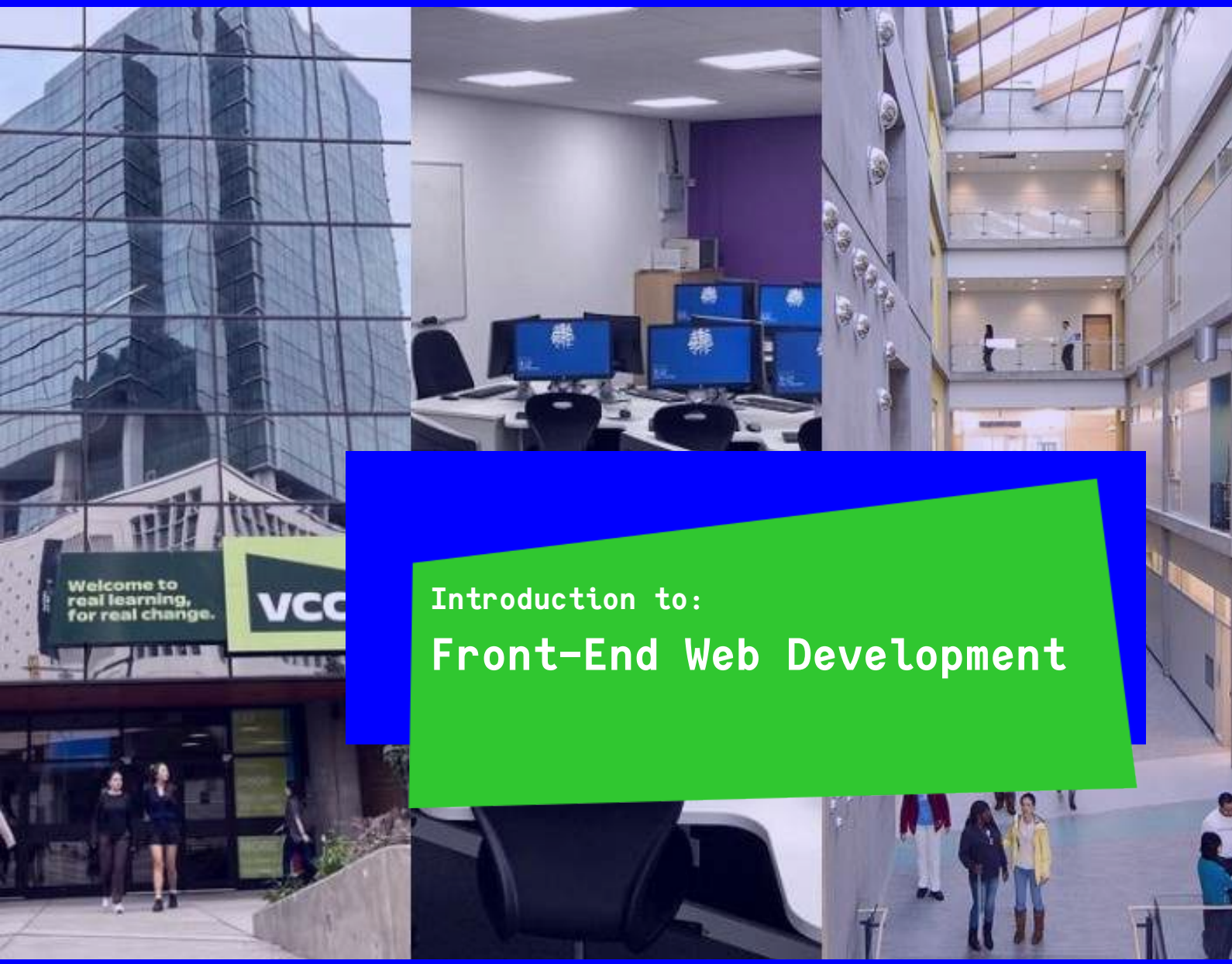
PIDP 3210: Curriculum Development
COURSE REFLECTION 2

Instruct



Introduction to:
Front-End Web Development

Course Developer: Yuri Tricys
Date: March 21st, 2025

Assignment 1, Part 2, Subject: “Should instructors be responsible for teaching 21st century skills such as active listening, time management or social perceptiveness as well as the subject matter content of their field?”

