run properly in different browsers?*

While the recorded tests are generally ready for playback, you may want to increase robustness of the tests, or add variables to iterate over several test data sets.

? *Can you view the script as a list of "keywords," i.e., application objects that control what's happening at that point?*

In order to improve the review process and enable stakeholders without programming skills to participate in test automation, you may want to consider solutions that provide natural language-like interface for test editing and review.

? *Can you view the script as a sequence of screen shots, each of which shows what the user sees in the browser at that point in the running application?*

The ability to track what the user sees onscreen to the corresponding step in program execution helps testers locate software defects faster.

? *Can you see which screen shot relates to which object just by clicking on a screen shot or an object?*

An important aspect of test automation is the ability to correlate test steps with the information presented on the user interface. During the review of test results you can see the sequence of interactions with the application and get a better understanding of steps that didn't unfold as you may have expected.

? *Can you insert checkpoints in the scripts that test for certain conditions, such as whether a control has the right property value or input type at that point in the running application?*

The investment in test automation returns through repeated and automated test execution. The more you run your automated tests, the more you save. Without checkpoints you would have to manually review all the test results and test data and that would reduce the value of test automation altogether.

? *Is the list of checkpoint types (e.g., property values, file contents, web service responses) comprehensive?*

One could argue that a little bit of programming skills could lead to definition of many different custom checkpoints, but custom scripts would increase the cost of test maintenance. Furthermore, with the increased adoption of test automation it may get more difficult to find testing resources with programming skills. Built-in checkpoints provide the best of both worlds — consistency of the solution for easier adoption as well as the flexibility and power required for robust evaluation of results.

? *Can you automatically generate a complete object map showing all object internals and dependencies, as in Figure 1?*

To optimize recorded tests and to increase test robustness you may want to enhance your test scripts with information available in objects that are not represented in the user interface. To do so, you need to be able to conveniently access all application objects and their methods and properties.

? *Can you isolate and test internal methods (e.g., JavaScript web workers) whose operations aren't visible to users?*

Some defects create obvious errors, often in the form of handled and unhandled exceptions. Others are latent and waiting for the right use case to expose them, or they may stay undetected. To expand checkpoints beyond what is visible you may need access to object internals.

Answer "Yes" to these questions if you want scalable testing.

Why it makes sense to automate by recording user interaction with the software under test.

Recorded user actions make tests more “lifelike,” require far less labor than manual code inspection and eliminate the need to implement a shadow testing script development effort in parallel to application development.

Testing is also easier for lower-skilled testers, especially with screen shots to guide them.

Higher-skilled developers, meanwhile, get to tailor testing code at will — with help from automated object maps and object analyzers — to meet any complexity challenge today’s RIAs present.

For example, running the same test successfully under different browsers, operating systems and even screen resolutions is more likely, given that every one of the application’s onscreen items is linked to its controlling object by name rather than by lines of code or screen position.

This is the strength of the object oriented testing model. It’s independent of all the layers that would otherwise complicate RIA testing and thus defeat scalability: changes to standards, browsers and even the application code.

It makes the model infinitely extensible as new layers are added and keeps all three perspectives on the application in sync — whether you work with code as a developer, with scripts as a tester, or with onscreen behavior as a user.

That’s the test of an HTML5 testing solution.

Testing HTML5 - Additional Resources

SmartBear Web pages and Blogs

Blog: HTML5 Test Automation for Beginners

- Getting started with test automation and HTML5? This article introduces both topics in clear and simple terms.

Blog: How To Automate HTML5 Testing - Webinar Q&A with Nick Olivo

- Learn how to test offline browsing in HTML5 and other tips and tricks.

Blog: How to Automate HTML5 Testing

- See what other testers want to know about HTML5 test automation in this overview of questions for the webinar discussion.

Videos

How to Automate HTML5 Testing

- Overview of HTML5 standard and examples of test automation for the new HTML5 tags like <video> and <canvas>, CSS3 style sheets and JavaScript logic in HTML5 web applications.

Webinars

A SmartBear/SQAForums Webinar: How to Automate HTML5 Testing

- In this webinar, Nick Olivo, SmartBear's test automation expert, demonstrates critical tips and tricks to effectively automate HTML5 testing: what testers need to be aware of when automating an HTML5 site; how to create automated tests against HTML5 sites with help of TestComplete; and how to automate cross-browser testing of HTML5 sites.

