Group 20

Sean Hinchee

Sam Westerlund

Language(s) used: Node.JS, HTML(5), Javascript

External Libraries: None

To Run:

*cd* to the base project directory.

Run *node server.js*

Checklist

* Templating system used within \*.html files
* Statefulness without cookies or PHP
* State-machine-esque handler system
* Unified, monolithic design within primary, listening, function
* Canvas/image integration from browser to the server to a new browser
* No page redirection or messy url’s, all internal management done through POST/GET (again, without PHP)
* Fully asynchronous system
* Durable design with little room for errors
  + State-machine-like design ensures transitions between states within the client are controlled and systematic (See: *ordo*)
* Purposefully limited backend with GET only having two cases (base page and picture requests)
* Self-hosting web server, does not require Apache, NGINX, nor any other existent web server

Group 20

Sean Hinchee

Sam Westerlund

Node.Paint – An exploration of Node.JS programming within a stateful web application.

Table of Contents

1. Overview
   1. Initial considerations
   2. Initial design
2. New and Complex
   1. Introduction to Node.JS
   2. Self-serving web applications
   3. Asynchronous Node.JS programming
   4. Templating and injection techniques (initial and final)
   5. Statefulness without cookies or PHP
3. Bloom’s Taxonomy
   1. Language choice
   2. Asynchronous systems
   3. Web applications (new and old)
   4. Front and Back-end unification
   5. Design complications and implementation difficulties

Overview

Node.Paint was created with the intent of making an interactive web application that one user could connect, paint using their cursor within a canvas along with an answer. The answer would be accepted by the server as an image and answer pair. A secondary user could enter a session identical to a submitted image and answer pair and attempt to guess the answer to the image and be informed whether they were correct, or not.

After brief conference, we concluded to use Node.JS as our language of choice due to both group members not having used Node.JS before and looking to expand our experiences. Furthermore, the decision to create a web application was decided to the still infantile experience both members possess with web development. Portfolio one would have been both of our first formal web applications so we looked to design and build a different model of web application with a different take on infrastructure while maintaining a well-designed system. The two portfolios, should there be any doubt, are wholly separate.

Portfolio one was designed in a more systems-oriented mindset, Node.Paint was designed to be a wholly web-based application with a standalone, non-dependent infrastructure. As such, many particulars and procedures within Node.JS had to be learned to work around crutches such as cookies and PHP which may be traditionally used for certain elements of web applications. Statefulness was a requirement since a given “Session” had to possess a guessable/drawable state regardless of the system accessing the server.

Multiple over-arching designs were cycled through, with the final result being a logically-monolithic, state-based design, taking cues from state machine systems.