5.2 Collaborative Apps

Music Machine 2015

5.2.1 Recap and Introduction

Music Machine 2015

5.2.1 Recap

- We can use Meteor to allow us to create full stack web applications that share code between client, server and database.
- We can create standard static sites more easily with live preview of code.
- We can also create dynamic sites more easily that integrate social features nicely.

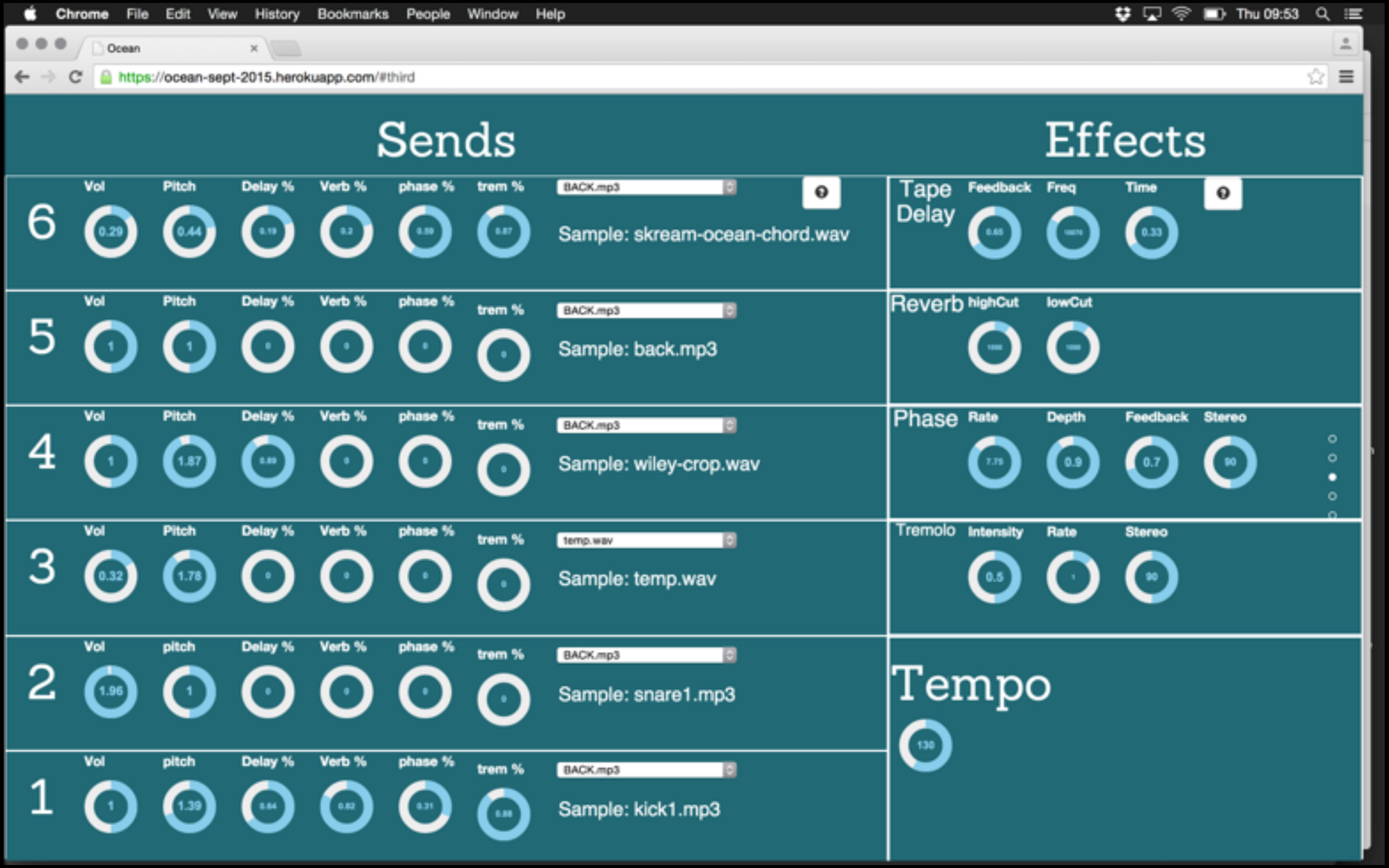
5.2.1 Collaboration

- But this isn’t what Meteor is best at.
- Meteor is great at making collaboration simple.
- You can make collaborative sites in a day or so that work across many devices including mobile.

5.2.1 Collaboration

- By the end of this session you will
    - Understand how to structure and build collaborative, real-time applications
    - Understand how to integrate the web audio API into Meteor applications using Maxim
    - Understand how to specifically set and get variables stored in the database, linked to user interface controls.

