

# Lessons Learned from Teaching at a Coding Bootcamp

# What's Coming

- A Year And A Half in Review
- The Biz
- The Task of Teaching
- The People
- The Hardest Things about Teaching
- Recommended Books

# A Year and a Half in Review

- Mar 2016 (3) – Doing my best
- June 2016 (3) – What it is like to teach people who learn at roughly the same pace
- Oct 2016 (7) – Medium size classes are easier than small classes and what it is like to have students who can go at radically different speeds
- Mar 2017 (8) – Eric gets his groove
- July 2017 (7) – Piloting someone's new curriculum is entirely unpleasant

**THE BIZ**

# Do They Work?

- “Why Coding Bootcamps Don't Work” – (Inc, <https://www.inc.com/geoffrey-james/why-coding-bootcamps-dont-work.html>)
- I have seen them work. So yes.
- I think we should avoid confusing being educated with going to school.

# Are Code Schools a Replacement for a 4 Year Degree

- No, in terms of material covered, code schools are not a replacement for CS degrees.
- Yes, in terms of getting into a programming career, they certainly can be.

"In the last five years, dozens of schools have popped up offering an unusual promise: Even humanities graduates can learn how to code in a few months and join the high-paying digital economy." - NY Times

THEN WHY ARE THEY FAILING?



- Failure because of lack of innovation? - <https://techcrunch.com/2017/08/26/an-insiders-take-on-the-future-of-coding-bootcamps/>
- Dev Bootcamp – “...we were simply unable to find a sustainable business model”.
- The Iron Yard - “...ultimately unable to sustain our current business model.”

# Things to Think About

- Pay for experienced developers or use less experienced developers and run a greater risk of a sub-par experience?
- In-person, online, or blended?
- Intensive or drawn out?
- What will you provide that a book or some videos can't?
- One campus or many?
- Back-end? Front-end? Mobile? Reality – before they get into it, most students have no idea what they will actually enjoy.

# What Can an In-Person Bootcamp *Potentially* Do Better?

- Provide better career support
- Help students one-on-one
- Provide a daily routine and structure
- Give the students a group of peers to collaborate with

# How Are You Going to Measure Success?

- Learning goals?
- Jobs?
- Student satisfaction?

# **THE TASK OF TEACHING**

# Want to Be a Teacher?

- Obvious qualities/skills you must have: public speaking, empathy, the ability to explain things
- Teaching at a coding school is extremely different than speaking at meetups
- The Curse of Knowledge is real
- Make sure the school has a plan to onboard you as a teacher, because developing and teaching are very different things
- Related, just because you are a good developer doesn't mean that you will be a good teacher.

# The Discipline of Teaching

- You know, they have degrees for this sort of thing.
- Two primary areas to think about:
  - Designing curriculum
  - Designing the classroom

# Curriculum Design Nuggets

- You need to design not only “what” but “how much”
- There is a tension between practical practice and grasping theoretical concepts
- There needs to be an appropriate amount of repetition
- The hardest thing to do: create a curriculum that challenges fast and slow learners appropriately *at the same time*



# Thoughts on Good Documentation

- If you want to teach someone, you have to find stuff for them to read/watch.
- Easy to understand and follows a slow buildup, e.g., JavaScript for Kids.
- Easy to understand but doesn't define a path for a beginner, e.g., the React documentation.
- Hard to understand and impossible for most beginners, e.g., the Redux documentation.

# A Good Educational Experience

- provide structure for learning
- materials at a proper level (small conceptual buildup is better than large leaps)
- graded exercises with some repetition, often in different contexts
- supportive environment

