

ExtJS: An Introduction

Don't Be Alert: The world doesn't need more lerts

The Red Pill: Functional Programming in JavaScript

What's New in jQuery 1.3

Community News

Welcome to the first JSMag!



JSMag aims to bring the best JavaScript-oriented articles to you every month, with a focus on some content you might not see many other places. While some of the articles we will run may be a bit “beginner” for some readers, we’ll be trying to bring at least a few articles that will help even seasoned developers learn a new trick or two.

But wait — aren’t monthly magazines on the way out? Aren’t blogs making monthly content outlets obsolete? Possibly, but personally, I don’t see it that way. Let me share some of the principles on which this magazine is founded, and where we see JSMag providing value.

Between a blog and a book

While there are numerous blogs available, most are journals, outlining a fellow developer’s experiments with new technology. Blogs can be great for keeping up with the cutting edge, but usually don’t go deep enough in to a subject to help a large number of people. On the other end of things, technical books provide large amounts of great info, but are often outdated within weeks of publication. By providing a monthly stream of information, we’ll be bringing you content that’s more polished than most blogs, and more timely than most books.

The how and the why

I ask all contributors to use their own voice and experience as much as possible. It’s very easy to find JavaScript code snippets without good context. It’s also very easy to find forums threads

filled with academic debate devoid of code. My charge to our writers is to give both the technical “how” part coupled with the “why” from their own experiences.

Subscription-driven

JSMag (and its sister magazine, GroovyMag) are first and foremost put together for developers. Our aim is to bring you quality content event month. Period. We’re not in the business of delivering eyeballs. An advertising-driven approach can inevitably bring conflicts of interest, at the expense of the integrity of the magazine. We will occasionally have advertisements that we deem will be of sufficient value to the readership, but JSMag is not advertising driven. Our primary (and majority) revenue comes from the sale of the magazine.

In this issue, we have Rebecca Murphey detailing some of the new features in the recent jQuery 1.3 release. Matt Henry walks us through Unit Testing JavaScript with the YUI framework. Robert Fischer may cause you to rethink some of your JavaScript development with his piece on Functional Programming, and Robert Cameron outlines some of the ways that you can test your code without resorting to the all-too-common “alert()”. Lastly, we’ve got Brett Harris and Jed Brubaker giving us all a hands-on demonstration of the ExtJS toolkit.

I hope you enjoy our first issue and look forward to your comments and feedback. Please write me at editor@jsmag.com.

Thank you!

Contents

The Red Pill: Functional Programming in JavaScript	3
ExtJS: An Introduction	9
Don’t Be Alert: The World Doesn’t Need More Lerts	14
JavaScript Community News	19
Unit Testing JavaScript	20
jQuery 1.3	28

Editor

Michael Kimsal
editor@jsmag.com

Technical Editors

Darryl Bloch
Craig Wickesser

Graphic Design

Rebecca Murphey

Contributors In This Issue

Robert Fischer
Jed Brubaker
Brett Harris
Rob Cameron
Matt Henry
Rebecca Murphey

jsmag.com twitter.com/jsmag

The Red Pill

Functional Programming in JavaScript



By Robert Fischer

Functions are the basic building block of JavaScript's structure. This article will provide a brief introduction to some key ideas of functional programming in JavaScript, as well as demonstrate some of the lesser-known features of JavaScript functions. After reading this article, the reader will be able to manipulate functions in a much more direct way, including writing their own higher-order functions.

The average JavaScript programmer routinely overlooks one of the most powerful features of the language: function manipulation. Functions are used throughout JavaScript, including forming the basis of the object model. The libraries/frameworks provided to JavaScript regularly utilize anonymous functions in the form of callbacks: they look like `function() { alert("Done!"); }`. These hooks have much more power than most people realize, because each method, function definition, and callback opportunity can change the execution of the application at the most basic level.