Objective

The focus of this exercise is a quote by Ben Collins-Sussman, an American software engineer, composer, and author known for co-creating the Subversion version control system and managing teams that power Google’s search infrastructure [1].

> *In software development, the learning never stops. You must constantly stay up-to-date with the latest technologies and programming languages. (Quotes on the importance of learning and growth in programming, quote 1)*

The quote is relevant to the question listed in the reflective exercise assignment description, in the sense that life-long learning is, according to the humanists, a learnable and teachable skill, and in the computer sciences, arguably a mandatory skill. ‘Arguably’ because the computer sciences are problematic.

The first problem is that learning computer sciences is hard.

A group of researchers, led by Yorah Bosse & Marco Aurelio Gerosa, studied experiences of first year programming students, and their professors, and noted in their 2017 paper, “Difficulties of Programming Learning from the Point of View of Students and Instructors” (it’s published in Portuguese), failure rates of 27.5% for classes that used Python and 54.9% in classes that used C.

Those rates are similar to rates found by Bennedsen and Caspersen, in their 2007 paper “Failure Rates in Introductory Programming”, and such rates are relatively high when considering that the students who take computer sciences generally sign up to take computer sciences.

Bosse & Gerosa reveal a lot of specific challenges, like syntax errors, declaration errors, error interpretation, and debugging problems — and I can relate to those problems having slugged through a lot of nasty code in my day, but the real culprit, in my view, is the second problem.

The second problem is that the knowledge base for computer sciences is incredibly wide. There are thousands of libraries, seemingly endless languages, hundreds of frameworks, and rabbit holes of every shape and size for almost every type of problem.

In a field like this one, knowing how to learn effectively is probably the most important skill (being persistent is the probably most important characteristic).

If life-long learning and learning effectively are ‘21st century skills’, like ‘active listening’, ‘time management’, or ‘social perceptiveness’, then there is certainly a strong case that an instructor of computer science related material should be responsible to teach them.

Reflective

As a life-long learner, this subject is important to me, not just because my journey to web development mastery is more than 10,000 hours old, and during that time I’ve learned a thing or two about how not learn (and therefore also about its antithesis), but also because my view, after all of that, is that only an insane person should consider becoming an expert web developer.

Okay, that sentence was an exercise in hyperbole, but I didn’t get heavily into the computer sciences until after the age of 40. Keeping insane hours was the only way I could cram 20 years of work into a 10 year time-frame.

The usual way to do it is to get an early comp-sci degree, then learn from a bunch of jaded, aged, and cranky developers, through osmosis, for the following twenty years. That’s the path that gives developers the most time for their expensive hobbies, like buying and selling property in San-Francisco, and founding start-ups.

Having said that, what Ben Collins-Sussman says about learning and programming is more relevant now than it every was, because the field is getting rapidly wider.

Interpretive

The question however is about responsibility. In our context, ‘should instructors be responsible …’, is similar to saying ‘should life-long learning and learning to learn be a part of the curriculum.’ There is a wide debate around such questions on other topics, like ‘should students be required to volunteer in their community’ [2] and if ‘High School Attendance Should be Voluntary’ [3].

What those topics have in common with ours is the commodity that in a free society is traded voluntarily for a perceived reward: time. Students forced to volunteer in their communities may not benefit from volunteering as much as they might from substitute activities, and the community might also benefit more if those students were to pursue those substitute activities.

In the high school attendance case, there is society wide agreement that a base-level of time should be spent ensuring students engage in an educational setting with other students, teachers, and administrators from their communities.

And this is where my own learning experience as a life-long learner and a master web developer is relevant: the time commitment required to learn to learn efficiently in the computer sciences is large.

For me, changing the rate at which I learn, and my ability to store and recall information is largely about technology development and use, and time.

I use highly specialized and customized tools to find, organize, store, and recall information. And the process is best described as a 'way of life'. For me personally, it's tremendously rewarding. I get the direct benefit of what economists call utility, in that I love living my life that way.

But not everyone is like me, and if everyone was the same, the world would be a pretty bleak place.

To the extent that life-long learning and learning to learn are 21st century skills, an instructor should not be responsible to teach them, unless the program he or she teaches promotes them and the student's sign up for it.