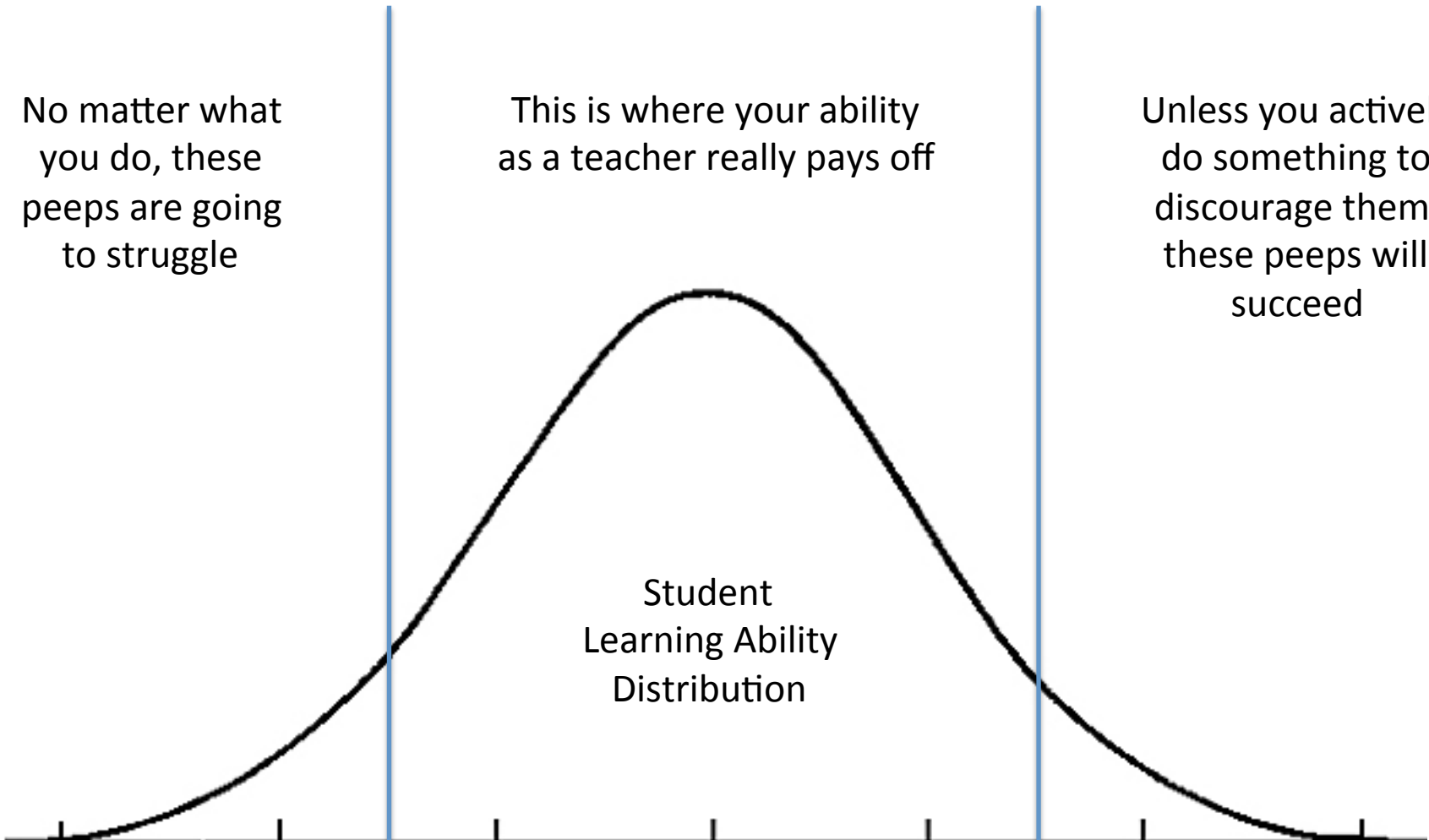
# Teacher Awesomeness

No matter what  
you do, these  
peeps are going  
to struggle

This is where your ability  
as a teacher really pays off

Unless you actively  
do something to  
discourage them,  
these peeps will  
succeed

