Programming as Writing: Syllabus

`Last updated: 9.11.2018`

“Every language is a special way of looking at the world”

Fall 2018
Art 749a: Programming as Writing
Yale University, 107 Green Hall
Tuesdays 1:30–5:20pm
http://suddenly.rocks

Instructor: Laurel Schwulst, laurel@linkedbyair.net
TA: Rosa McElheny, rosa.mcelheny@yale.edu

Questions

- What is programming?
- What is writing?
- Is programming like writing?
- Is writing like programming?
- Is coding writing?
- Is writing coding?
- (Are programming and coding the same?)
- Why does one create a program?
- Why does one write an essay? A poem? A letter?
- What is interactivity?
- What is hypertext?
- Is literature like hypertext?
- Can something be more interactive than something else?
- How does writing influence programming?
- How does programming influence writing?
- Can coding be creative?
- How much subjectivity can a computer program sustain?
- Are programming languages really languages?
- Is the language we speak some sort of programming language?
- How do you define a language?
- Can programming be difficult to read or illegible like handwriting?

- How might generosity affect a program?
- Can a program be an poem, an essay, a conversation?
- Can you rhyme in code?
- Does code have a beginning, middle, and end? Can code tell a story?
- What's the purpose of a poem?
- Does a poem have to be made of words?
- How could a poem be different as web-based or networked?
- How does a piece of writing die?
- When a website dies, does it stop being relevant?
- While websites on the www can be seen worldwide, does code have some sort of geopolitical power?
- Is programming a limited language for most of us who are not a professional programmer?
- If the physical world disappeared today, would the internet still exist? Could the internet recreate the physical world?
- People who write literature are into genres. Are there coders who code in certain genres? Or people who make websites in specific genres?
- Can a website be archived or inherited? How good does the writing have to be on a website to be preserved?
- Is there such a thing as coding language literacy? How can you know if you're literate coder?

Overview

"Learning to code through reading and writing" — This studio course introduces fundamental concepts of programming for the web. As a participant, you will develop technical skills through the development of your own writing. The course believes that programs should be written not only for computers to process but for humans to read. While best practices are discussed, a variety of techniques that consider craft, tone, and style—challenging the notion of a singular, universal method—will be discussed and explored. After being introduced to document structuring and semantic HTML, you will navigate from client-side to server-side programming by learning JavaScript and PHP. In this course, your writing surfaces include not only a forward-facing web application and its constituent code, but also the code's annotation, written for a future reader.

Participants

This course is intended for first-year students with little or no programming experience. It's also a prerequisite for Networks and Transactions in the Spring.

Course design & projects

In this class, you'll focus on your own writing.

Each week, you'll creatively write (poems, prose, proems, manifestos, etc.). And each week, we'll study a technical skill—a programming language or a specific aspect of one. Whatever you write each week, no matter its length and style, you'll approach with the technical skill—seeing how it can be used to illuminate, distribute, elucidate, change, recreate, etc. your writing.

In other words, there are no large projects in this course. Instead, you will make small weekly projects based on writing/poetry prompts. You will write poems all semester, and the format of these poems will be dependent on the web development skills you are learning.

In addition to your weekly poems (which are the focus of this class), you'll also create two collections for your writing:

1. Journal — This will house all your writing in this course, from process to final and anything in-between. You might want to ask yourself: If you could design your own journal, what would it look like? How might it present your writing? What would it feel like to write in? Could your journal be generative or surprise you, either through its juxtapositions or possible trajectories? This can be a private, public, or somewhere in-between journal.

2. Anthology — This will house a curated selection of your writings from this course. In other words, you will select specific writings from your journal for your anthology. Consider your works together collectively to form a thoughtful body of work. How, when presented together as a website, can they be more resonant? And could your anthology be designed as a tool in reading aloud or performing your poems?

Class design

This course will meet for 15 individual classes.

Each class will likely follow this format:

Part 1
- 1 minute: Post your project URL
- 15 minutes: Constraint-based writing exercise
- 30 minutes: Independent free writing
- 30 minutes: Discussion of readings with your slides

Part 2
- Looking at project URLs, discussion
- Tutorial
  - Overall demo
  - Individual meetings

Skills

In this class, we will cover a web development toolkit:

- HTML
- JavaScript
- PHP
- Kirby CMS (PHP-based)

CSS will not be taught in this class, but please feel free to learn it on your own and incorporate it into your projects.