Post-secondary students should have the option of allocating their time in the way that suits them – and hopefully society – best.

Decisional

Life-long learning, continuous learning, embracing new challenges, and tackling difficult and sometimes seemingly useless material is mandatory if you want to achieve mastery in web development and you are a relatively normal person (by which I mean not Mozart); but, mastery in humanity is voluntary.

My approach then, is something along the lines of that Antoine de Saint-Exupery quote:

> *If you want to build a ship, don't drum up the men to gather wood, divide the work, and give orders. Instead, teach them to yearn for the vast and endless sea. (Antoine de Saint-Exupéry, para. 1)*

I hope to inspire those who would be as interested as I am in life-long learning and learning to learn, and I anticipate if they are, the inspiration and example I set, through my energy and passion, will be enough to 'teach them to yearn for the vast and endless sea.'

And, in terms of more practical ways to ease the burden of learning web development, both Collins & Dolan, 2015 and Bosse & Gerosa, 2019 are helpful.

Collins & Dolan offer four tips in their paper, "We must teach more effectively: here are four ways to get started", published in the National Library of Medicine in 2015:

- Design a course back to front
- Aim high—beyond just the facts
- Pose messy problems
- Expect students to talk, write, and collaborate

And Bosse & Gerosa, for their part, are full of excellent and pertinant advice. They advise to "omit some concepts and language details", for example, and to "explain using problems and similar exercises" and that the students they studied found that "the most difficult programming concepts are repetition, recursion, lists, pointers, passing parameters, abstract data types, and the use of libraries." (Bosse Gerosa, 2019)

While I may, if I become an instructor, refrain from overtly prompting students toward excessive amounts of academia in their lives, or building out high-end custom made software tools, deliberately designed to tweak the learning curve (by which I mean to flatten it 🙄), I will find ways to strategically increase the amount of learning achieved in the time spent learning it.

Footnotes

[1]: From "Ben Collins-Sussman," by Wikipedia, n.d. (https://en.wikipedia.org/wiki/Ben_Collins-Sussman). In the public domain.
[2]: See "Should Students Be Required to Volunteer in Their Community?" by the Center for Creative Placemaking, n.d. (https://centerforcreativeplacemaking.net/should-students-be-required-to-volunteer-in-their-community/).
[3]: See "High School Attendance Should be Voluntary" by Debate Nirvana, n.d. (https://debatenirvana.com/research/topics/high-school-attendance-should-be-voluntary/CON).

References

Codelamps.inc, (2024). "50 Inspiring Quotes About Programming That Will Motivate You to Code." Medium.com https://medium.com/@codelamps/50-inspiring-quotes-about-programming-that-will-motivate-you-to-code-8d5152a85b7b

Goodreads.com, (n.d). *Quote by Antoine de Saint-Exupéry: "If you want to build a ship, don't drum up the ..."* Goodreads.com https://www.goodreads.com/quotes/384067-if-you-want-to-build-a-ship-don-t-drum-up

Dolan, E. L., & Collins, J. P. (2015). We must teach more effectively: here are four ways to get started. *Molecular biology of the cell,* 26(12), 2151–2155. Retrieved From (https://doi.org/10.1091/mbc.E13-11-0675)

Bennedsen, Jens & Caspersen, Michael. (2007). Failure rates in introductory programming. *SIGCSE Bulletin.* 39. 32-36. 10.1145/1272848.1272879. Retrieve from: (https://www.researchgate.net/publication/220613179_Failure_rates_in_introductory_programming)

Bosse, Y, & Gerosa, M.A. (2017). Difficulties of Programming Learning from the Point of View of Students and Instructors. *IEEE Latin America Transactions.* 15. 2191-2199. 10.1109/TLA.2017.8070426. Retrieved From: (https://www.researchgate.net/publication/320768710_Difficulties_of_Programming_Learning_from_the_Point_of_View_of_Students_and_Instructors

Bosse, Y, & Gerosa, M.A. (2019). Pedagogical Content for Professors of Introductory Programming Courses. 10.1145/3304221.3319776. Retrieved From: (https://www.researchgate.net/publication/332299729_Pedagogical_Content_for_Professors_of_Introductory_Programming_Courses)