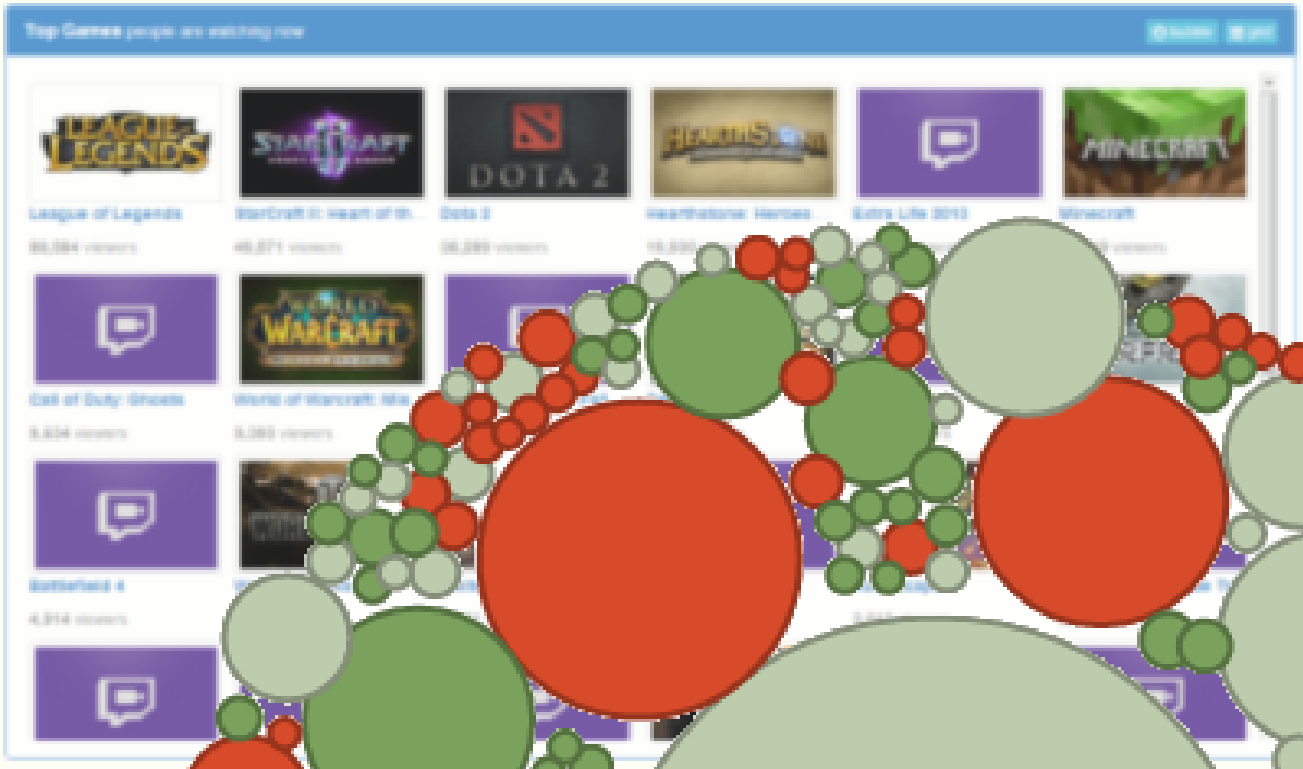
Building Scaleable Javascript Applications with AMD



Marionette Exposé

Jack Killilea

Marionette Exposé

Learn to write modular Javascript applicatons using Backbone Marionette and RequireJS/AMD

Jack Killilea

This book is for sale at http://leanpub.com/marionetteexpose

This version was published on 2013-12-02

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

1 Marionette Expose Preface

This book is about *my* open source journey. It was a trip. Creating a modern Javascript application these days is a wild ride through the world of open source software. For many developers, open source software can be an intimidating, yet very powerful. My plan is to take you on this journey and along the way you'll learn to embrace the ease and power of open source. In this book I'll be breaking down the creation of a Backbone.Marionette Application using the RequireJS module loading system. In doing so I'll be drawing upon several open source projects. There are so many ways to structure a JS application. With a multitude of frameworks, and libraries to choose from… where do you begin?

