rMIXr: how we learned to stop worrying and love the graph

Ben Fields and Sam Phippen
Fun and Plausible Solutions
London, UK
{ben | sam}@funandplausible.com

ABSTRACT

In this talk we present a case study in the use of the web audio APIs. Specifically, our use of them for the creation of a rapidly developed prototype application. The app, called rMIXr (http://rmixr.com), is a simple digital audio workstation (DAW) for fan remix contests. We created rMIXr in 48 hours at the Midem Hack Day in June 2015. We'll give a brief demo of the app and show multi-channel sync. We'll also show various effects as well as cutting/time-slicing.

Throughout the development process of this app we encountered difficulties in the web audio API and in some third party APIs. We overcame these difficulties through the use of some novel (dubious!) techniques. In particular we will speak to these issues:

• The existing web audio APIs have particular notions of how an audio graph should be built. We found violations of these notions in existing third party APIs. Where this abstraction boundary is violated, developers have to come up with their own notions of how to build a web audio graph. In this talk, we'll cover the abstractions we came up with for rMIXR, and speak to how they might generalize. We'll also cover some of the advantages and disadvantages that we discovered through these approaches.

• We'll speak to the construction of UIs that work for modification of the DAW and audio graph in real time. In particular, this will cover how we bind effect parameters into HTML controls. We'll also look at how we pass information from those controls down to the web audio graph and the layers between.

• As the app is for contests and fan use, being able to share state is a priority – for this and other design reasons we wanted state preserved locally. How best to achieve this with state across many windows was non-obvious. We'll cover the approach that we took, which specifically takes advantage of the share-ability of URLs on the web.

During the talk we will go through our solutions to these issues, showing by example a number of practical ways to get things done with the Web Audio APIs. We will walk through our abstractions and object models for building a fully functional DAW in the browser. These abstractions were necessary because we included a number of effects from third party libraries. Some of these effects libraries do not comply with the "source and sink" model that the web audio API predicates. Our abstractions gave us a more flexible graph structure that was critical to making this application work. While not perfect, it was certainly excellent for the rapid prototype scenario of the hackday. We'll also talk about some potential improvements to our abstractions. We will then show how to use localstorage to communicate across browser windows and how this approach can be harnessed to quickly create flexible UI elements with multiple windows.

Lastly, and perhaps most ridiculously, we will discuss our use of the URL as a means to locally store all of the state of the web application. Here we will speak to why this is both the best and worst idea. All of the state for a particular rMIXr session is stored in the # part of the URL. This means that a current rMIXr session can be shared between users simply by copying the URL and sending it to someone else.

We will aim to leave the audience with a collection of strategies they can use in their own web audio apps. We'll aim to provide a better understanding of some of the Web Audio API's structures, by way of example. The audience will also learn of an excellent means to discover what the maximum length of a URL can be before various modern browsers will crash (hint: it is quite large).

Web Audio Conference WAC-2016, April 4–6, 2016, Atlanta, USA.