Before we set out, a warning is probably in order. Like most things in JavaScript, the user is given all the power they want (and possibly more). As will be demonstrated, there are very few safety checks in JavaScript function calls, which can be both a blessing and a curse. This makes the re-writing and function mangling possibilities in JavaScript effectively endless, but this looseness can also make bugs tricky to track down.

Basics of Functions

Even an elementary JavaScript developer is familiar with defining and executing the basic function call, as demonstrated in Listing 1.

The name of a function, however, is purely optional: it is valid JavaScript to leave out the `funk` from the example above. Such a function is called an anonymous function, and is usually assigned to a variable (as in Listing 2) or passed directly into another function (Listing 3). When a variable references an anonymous

function, the variable can be called as if it were a function – so storing a function into the `it` variable means that `it()` calls the stored function.

```
function funk(who, what) {
  print(who + "'s " + what + " has got the funk!");
}
funk('Robert', 'code');
funk("Joe", 'towel');
// This prints out:
// Robert's code has got the funk!
// Joe's towel has got the funk!
```

Listing 1: The Basic Function Call

```
var myFunk = function(who, what) {
  print(who + "'s " + what + " has the funk!");
};
myFunk('Robert', 'code');
// This prints out:
// Robert's code has the funk!
```

Listing 2: Function in a Box

```
function doIt(it) {
  it();
}
doIt(function() {
  print("Sneaky!");
});
```

Listing 3: Function Without a Box

The body of a function (anonymous or otherwise) can refer to variables defined outside of that body, as demonstrated in Listing 4. This capturing of the surrounding variables is called enclosing its scope, and so the function is said to be a closure. Since normal JavaScript functions can refer to their outside scope, they are technically closures, but the word closure is often wrongly used to mean the same thing as anonymous function.

ExtJS: An Introduction



By Brett Harris and Jed Brubaker

There is a world of AJAX libraries available. They are powerful. They let us make great websites. They provide slick and sexy interfaces. However, some of us don't want that. After all, slick and sexy is what resulted in the <blink> tag. Some of us want to make desktop-like applications on the web. Our numbers are growing quickly — just look at Google and Microsoft. There was a time when Google Maps blew our minds, and it won't be long before you open a browser to use Microsoft Office. These rich interactions, however, are complex and challenging to implement. There is a solution. It is called ExtJS.

ExtJS is developed and maintained by Ext, a company that also develops Ext GWT, a Java library for building Rich Internet Applications with GWT. ExtJS 2.0 was released in December 2007 and has undergone 2 revisions since. Since its inception, core development on ExtJS has been led by Jack Slocum. He and his team members actively participate on the forums, so help is never too far away. While ExtJS is licensed under GPL v3, Ext also provides enterprise licensing, support, and training for the framework.

The ExtJS JavaScript framework allows developers to easily implement desktop functionality on the web. It provides a base library for all those little things: finding and manipulating DOM elements, making AJAX requests, managing events and listeners, and a whole mess of other trivial and mundane JavaScript tasks. However, where ExtJS really shines is with its portfolio of feature-rich widgets. There is a whole set of components that have lived on the desktop since the early days of the GUI, but somehow seem strange and new to web developers. How did we live without tree interfaces, drag and drop, and the oh so important interactive data grid? It is time to take back the browser.

ExtJS is a complete package for Rich Internet Application (RIA) development. ExtJS was originally built on Yahoo's YUI! framework, but has since abstracted the implementation of its base library. Developers who work with YUI!, Prototype, or JQuery can continue using their favorite JavaScript framework while leveraging the extensive ExtJS widget set. The portfolio of widgets jump starts your RIA development by providing all the desktop

functionality you will need in a componentized, extensible model. You will spend considerably less time building the features that today's users expect, allowing you to spend more time on development and widget tweaking to meet your specific business needs. Since today's hottest interactions are provided to you out of the box, you can also spend time designing new interactions to improve your users' experiences.

The good ...