I have come up with my own approach that can be accurately described as a hybrid of the Rails Asset pipeline and the RequireJS module loading system. I'll be using a convention for saving my assets, much like the Rails Asset Pipeline but I'll also be using Asynchronous Module Definition (AMD) and a modular message bus to help create a highly decoupled architecture.

First, a little bit about me which led to me creating the app. I play PC games alot. I held rank one in 5v5 in World of Warcraft a while back when I played. I also play Starcraft at relatively high level of play (masters protoss), and play a lot of Dota 2. Basically I'm a nerd. Online gameplay can be live streamed and watched by thousands of people all around the world. Sites like Twitch.tv are popular and provide a service where users can monitor their favorite game's live steams. Users can both publish and consume live streaming content. After having spent so much time on the TwitchTV site I'd become accustomed to certain shortcomings in the user experience. Lots of full page post backs when selecting games and live streams… waiting for a stream player and chat views to load… etc.

So here was an idea for a project near and dear to me. Could I improve upon the TwitchTV user experience? I did a bit of research and learned that TwitchTV has an open source repository hosted on Github. TwitchTV also support a REST API that is well documented. Could I use the TwitchTV API to create a Single Page App and improve the user experience?

Should be quite interesting. Let the journey begin.

1.1 What is Twitch.tv ?

Twitch.tv is the world's leading video platform and community for gamers with more than 44 million visitors per month. They aim to connect gamers around the world by allowing them to broadcast, watch, and chat from everywhere they play.

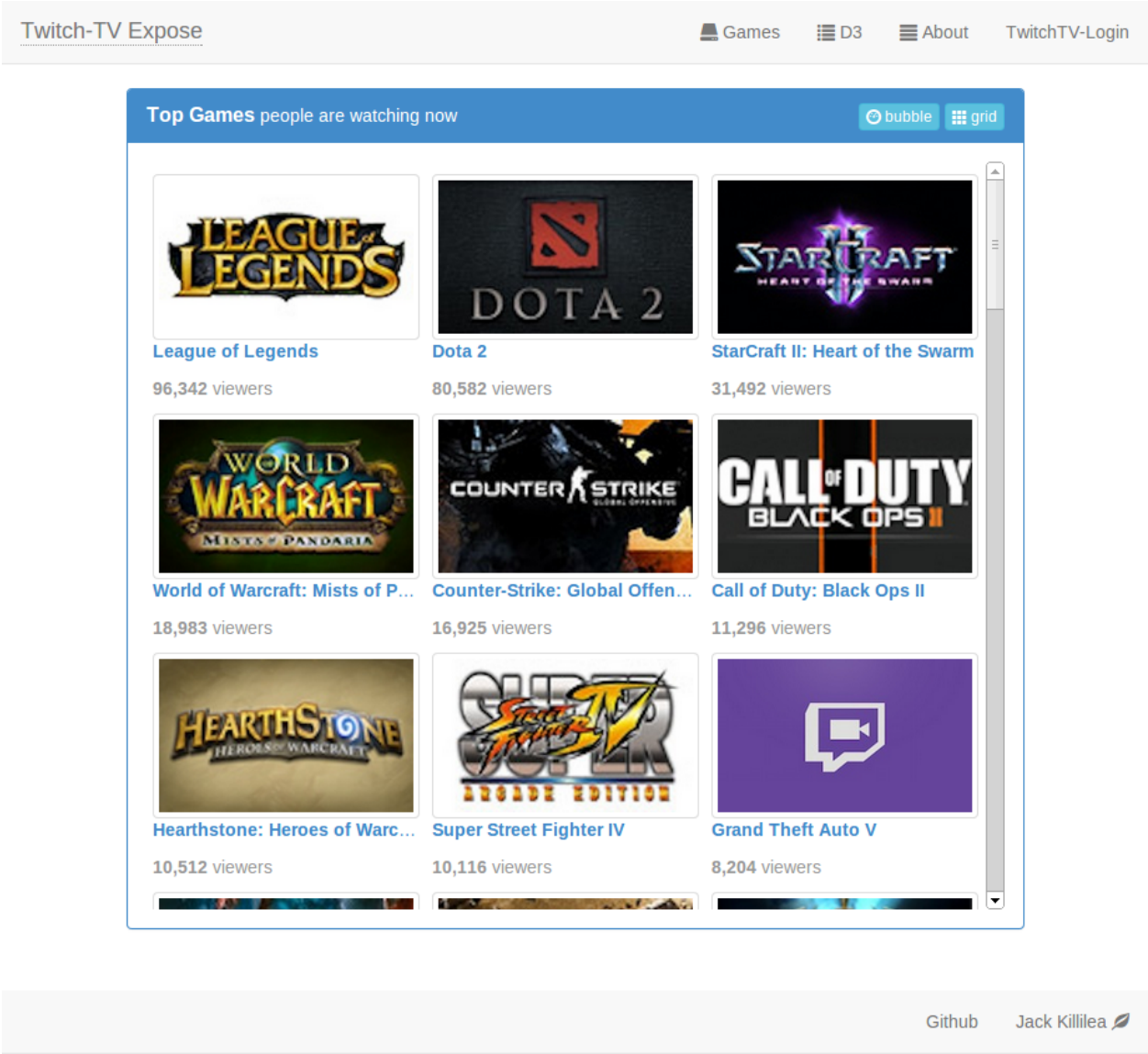
From what I saw, the UI was a bit lackluster. Clicking a game would bring you to another page where it would list out a bunch of streams, then another click would bring you to another page… I found the process very tiresome as I would click around and go stream to stream. As an avid user, I saw that this could be eliminated by creating a single-page app around it.

