How novices learn to code:

WHAT I'VE LEARNED Teaching in a Coding Bootcamp

IMPORTANT!

This presentation was created in 2020 and is shared solely for historical and educational reference. Market conditions, hiring practices, and educational outcomes in the software industry have changed significantly since that time.

The placement statistics referenced were accurate when originally presented but may no longer reflect current conditions. Readers are encouraged to independently verify up-to-date outcomes, placement data, and program effectiveness before making any educational or career decisions.

The views expressed reflect my personal perspective as of the date noted and do not represent any organization, past or present.

John Fulton, 12/30/2025

Why do you care?

[Helping a friend become a developer?]

[Onboarding of junior developers?]

[Mentoring new developers?]

[Giving advice to someone who wants to become a developer?]

JOHN FULTON

- 35+ years of being paid to program
- Assembly language to COBOL to C# and many things in between
- B.S., M.S., M.S.
- 10 years as an Adjunct Faculty Member – taught 42 sections of 10 different undergraduate CS courses
- 3+ years (10 cohorts) as an instructor in a coding bootcamp

Tech Elevator

Tech Elevator is an intensive in-person education provider helping individuals and companies acquire in-demand technology skills for the modern workforce.

Through our 14-week, full-time coding bootcamps, we teach students from diverse backgrounds to become software developers while also helping them to build necessary career-readiness skills and career connections through our nationally recognized Pathway Program™.

Founded in 2015, with a focus on quality and care in everything we do, we're proud of our outcomes. As an early member of the Council on Integrity Results in Reporting (www.cirr.org), Tech Elevator has taken a proactive approach to transparency. Our results speak to our focus on quality and on student success which has earned us a leading national bootcamp position based on the job placement rate of our grads.

95% Graduation Rate   94% Job Placement Rate   1000+ Grads Working in Tech

What I've learned...

- Or what I think I've learned
- Empirical, not research or experimentally based
- There is lots or research about how adults learn, how technical skills are built, how students with special needs learn, etc.
- All the points are important, but not all of them are important for all students

Not everyone is cut out for this

- Yes everyone can code
- Not everyone should do it for a living
- Some people's least appealing job
- Vetting candidates for problem solving, algebra, and motivation

Programmers learn by programming

- Reading, lecture, videos are wonderful
- Students need to build "muscle memory" for coding, debugging and testing
- It's only by using tools that you know how to use them
- Failure and correcting from failures seem essential to mastery

Some things are harder to learn than others

- Cliff events
  - Collection objects
  - Calls to functions with parameters
  - Encapsulation, Classes, and objects
  - Frameworks, Inversion of control

- Soak time

Repetition is critical

- Saying something is not teaching something
- About three repetitions of important points is about the right number
- And if the classroom was distracting, remote is even more distracting
- Not just one exercise using a concept, but "wax on, wax off"

Reviewing code is boring

- It's tempting to lead students through blocks of already-created code
- Better to live-code in front of them
- Better yet to have them code along with you.

Schadenfreude

- Students love to see failure
- It's one thing to tell them that all rpogrammers amek mistkes, it's another to ee it happen
- Students remember the instances when things go wrong

Programmers get stuck

- A dozen things can be wrong in any single line of code
- It's not easy to see what's wrong, even with experience
- Patient, experienced, knowledgeable help
- Could be peer support

Putting code in the right place is important

- Much of what we do beyond basic coding is organization
- Object oriented programming, design patterns, and frameworks

Programmer doubt themselves

- Programming is full of daunting and unpredicatable roadblocks
- Positve feedback on developing intuition and problem-solving
- Imposter syndrone is real

It's difficult to get students to use debugging

- "What does the dubugger say"
- Code is not a black box, but it's hard to convince students of that
- Drive the student back to the debugger

Problem solving skills and intuition are important

- This is a creative activity
- Ability to understand and abstract problem
- Practice different types of problems
- Develop intuition about how to view problems in terms of what the computer can do
- Learn to trust intuition

Programming is not Computer Science
Computer Science is not Programming

- Elements of each in the other
- Computer science is a rich, multi-factied dicipline
- Programming is largely a craft and trade
- That's why we can teach the elements of it in a vocation setting in 14 weeks

Teach programming, you need to teach three things at once

- Think of it as juggling three balls...

- Syntax
- Tools
- Problem solving

Programmers don't get jobs because they can program

- We don't know how to hire programmers
- We want programmers to program, but we hire them largely because they can talk about programming
- So in addition to teaching them to program, they have to learn to talk about programming
- And they need resumes, and LinkedIn pages, and Elevator pitches

Everything else can be forgiven…

- Respect for each individual, all day – every day
- Passion and desire to see each student succeed to the best of their ability

SUMMARY

[This isn't for everyone]

[Programmers learn by doing]

[Creative and intuitive]

[Positive and encouraging support]

Questions?

John Fulton

john@techelevator.com

https://www.linkedin.com/in/johntechelevator/

614-565-8382