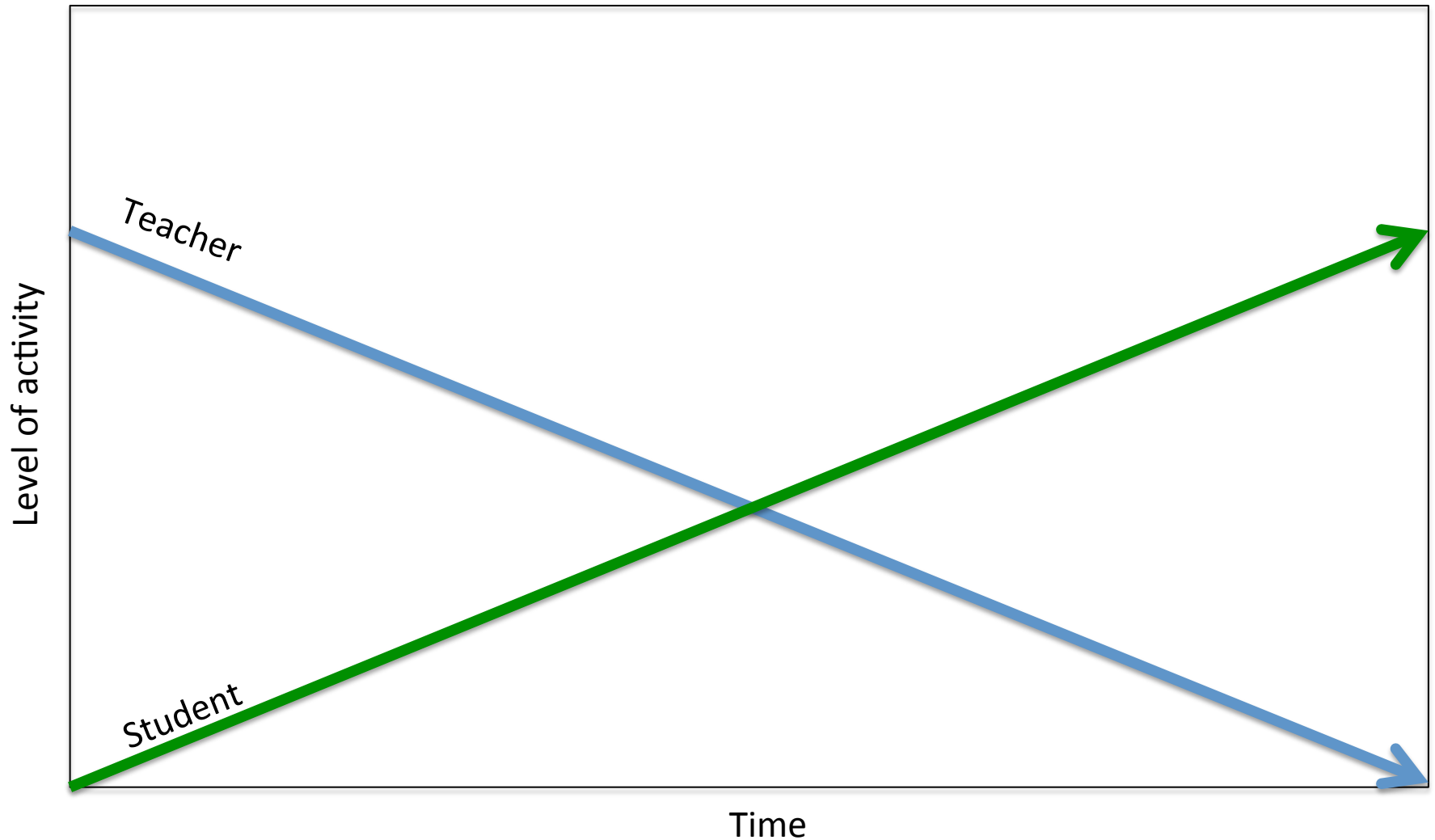
Student  
Learning Ability  
Distribution



# Designing the Classroom

- It's about the learning, not about the teaching
- Ratio

# Ratio – The Bowtie



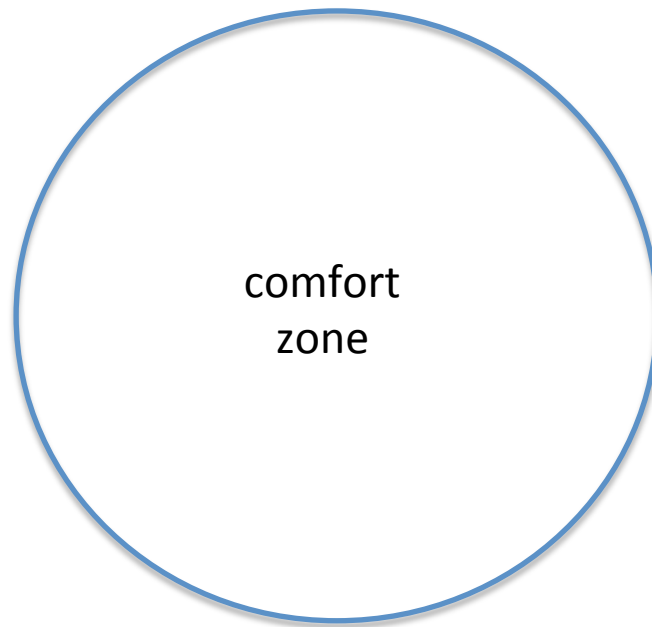
# Designing the Classroom

- It's about the learning, not about the teaching
- Ratio
- Pacing - Keep your students uncomfortable but don't let them panic



