**CART 351 REFLECTION 1**
**Mollika Chakraborty**

Procrastination has always had a way of luring me but with Paul' Ford's article "What is code", I did "useful procrastination". The term might sound dramatic at first, but it does explain in some way the interestingly vibrant background patterns that the article has and some fun interactions like getting the scan codes for the pressed keys on your keyboard, being able to write code within the article, a unique character trying to mock you while you go through the pages too fast (or slow). Simulated circuitry that shows a demo of logic gates and the flow of current, depiction of a tree and (graphic trees), etc. The article is full of interesting visual elements, it would make you want to play with the elements, and you would have unknowingly passed 30 minutes of your time searching for other visual cues throughout the article or playing with the ones you came across while reading and spending too much time there.

I have always thought of programming as a technical field, and you must be extremely good at mathematics to be able to do programming. I have done various projects that sometimes involved programming or didn't but regardless I thought of it as an extremely technical side of a project and not creative at all. Creative, yes from the point of view of concept but its execution is always technical. When an efficient programmer like Paul Ford, calls himself someone weak at math, or not a "naturally gifted" coder, I loved the article instantly. This time I got the real motivation to finish this, I had to hear from a professional programmer who is not good at math, and he mentions that quite often. I wouldn't say that from a 38000-word article only this resonated with me, but it is one of the ideas that resonated with me that you don't need to be exceptionally good at math or naturally gifted to be able to code.

British Artist and Designer William Morris as mentioned in the article said, "You can't have art without resistance in the materials". Paul (the author) refers to code as art, the computer and its parts are the materials, which have complexities that need to be bypassed to look for the solution to a problem. This has given me a way to think of coding beyond its technicalities. It indeed is a process that requires patience, problem-solving, and creativity to be able to transfer abstract ideas into practical output. Therefore, it can be technical by nature, but it does mirror creative practices or at least our understanding of the creative practices, we all have gone through. For example, when we make a game, we all go through phases that make us think about how this is the worst game of our lives and we have no talent for any of it, losing patience because the console found 102 errors and we don't know where to start fixing them but it does come along, when it is complete it looks like an achievement and our faith in coding is restored.

Although the article does in a way have a compelling tone to it, it wants you to pursue coding as your career, how it will give you a salary hike that other jobs might not, why? We are surrounded by computers, as Paul mentions "Coding has been my life, and so has been yours", it gives you a moment to look around your surroundings and realize that he

is not wrong. Our dependency on technology and applications would propel us to learn programming to be able to find a place (employment) somewhere that would help us put bread on the table. It is not desired, but it is the reality, and it is the speed with which the world is progressing, the sooner we accept it and start learning the better it becomes (at least it's fun).

There are many interesting things in the article, it sure doesn't teach you how to code, and that's why it's not titled the same and is about "what is code". From introducing us to different types of languages and how some of them made it in the industry, became famous and we all know about them to how some never made it through that scrutiny but were there when computer languages were at their experimental stages. How software is developed in a company and what is the role of "the developer" there. How there were stricter forms of software product delivery mechanisms that slowly evolved into adapting AGILE methodology which is a lot more based on feedback and reviewing than strictly following instructions. What importance do different types of computer languages hold? Why it is important for a budding programmer or an experienced one to attend events and be at conferences where meaningful information is exchanged in the forms of learning and networking. This article is more of a journey, it gives you a little bit of history and a comparative analysis of the present environment. It highlights the decentralization of the software industry, where developers from countries like India and Hungary work with people at Silicon Valley which is quite an example of diversity and interconnectedness. I believe the decentralization fosters exchange of ideas, approaches, and perspectives which further enhances the development process with valuable insights.