They Don’t Make ‘em Like They Used To

Integrating Junior Developers into Your Team

April 26, 2019
We don’t want Redshirts

“Captain I see something!”
Who Am I?

- Robin Clower
- Career Path
  - High School Math Teacher
  - Junior Developer
  - Productive Team Member

- Workstate Consulting
  - Drupal, Liferay, DevOps
Outline

- All Aboard the USS Enterprise – Onboarding Best Practices
- Set Phasers to Stun – Replicable Technology Set-Up
- I’m a Doctor, not a Teacher! – Effective Technical Teaching Methods
- Live Long and Prosper – Integration and Development Opportunities
All Aboard the USS Enterprise
Onboarding Best Practices
Make Day 1 Count

- Have a computer available
  - Allows time to customize computer preferences and learn company-specific tools
- Plan a Team Meeting / Lunch
  - Ask remote members to come in when possible
- Assign a buddy team member
  - Ideally with a similar experience level and skillset – for the little questions
- Keep it low stakes
Weeks 1 - X

Your Responsibility

- Maintain documentation on tools and installation
- Develop confidence-building mini-deliverables
- Check in on the buddy system
- Provide contacts – they’re just as important as answers
- Communicate a flexible (but defined) timeline

Junior Developer Responsibility

- Technology set-up
- Document pitfalls for future onboarding
- Prepare Mini Projects / Presentations
- Rely on the buddy when embarrassing issues come up
- Reach out to a variety of sources on the team for help when questions come up
- Meet deadlines or communicate in advance if a delay comes up
Structure is key

- Maintain a technical onboarding document
  - Help your junior developer help themselves
- Communicate expectations clearly
- Set goals - short and long
Self Reflection

ACTIVITY (get out some paper or a phone):
- 4 minutes
- One positive, one negative onboarding experience you’ve had
- One positive, one negative about your team’s most recent onboarding
- Talk to the person next to you
Set Phasers to Stun
Replicable Technology Set-Up
What tech does your Junior Dev need?

ACTIVITY:

▪ 2 minutes
▪ Write all tech (hardware, software, languages) you use
▪ Think through entire day
▪ Include version number if important
Me:

- Linux (Ubuntu)
- Bash
- Vi/Vim/Nano
- Yarn
- Gulp
- Composer
- Drush
- Slack
- PHP 7.0
- PHPStorm
- Xdebug
- Codesniffer
- Apache2
- MySQL
- MySQL Workbench
- Synaptic Package Manager
- Chrome
- Page Ruler
- OpenVPN
- AMP Validator
- Siteimprove
- Drupal 8
- SCSS
- Javascript
- Jquery
- Zoom
- HTML
- Twig
Assess what technology you use

- Categorize your list
  - **NI** - Not Important
  - **I** - Installed / Intuitive
    - Shouldn’t need to teach
    - Examples: Slack / Atom / Chrome
  - **U** - Understand
    - Will need to teach
    - Examples: Bash / npm / Node.js
Me:

- Codesniffer - **NI**
- Page Ruler - **NI**
- Synaptic Package Manager - **NI**
- AMP Validator - **NI**
- Siteimprove - **NI**
- Xdebug - **NI**
- Apache2 - **I**
- MySQL - **I**
- MySQL Workbench - **I**
- PHPStorm - **I**
- PHP 7.0 - **I**
- Linux (Ubuntu) - **I**
- Slack - **I**
- Chrome - **I**
- Zoom - **I**
- Composer - **U**
- Vi/Vim/Nano - **U**
- Drush - **U**
- Bash - **U**
- OpenVPN - **U**
- Yarn - **U**
- Twig - **U**
- SCSS - **U**
- Javascript - **U**
- Jquery - **U**
- HTML - **U**
- Drupal 8 - **U**
- Gulp - **U**
Homework

- Make your list a living document
- Share with team members (team drive) and ask for their additions
- Organize based on logical steps / importance
- Add time estimates
- Find resources
- List pitfalls
- Share with your new Junior Developers!
Git is hard: screwing up is easy, and figuring out how to fix your mistakes is impossible. Git documentation has this chicken and egg problem where you can't search for how to get yourself out of a mess, unless you *already know the name of the thing you need to know about* in order to fix your problem.

So here are some bad situations I've gotten myself into, and how I eventually got myself out of them *in plain english*.

**Oh s**\(^\dagger\) I did something terribly wrong, please tell me git has a magic time machine!?!?

```bash
  git reflog
  # you will see a list of every thing you've done in git, across all branches
  # each one has an index HEAD@{index}
  # find the one before you broke everything
  git reset HEAD@{index}
  # magic time machine
```

You can use this to get back stuff you accidentally deleted, or just to remove some stuff you tried
Homework

- Make your list a living document
- Share with team members (team drive) and ask for their additions
- Organize based on importance / logical steps
- Add time estimates
- Find resources
- List pitfalls
- Share with your new Junior Developers!
I’m a Doctor, not a Teacher!
Effective Technical Teaching Methods
Effective Teaching Methods

- Backwards design in the forefront
- Differentiation for each developer
- Scaffolding to help build developers’ confidence
- Keep Gardner’s Theory of Multiple Intelligences in mind
Just Kidding

- Especially in coding, vocab & jargon matter
- Coding is like a foreign language
- Meet your Junior Developer where they are
The Tree Model of Learning

- **Roots** - things junior dev should know
  - How to read, etc.
- **Trunk** - solid base of knowledge
  - HTML, CSS, PHP
- **Branches** - more specific knowledge
  - SCSS, Drupal
- **Twigs** - real nitty gritty
  - Syntax, jargon
- **Leaves** - visual demonstration of skill
  - Useable end product

- **Can’t have leaves without a solid trunk**
Assessing your Junior Developer’s Baseline

- 90% of teaching is asking questions
- Broad > Narrow questions
- Open ended but leading questions
Answering Junior Developer Questions

- Prepare for common questions
- Avoid tangents
- Don’t talk down
- Ask questions back
- Check in often
- No stupid questions
Live Long and Prosper
Integration and Development Opportunities
Becoming a better team member

- Lose the ‘Two Miles to School Uphill Both Ways’ Mentality
- Help prepare resources
- Find *somewhere* online that explains git well
- Be patient
- Stand up for your people
Help Avoid Junior Developer Pitfalls

- Ask about real weaknesses
  - Procrastination
  - Discomfort asking questions
  - Can’t handle pressure
- Overcome overreliance on internet (stack overflow, copy/paste)
- Provide specific expectations/assignments
Code Review: the most effective teacher

- Goal isn’t to merge code, it’s merge good code
- Overall review should follow the tree model
  - Start with big picture code problems
  - After those are fixed, smaller improvements
  - Lastly, refactoring, syntax, spacing
- Avoid personal attacks
- Comments should be constructive, ask open ended questions
- Be empathetic
Summary

- Prepare in advance
  - Hardware
  - Onboarding Materials
  - Assignments
- Think of a skill tree
- Ask questions
- Be patient