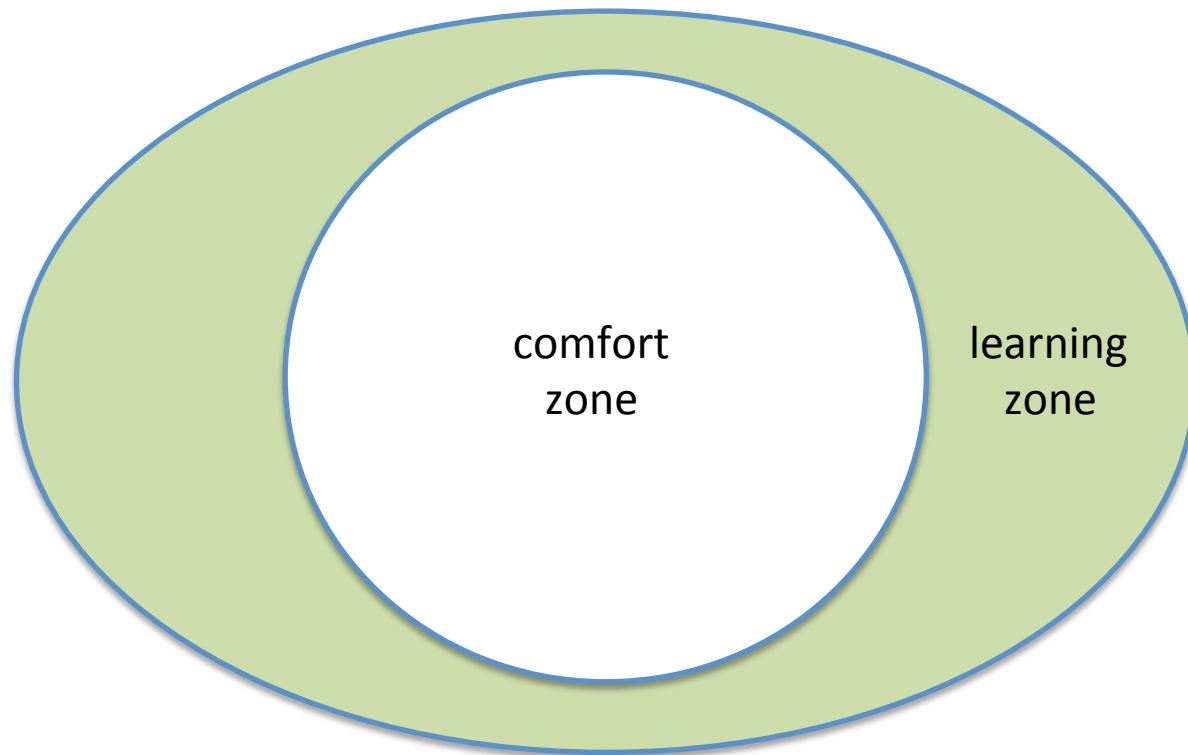
**DON'T PANIC**

# Don't Panic

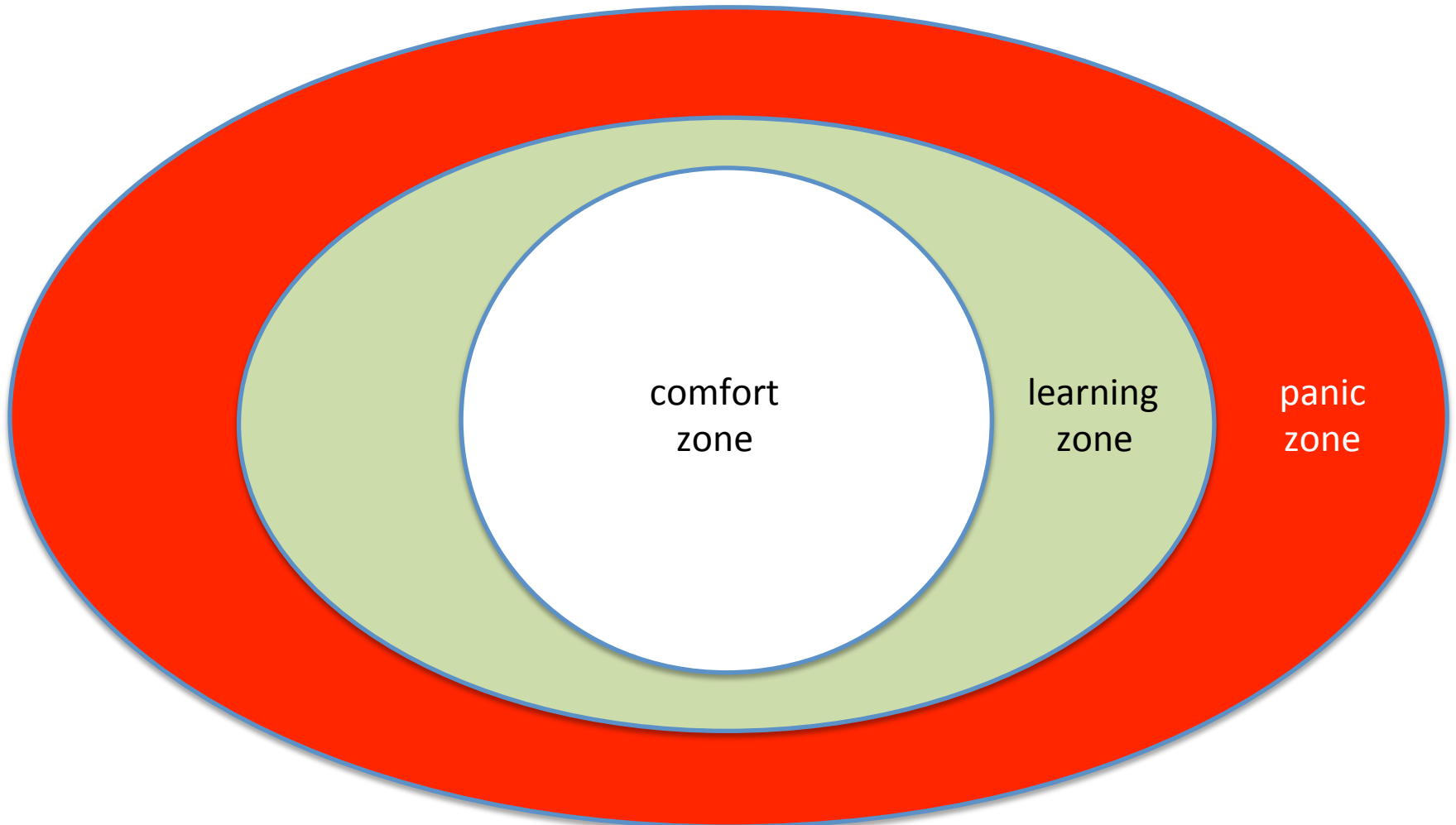




# Don't Panic



# Pacing and Panic



# Designing the Classroom

- It's about the learning, not about the teaching
- Ratio
- Pacing - Keep your students uncomfortable but don't let them panic
- Assessment – **Never** trust your students

# Not This



This



# Designing the Classroom

- It's about the learning, not about the teaching
- Ratio
- Pacing - Keep your students uncomfortable but don't let them panic
- Assessment – Never trust your students
- Flip the classroom?

# Flip the classroom?

- Definition
- The material has to be solid
- When class starts, you need a good understanding of where everyone is at

# Designing the Classroom

- It's about the learning, not about the teaching
- Ratio
- Pacing - Keep your students uncomfortable but don't let them panic
- Assessment – Never trust your students
- Flip the classroom?
- Create lesson plans



**IT'S ABOUT THE PEOPLE, MAN**

# You Can Grow Up And Be An Astronaut If You Want To!

The hardest question is a per-student question:  
will it work for that student?

# Student Factors

- Most (all?) people can learn it but not all can learn it at an intensive pace
- Things that get in the way: illness, death in the family, breakups, finances, past education, lack of stress management, learning disabilities