What better way to go about this than using Backbone.Marionette.

1.2 TwitchTVExpose - My Application

TwitchTVExpose is a Single Page Client App implementing TwitchTV's API functionality. Using Backbone.Marionette, RequireJS, Coffeescript, and a little D3 data-visualization to mix things up.

Check out the live site : TwitchTVExpose



Twitch Expose

Briefly, the main Games App presents a list of top streaming games by viewers. By selecting a game from the list view, the UI transitions into a `game:detail:view`. The detail view shows a single game with a list of live streams for that game. Clicking on one of live steams transitions to another layout with a player view and a chat view.

1.3 Why Did You Do It, Jack?

Over the summer I did a lot of research and came to the conclusion that I could rework Twitch's UI. I felt confident enough.

From there, I decided it would be a perfect first project to take what TwitchTV had, and do it better. I'd watched several of Brian Manns' BackboneRails tutorials and determined that Backbone.Marionette would be the framework I'd be using to build a responsive front-end app. I got to work all summer and this project was the outcome.

When I land at Twitch.tv, I just want to find my favorite games with their live streams and watch it quickly. This app takes twitchtv, and makes the selection and viewing process faster and a better experience for the user. No hassles with page-postbacks and such. All client-side JS, in a single-page app.

1.4 Working with Coffeescript!

Being the hipster nerd I am, I've moved on from writing JS.

I could go on and on about how great it is, but I dont want to be bashed. My thoughts are: If you want to read code in a book…. reading Coffeescript is far easier.

[Coffeescript][http://coffeescript.org/] is pretty gangster. It makes writing JS faster, in my opinion, and thus you will be reading all my source code in Coffeescript, so sadly, you'll have to deal with it.

For you nerds that are looking forward to learning a thing or two about it, great thats awesome.

For those of you distasteful few that just want to read JS, have no fear. I took the liberty of linking you to my github repo for my project that contains the javascript source as well.

[Read The Raw JS][https://github.com/xjackk/TwitchTVExpose]

Go crazy.

Let's look at the application's Architecture in detail as this is a key point to be understood.