ExtJS is truly an application-centric framework and comes with a plethora of widgets most commonly found in desktop applications. There are toolbars, windows, tabs, and buttons, as well as sophisticated forms and data grids. You could replicate an entire OS inside your world of <div> tags. Need a resizable grid with sortable data columns? How about client-side form field validation with pretty tool tips? ExtJS can enhance the experience of your users without batting an eye.

ExtJS doesn't just stop at the implementation. Ext, the company, has backed up ExtJS with excellent documentation and support. They provide samples for each of the widgets to get you started. We often find ourselves copying the sample code as a starting point and tweaking bits and bytes to suit our needs. There are multiple samples for most widgets, allowing you to see different features in isolation and to see different approaches to similar problems. But, the samples are just the jump off point. You will quickly fall in love with some of the best API documentation that we have ever seen. It is thorough (with a few caveats — read on, grasshopper) and well organized. The documentation gives clear descriptions of configuration options, public methods, and events for every widget and class in the framework. Each page in the documentation lets you inspect the code behind the scenes, so you can always dig further for a better understanding of the nuts and bolts. Oh, and the best part is that the entire API is written using ExtJS. Gotta love a company that drinks its own Kool-Aid. But the treats don't stop there. The trifecta of documentation is completed by the forums. Sure, there is lots of help for all frameworks, languages, and DIY hardware hacking, but we have encountered few product forums where lead developers

Unit Testing JavaScript



By Matt Henry

Unit testing has become an indispensable part of many programmers' workflow. Recently, several tools have been created that allow JavaScript developers to write unit tests for their code. In this article, we will take a look at what tools are available, and then go through some detailed examples of how to write unit tests using the Yahoo! User Interface (YUI) Test library.

Introduction

Since JavaScript has started gaining acceptance as a “real” programming language, its users have begun to borrow concepts and techniques from other parts of the software world. One of the most beneficial practices to take root in the JavaScript community is unit testing.

What is unit testing?

Unit testing is the practice of isolating the smallest bits of functionality (viz. the “units”) in a piece of software, and programmatically testing their behavior in well-defined input scenarios. Depending on the language, for instance, the smallest units may be methods (in object-oriented languages such as Java, Ruby, or JavaScript) or functions (in functional languages like LISP or functional-flavored JavaScript). The idea is that you will write a suite of tests that cover every piece of functionality your application is designed to provide under every possible circumstance — a lofty goal, to be sure, and one not easily achieved, but no less worth striving for due to its elusiveness.

Why test?

There are exactly as many reasons to write tests as there are potential mistakes in your code. Less prosaically though, we can enumerate some of the benefits as follows:

- **Fewer surprises** – By writing a wide variety of test cases, you can observe how your application behaves when presented with many kinds of data. And when one of your users reports a bug based on some data that you did not anticipate,

you can write a case for that data as well. For instance, the benefits of testing become readily apparent when writing a client for a web service (as we will see later on).

- **Regression proofing** – Say you are working on fixing a bug in your application. If you have a thorough test suite in place, and frequently run your tests after changing your code, that will go a long way to ensuring that any bug fixes or new features don't introduce any new bugs.
- **Baked-in documentation** – If you give your tests descriptive names (by the way, please give your tests descriptive names), then anybody who is reasonably code-savvy can look at your test suite and quickly surmise how your application is supposed to work.

Test-Driven Development

One concept that is tied very closely to unit testing is that of Test-Driven Development (TDD). TDD is generally advocated by proponents of agile programming, and is essentially the process of writing tests for a piece of functionality before even writing the code to implement that functionality. Some of the benefits to this kind of workflow are that it makes it considerably easier to achieve total test coverage of your code, and it forces you to think ahead about what your code is supposed to do rather than getting lost in the tall grass of implementation straightaway. TDD adherents will surely be able to tell you at length about all of its other advantages, but for our purposes here, it suffices to say that it can be a real boon to those who find it a natural way of working (and perhaps even more so to those who don't!).

Now that we have a sense of what unit testing is and what it can do for us, let's take a look at some of the tools that are available for unit testing JavaScript.