5.2.1 Collaborate Creative Music Application



6 — BACK.mp3 — Sample: skream-ocean-chord.wav — Vol 0.29, Pitch 0.44

5 — BACK.mp3 — Sample: back.mp3 — Vol 1, Pitch 1

4 — BACK.mp3 — Sample: wiley-crop.wav — Vol 1, Pitch 1.87

3 — temp.wav — Sample: temp.wav — Vol 0.32, Pitch 1.78

2 — BACK.mp3 — Sample: snare1.mp3 — Vol 1.96, pitch 1

1 — BACK.mp3 — Sample: kick1.mp3 — Vol 1, pitch 1.39

Tape Delay — Time 0.33

Phase — Depth 0.9, Feedback 0.7, Stereo 90

Tremolo — Intensity 0.5, Rate 1, Stereo 90

Tempo 130

OCEAN by Robin Hunter

- Creative Computing student at Goldsmiths.
- Built as part of his degree programme.
  - Uses web audio to allow users to collaborate on the same piece of music remotely.
  - Can grab audio from Youtube videos and automagically insert it into sessions.
  - Ocean Demo

5.2.2 Project Structure

Music Machine 2015

5.2.2
Project Structure

- We’re going to be adding our own JavaScript libraries that contain functionality we want to use
- We’re doing this instead of adding meteor packages, as often, packages might be broken, and it can take more time to do what we need
- We’re not going to need user permissions - autopublish works perfectly well for these types of applications

5.2.2 Project Structure

- Our JavaScript Libs need to go in the Client Folder
- We'll also put our meteor templates in their
- We will then use the main html and js files to organise how these are accessed
- Functions from our JS files in Client/ will be called from both our template and our main meteor JS file.

5.2.2
Project Structure

- Things to Remember :
  - JavaScript functions need to be globally scoped if you want to use them this way.
  - Create some global functions that encapsulate more complex interaction - you can then access these in interactive

5.2.3 Adding Sound

Music Machine 2015

5.2.3 Adding Sound

- The HTML5 webAudio framework is a great way to do sound interaction
- However, it is updated often and code breaks all the time
  - Most meteor packages struggle to provide even the most basic audio support in a reliable way
- Maxim.js (from Goldsmiths) is a solid, up-to-date way of making sure you can do complex creative applications with sound, without the pain.

5.2.3 Adding Sound

- Make sure maxim.js is in your Client directory
- Create a new JS file.
- You don't need to include the file anywhere - Meteor takes care of that
    - Be VERY careful and specific when specifying global function names. It's not usually advised, but this time we'll get away with it.

5.2.3 Code Review

```
//Create an Audio Context

acontext = new webkitAudioContext() || new AudioContext;

//Now we can create an instance of our waveform generator and play it.

waveform = new Synth(acontext);

//Or create a sampler and load a sound into it

maxim1 = new Maxim();

player1 = maxim1.loadFile("drums1.wav");

player1.loop
```

5.2.4 Adding Interaction

Music Machine 2015

5.2.4 Adding Interaction

- We can use buttons and sliders to create collaborative interaction
- We could call JavaScript functions with ‘onclick’
- But it’s better to use get and set methods so that we can insert the button / slider state into our mongo collection

5.2.4 Calling Functions Based on Variables

```
  "startdac": function () {
// Let’s create a variable and use it to search for a key in the database
// Our collection is called MusicMachine
    var starter = MusicMachine.findOne();
    if (starter) {//check to make sure something was returned
      if (starter.start==1) {//start is a variable that we have retrieved fro the colleciton
        playAll();//If it’s equal to 1 we call our JS function


      }
    }
```

5.2.4 Setting Variables

```
//when a button called ‘startButton’ is clicked

"click button.startButton": function () {

//set the value of ‘startdac’ to 1

      Session.set('startdac', 1);

// Search for the appropriate key so we can store the value of startdac in the
database

      var val = MusicMachine.findOne({});

//update the variable

      MusicMachine.update({ _id: val._id }, {$set: {start: 1}});

    },
```

5.2.5 Collaboration

Music Machine 2015

5.2.5 Making it Collaborative

- Now that we have values stored in the database, we can create a range of sounds and add them to the project
- Then we can mix these sounds together, and vary their properties
- Because they are stored in the database, updates from one local session will impact on updates from another one!

5.2.5 Making it Collaborative

- Recap
  - We can add our own libraries.
  - We can store variables in the database discretely.
  - We can get and set these methods that call functions in response to interaction across the internet.