FOUR UI TEST AUTOMATION **TIPS**

for HTML5 Applications

Telerik
TEST STUDIO

Four UI Test Automation Tips for HTML5 Applications

Every application needs a solid mix of automated testing: unit, integration, and functional/User Interface. UI automation has always been at the top of the test pyramid, but not because they're the best type of automated tests. Rather, UI tests should be a few carefully chosen tests built on top of a wide base of unit tests and a solid middle layer of integration tests.

UI automation has always been challenging for many reasons, and many teams shy away from creating UI tests at all. This is unfortunate, because carefully crafted UI automated tests bring a great amount of value to any project.

HTML5's ability to mix in a wide range of new functionality at the UI level brings a number of challenges when choosing what tests to automate, and how to go about automating them. There are also a wide range of HTML5 features that should be left to manual testers' validations during exploratory testing sessions.

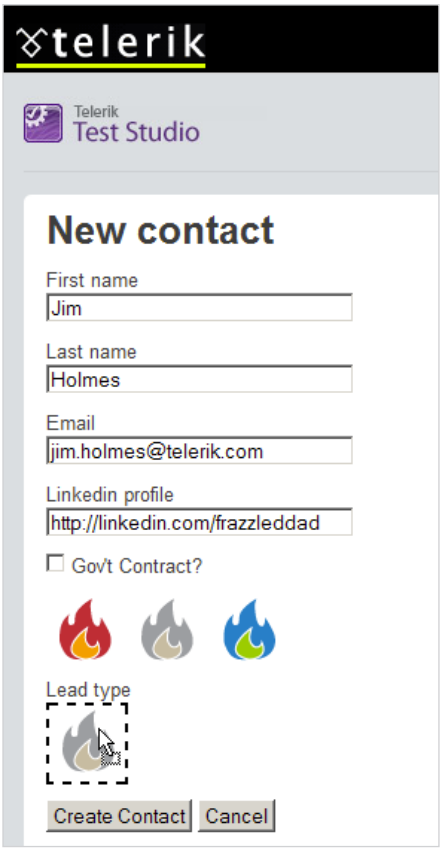
Focus on Value

As with any testing, UI automation, regardless of whether it's on an HTML5 application or not, has to be brutally focused on providing value over the long run. "Long run" means looking to minimizing maintenance costs around updating tests as the system evolves. This means teams writing the automation should focus on validating business-level features. Automation shouldn't (generally) be written to cover look and feel of pages or views—this is high effort automation that's extremely brittle and will suck up significant amounts of time to modify when small changes are made to the UI.

That said, we do need to focus on HTML5 features that might be part of a high-value functional slice. Let's discuss a few examples of these features that might be good targets for automating.

Drag and Drop

Normally I'm not a fan of automating UI "bling" like drag and drop; however, in some cases D&D is part of a critical path for a functional slice. An example is in the Test Studio demo app where creating a new contact requires a D&D operation on the lead type for the contact.



Four UI Test Automation Tips for HTML5 Applications

You can't create a new lead without dragging a contact, so automating that D&D is crucial to the test. Sensible automation would focus on automating the basic drag and drop operation as part of an overall "create a new contact" test. It wouldn't be sensible, however, to try and validate the appearance of the ghost image when elements are dragged. (Dragging Google's image on their search page is a great example of what the ghost image is.)

Trying to automate checks for that ghost image doesn't make sense. First, it's not mission critical to the functional slice. Second, you're likely trying to validate functionality you didn't actually write (it's part of the browser's handling of the HTML5 spec!). Finally, a tester can quickly confirm the ghost image as part of a general exploratory session.

Input Field Validation

Various input fields in HTML5 provide rudimentary validation as part of each browser's implementation of HTML5. These validations are a great example of things to carefully think about before automating. Again, you need to be careful about not automating functionality that's delivered from some third party — in this case Firefox, Chrome, Safari, Internet Explorer, etc. Is it mission critical for your test case to know if those browsers have changed or broken their basic handling of the HTML5 validation? Is it a high-value test case in the first place? If so, then adding some testing around the validation would make perfect sense.

Cross-browser differences can greatly impact this particular feature of HTML5, as can versions of each browser. The Art of Web has a great blogpost summarizing some of these differences. Think carefully about how you want to tackle automating this particular feature, and what sorts of browser differences might matter to you.

HTML5's Media Elements

One of the neatest things about HTML5 and its features is the media element. It's also one of the most confounding for teams working on automation. The complexity comes in with the many different media players that can be included with an application. The basic functionality of play, pause, stop is fairly simple to automate across all players; however, anything more in-depth requires dropping to JavaScript and attempting to interact with the player directly. This can add a lot of complexity to your test suites, so it's critical to understand whether such additional testing makes sense for the team and the project.

If your team does decide to go this route, then it's important to abstract all your automation suite's JavaScript calls behind some form of test infrastructure library. This allows you to update your calls as needed without breaking all your tests.

Testing Local Storage

Local storage for HTML5 applications is a powerful concept. You've finally got secure, stable access to handle offline storage and increase your application's performance. This is a boon to developers who've struggled with these issues in the past!

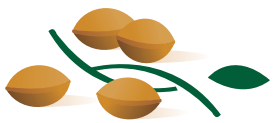
Testing local storage features for your application will again point you to creating APIs in a testing infrastructure library to handle validation and configuration of items on the local client system. You'll need to carefully think through the environment issues so that you can correctly point your tests to appropriate areas for validating file creation and update on the local system. You'll have to evolve your infrastructure APIs as you build up your tests, and you'll need to manage those environments across all the different browsers and platforms you work on.

Always Focus on Value!

Testing HTML5 applications isn't so tremendously different from writing automation for earlier versions of HTML. Yes, there are a number of new technology features to take into account, but the general principles of automation still apply:

- Automate only high-value functional slices
- Avoid automating functions that don't change often
- Avoid automating low-value features
- Avoid automating functionality from third party vendors
- Use test infrastructure APIs to hide complexity and minimize impacts of change

UI automation isn't ever easy, but it can bring great value to your teams' projects.



In search of an HTML5 app testing tool?
Check out Telerik's Test Studio.

Get yourself started with the Test Studio 30-day trial.

Jim Holmes

Jim Holmes is the Director of Engineering for Test Studio. He has around 25 years IT experience. He's a blogger and the co-author of "Windows Developer Power Tools" and Chief Cat Herder of the CodeMash Conference. Find him as @aJimHolmes on Twitter.