Calendar

Week	Day	Agenda
1	Tues, Sept 4	**Introduction** Guest: Christoph Knoth
2	Tues, Sept 11	**Writing with HTML**
3	Tues, Sept 18	**Writing with JavaScript (variables)**
4	Tues, Sept 25	**Writing with JavaScript (loops)**
5	Tues, Oct 2	**Writing with JavaScript (functions)**
6	Tues, Oct 9	**Writing with JavaScript (methods)** Guest: Harm van den Dorpel
7	Tues, Oct 16	**Writing with PHP with Tim Ripper (day 1)**
8	Tues, Oct 23	**Writing with PHP with Tim Ripper (day 2)**
9	Tues, Oct 30	**Writing with PHP with Tim Ripper (day 3)**
10	Tues, Nov 6	**Writing with & within Kirby CMS**
11	Tues, Nov 13	**Writing with & within Kirby CMS**
		~ November Recess ~
12	Tues, Nov 20	**Writing with HTML (again)** Guest: TK
13	Tues, Nov 27	**Writing with JavaScript (again)**

14	Tues, Dec 4	**Writing with PHP (again)**
15	Tues, Dec 11	**Poetry Readings**

Readings

I start this course by bringing two readings into the class (Butler & Delany in conversation and What can a website be?).

Each week, a different student chooses the reading(s). The readings should be somehow linked to the ones previous, in a "primitive hypertext" way. (This term, primitive hypertext, is from Octavia Butler, see below.) Ideally, the chosen readings haven't been read before by many in the class—from a different field or a fresh perspective.

"I generally have four or five books open around the house—I live alone; I can do this—and they are not books on the same subject. They don't relate to each other in any particular way, and the ideas they present bounce off one another. And I like this effect. I also listen to audio-books, and I'll go out for my morning walk with tapes from two very different audio-books, and let those ideas bounce off each other, simmer, reproduce in some odd way, so that I come up with ideas that I might not have come up with if I had simply stuck to one book until I was done with it and then gone and picked up another."

To respond to readings, you will place an image, text, video, etc. in the class Google Slides document for discussion, as a jumping-off point. This should relate or respond to the readings in some way, but it doesn't need to be direct.

Emails

I would like to be in email communication with you all for at least the first part of this course (September–October). Please write me an email (laurel@linkedbyair.net) with your biggest questions and anything you've been strongly thinking about. They can be related to programming and writing, but they don't have to be.

Resources

All technologies introduced in class are well documented online. While general overview of skills are given in class, the best learning of these skills happens through practice on projects you care deeply about. Since the web and its constituent code is constantly changing, there is no one resource that is best. Instead, you should aim to absorb resources from a variety of sources and put them to use through trial and error. If you are having difficulties, please take time to first troubleshoot online by yourself.

If you find yourself stuck while writing code (which is extremely common–even for the best programmers), first try breaking your problem down into smaller, more manageable parts. Search Google or Stack Overflow for how to solve those parts, one at a time. Remember that most of the time spent writing code will be fixing bugs. In fact, learning how to debug is what programming is all about! (And sometimes bugs will allow you to discover something new and never seen before.)

Attendance & Evaluation

Attendance is essential. Three or more absences will result in a failing grade. If you absolutely must miss class, please email me in advance.

As long as you attend class and are an active and curious participant, you will receive a passing grade. Therefore, how you use this class is up to you—please treat it as a space to explore an aspect of programming and writing you're deeply interested in.

While there will be in-class discussions, I understand that each student has their own way of participating. This class is designed with multiple communication pathways and mediums, allowing flexibility. But please speak to me individually if you ever would like clarity on your level of participation.

Materials

We won't be in a computer lab, so please bring your personal laptop to each class.

Academic Integrity

You will become familiar with using pre-existing language, images, and software as raw material while creating entirely new work. While making websites, we will explore which technologies could be appropriated and how to properly credit their inclusion.

From Academic Writing at MIT, "Writing Code":

"Writing code is similar to academic writing in that when you use or adapt code developed by someone else as part of your project, you must cite your source. However, instead of quoting or paraphrasing a source, you include an inline comment in the code. These comments not only ensure you are giving proper credit, but help with code understanding and debugging."

"You should not simply re-use code as the solution to an assignment. Like academic writing, your code can incorporate the ideas of others but should reflect your original approach to the problem."

You might consider retyping someone else's code instead of copying and pasting it. It might help you better learn. On that note, be careful about pasting huge blocks of code. I'd recommend doing things one step at a time so you really understand what each part is doing.

Literate Programming

This course takes inspiration from Donald Knuth's "Literate Programming":

"Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do."

"The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. He or she strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other."

Credits

This is a completely new class that has been generously shaped and supported by many, including Ayham Ghraowi ('17), Sheila Levrant de Bretteville ('64), Rosa McElheny ('19), Dan Michaelson ('02), and Matthew Wolff ('18).

Opening quote:
Clyde Kluckhohn, *Mirror for Man: The Relation of Anthropology to Modern Life*