Articles 742b: Networks & Transactions
Spring 2014
Green Hall, Room 209/210
Tuesdays 1:30–5:30

Class website: http://art.yale.edu/Art742b
Class email list: networks1@panlists.yale.edu

Dan Michaelson: dan.michaelson@yale.edu
Google Talk/Jabber: dan@linkedbyair.net
AIM or Skype: danmichaelson
Voice/SMS: 203-606-4492

TA:
Nir Bitton

Class hypothesis

How can graphic design influence and be influenced by the unpredictable encounters between one group and another? Or between quantities of unknown users on one side, and vast webs of fluctuating information on the other? In this course students develop typographies, visual languages, and motion vocabularies appropriate for these pervasive conditions of the modern world, found in experiences as varied as Facebook, YouTube “supercuts,” the game of chess, automated stock trading, and the organization and speech patterns of political movements. The course posits that designed form may sometimes be visible, and at other times be relational or latent rather than directly seen.

Class structure and discussion criteria

This class is primarily a studio course. Your work is critiqued as graphic design – as a part of your own design practice. It is not critiqued primarily as code, although the overall elegance of a design may relate to the fit between your project's visible design and its code. Nor is it critiqued for its salability in any "app marketplace," nor for its potential popularity in the marketplace of websites, nor for filling some perceived gap in the universe of existing website functionality. Your innovation should take place at the level of design.

But the course also includes a programming lab in which fundamentals of coding are taught through hands-on work most weeks. Only by programming can you test your project's design against human factors and the other externalities and network effects which are at the heart of this studio. Therefore, it's essential to begin programming quickly, and to test your project often by using it yourself and asking others to use it – even if the code doesn't completely implement your design intent yet. Furthermore, by programming, even when it's difficult and even if you decide not to program your own work in the future, you gain a deep understanding for the flows, protocols, and structures that are the carrier wave for your design. No previous programming experience is expected.

Weekly reading discussions from a range of sources complete a triangle of design, practice, and theory. Readings will be distributed most weeks; we will discuss the reading in class the following week, so you should prepare questions and observations as you read.

You are required to post your work each week on our class page. In addition, each week one student will be selected to post one relevant reading, visual example, or link.

Attendance each week is essential, as is weekly programming practice between classes, as well as weekly design progress.

Why Node?

Node is software that creates a server. It interprets code you write in JavaScript. With this it processes input requests from client connections (such as a web browser), and sends output responses back to them.

Node is relatively low-level and real-time software. You will be exposed to the nature of server protocols themselves. Other competing platforms, such as PHP, Ruby, or Java, tend to hide this from you. Nevertheless, to handle those protocols in Node requires only a few lines of code.

With Node, you'll create highly networked projects that reach out from themselves; not, generally, applications (like a "Processing sketch") that exist within the boundaries of a single computer or screen. Furthermore, Node can act as a web server, but it can just as easily be a real-time chat server or work at many other protocol levels – something which isn't possible with most other platforms.

JavaScript is best known as a language used for code that runs in web browsers, whereas other languages (PHP, Ruby, Java) are more often used on the server. Because Node uses JavaScript on the server, the JavaScript you learn in Node will also be useful to you when it comes time to add code on the browser side. In the past when learning to program PHP on the server, students have faced a hurdle when they want to later add JavaScript code on the browser side. In turn, the JavaScript you learn can be used in many other environments: PhoneGap or Titanium (mobile and desktop apps), scripting InDesign, Illustrator, and After Effects, or in embedded computing or physical computing by running Node on a device such as the Raspberry Pi.

Finally, Node is a very modern platform with much development and excitement around it. Its rise parallels another important trend, the rise of complex JavaScript web applications based on client-side platforms such as Backbone, Angular, and the hybrid solution Meteor.

Required software

Google Chrome. Safari and Firefox are also good, but the current version of Chrome includes a JavaScript/HTML/CSS debugging tool that is just a bit easier to use than the ones in the other two browsers.