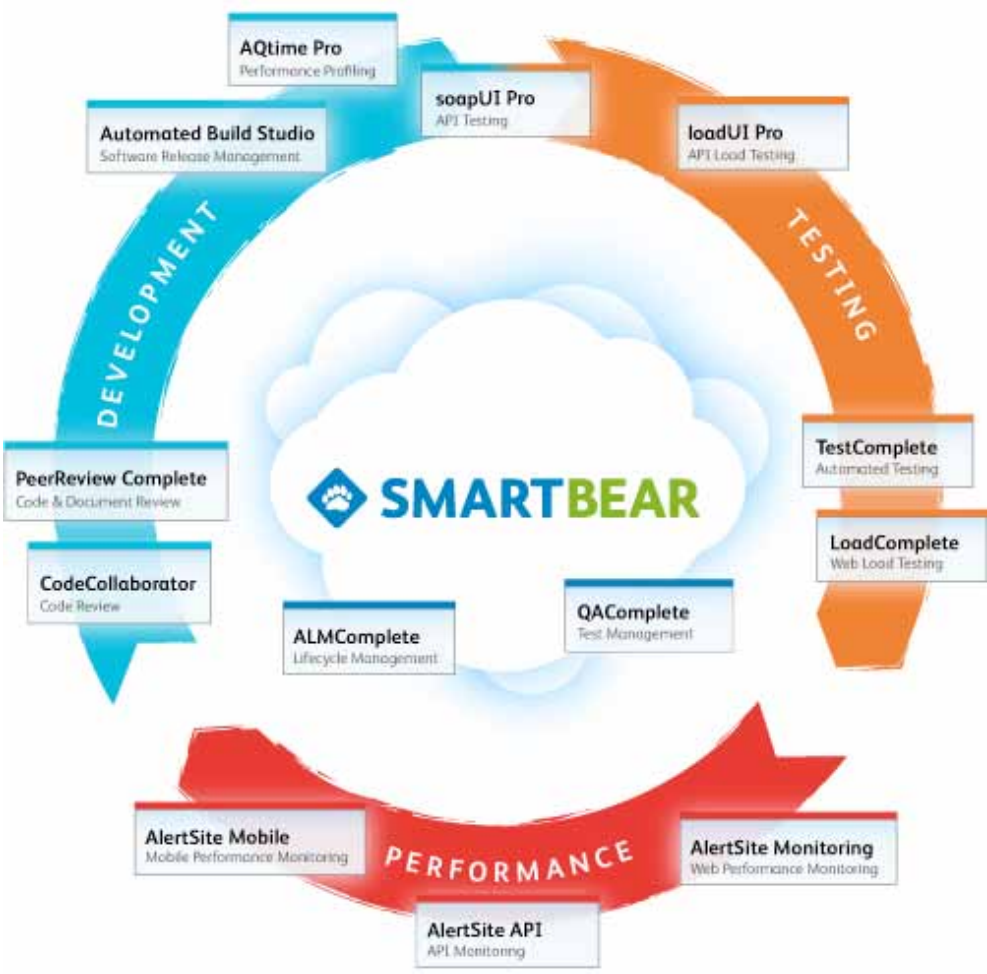
Industry Influencers and External Resources

- HTML5 Browser Compatibility Table
- HTML5 Browser Compatibility Test
- The Challenge with HTML5 - In Pictures
- Beyond the Canvas: Testing HTML5
- iPhone 4s and HTML 5 - the future of enterprise apps

SMARTBEAR

Quality Lifecycle Management

At SmartBear Software, our mission is to **WOW software developers**, quality assurance engineers and operations professionals with affordable tools that save time and improve quality. We want to make it easy for our users to improve the software quality of their desktop, mobile and cloud dependent applications.



SmartBear helps:

- Developers build better code and fix issues earlier
- QA test faster and increase coverage
- Web Operations to ensure optimal application performance

Join the *Software Quality Matters* Blog

Try any of our products free for 30 days.

Visit www.smartbear.com today or give us a call at 978-236-7900.