- TIY's model required two big risks for the students: they had to quit their job and pay a lot of money
- Having skin in the game is very important
- Woe to you if you have a very heterogenous group

# Two Primary Predictors of Success

- Hard worker
- Did lots of pre-work

# Learning and Getting a Job

- Learning the material is a prerequisite for getting a job
- When you are done, students are not at the same level and some are able to take more difficult jobs than others
- Learning is a prerequisite for work but not a guarantee

# Want to Hire a Junior Developer?

- Don't judge any coding school on a single student.
- Don't expect too much.
  - They probably haven't been coding very long at all.
  - If you are hiring them to take over important bits of your company's code solo, you are an idiot and deserve what you get.

# Want to Hire a Junior Developer? (2)

- Don't judge them on the tech you are comfortable with. Judge them on what they are comfortable with.
- I would focus on 3 things: ability to learn, general problem solving skills, and personality.
- When you interview, please, please, please give them feedback.

# The Hardest Things about Teaching (for Me)

- It was emotionally very draining
- Writing curriculum is the hardest thing I've ever done



**TWO VERY USEFUL BOOKS**

# How to Teach ADULTS

---

GET A JOB.  
PLAN YOUR CLASS.  
TEACH YOUR STUDENTS.  
CHANGE THE WORLD.

---

DAN SPALDING