Required accounts

GitHub. GitHub is a shared code repository that supports the Git versioning system. Sometimes I will give you starter code to clone from here. And you can also share your code to here. Sign up for a free account and **add your GitHub username to our class page.**

Cloud9. Cloud9 is a code editor and development editor in the cloud. All editing can be done within Chrome. Static files (HTML, client-side JavaScript, CSS) are hosted for public viewing immediately. Server code (Node) can also be run within Cloud9 and is available for public viewing at least until there has been an hour of no activity. **Sign up for a free account using the GitHub single-signon button** (don't create a separate Cloud9 account and password).

Optional: Heroku. To more permanently host server-side code, or to get your own domain, Cloud9 integrates with Heroku's hosting service.

Required programming books

We will refer to chapters from these books as places to review each week's lab. You should obtain both books now.

Pedro Teixeira. *Professional Node.js.* John Wiley & Sons, 2013.

Marijn Haverbeke. *Eloquent JavaScript.* No Starch Press, 2011.

Both books are available from Amazon Prime, or electronically via your free Yale Safari Online subscription (see below), Kindle, or iBooks. *Eloquent JavaScript* is also available as a free HTML book (with some interactivity) at eloquentjavascript.net.

Other online books and resources

(thanks Laurel Schwulst)

Shay Howe. A Beginner's Guide to HTML & CSS and An Advanced Guide to HTML & CSS. Free online books derived from classes at The Starter League.

Yale has a free subscription to Lynda.com.

Yale has a free subscription to Safari Books Online (alternate link), an extensive library of ebook versions of technology books.

Elisabeth Robson and Eric Freeman. *Headfirst HTML and CSS.* O'Reilly, 2005. A very beginner-oriented, fun if that's your thing, introduction to HTML and CSS. Available via your free Safari Books Online subscription, or we have a PDF available on request.

W3Schools. Tutorials and look-up references for HTML and CSS, and for JavaScript. Also includes online "try it yourself" windows.

DocHub. Quickly look up CSS, HTML, JavaScript, and jQuery documentation.

Stack Overflow. Often a solid place to search for troubleshooting. Pro tips: Include the library or platform you're dealing with in your search query, such as "node.js" or "jquery". Try putting the error message, if appropriate, in quotes, minus parts specific to your own code.

A List Apart. CSS tricks and tips.

Code libraries

All the code libraries on this page can be helpful for the kinds of projects in this class.

jQuery. Ubiquitous, even indispensable JavaScript helper library for your browser-side code (not for Node).

Reset CSS. Stylesheet that removes all the default browser styles from all HTML tags. Can provide a nice “clean slate” for your CSS. But, try a bit without this first.

Normalize.css. Makes styles more consistent on different browsers, fixes certain browser bugs and annoying CSS quirks. Does not actually “reset” the styles of HTML tags to nothing.

Only go below here if you know you need these or want to experiment.

Underscore.js. Additional useful JavaScript functions especially for browser-side code. A few of these functions are already built into Node’s JavaScript engine (Node uses a newer version of JavaScript than some browsers); or it’s possible to install Underscore in Node too.

Paper.js. Powerful JavaScript graphics, animation, and interaction library. For browser-side code, although work is underway to make it compatible with Node for server-side graphics and animation.

Studio assignments

Assignment 1. Alphabet.

What is an alphabet in the world of your work at Yale, of your thesis?

Using HTML and CSS, create an alphabet, a set of twenty-six things.

Before class, post a link from the Cloud9 preview, to our class page.

Assignment 2. Responses.

Consider a basic dynamic system, again in the universe of your interests and approaches so far at Yale.

Create a simple Node server. It should accept input from the URL request, such as

/1001/triangles

You decide what the input parameters are.

Create HTML and CSS output in the browser, that is different for each set of permitted inputs.

Assignment 3. Input-output.

We create a more realtime system. Using HTML and JavaScript, accept input from users. Send that input to Node using a WebSocket. Use Node to translate that input in some way, and optionally to persist data on the server so that future users can access it. Broadcast output to all users of the system, also using a WebSocket.

The input and the output can take any visual or typographical form. Consider how your system works to structure, incentivize, limit, or promote particular kinds of input and output. How do you expect it to evolve over time through its patterns of use?