A Simple Overview:
Uses & Best Practices

USES FOR GITHUB

- Posting/cataloguing projects
- Getting job interviews
- Contributing to Open Source
- Learning a new language
- Expanding personal visibility/influence

WHAT EMPLOYERS LOOK FOR ON YOUR GITHUB ACCOUNT

- Self-initiated projects
- Contributions to Open Source projects
- Evidence of your architecture, design, and overall coding abilities
- The passion demonstrated by contributing to projects outside your day-to-day work
- Breadth of types of projects, use of languages/frameworks/libraries, and techniques used
- Demonstration of patterns and use of libraries and frameworks that may be discussed in interview
- Use of patches to squash sets of commits into easily-consumable packages

PROTIP: *Don't leave a bunch of cloned/forked projects on your profile you haven't contributed to! Just creates noise.*

WHAT MAKES YOU UNIQUE & KEEPS YOUR GITHUB ORGANIZED

- Utilize patches to keep sets of commits compartmentalized and clean
- ORGANIZE your large projects (those with multiple repos) to keep them under control:
    - Use a Github Organization (much like Angular or jQuery do)
- Utilize branches wisely and keep them pruned and up-to-date
- Utilize tags/releases with concise descriptions
    - Make use of semver (semantic versioning (X.X.X))
- Use consistent repo names across your entire account

WAYS TO BECOME A GITHUB FLOW JEDI

- Organize your projects logically, semantically, and in a clear and concise manner
- Lower the barriers to collaboration and learning
    - Add wiki pages for extensive documentation
    - Utilize Markdown files for brief/overview documentation
    - SERIOUSLY, START CARING ABOUT DOCUMENTATION AND NEVER STOP
    - Create Github Pages websites to showcase libraries/frameworks
- Learn and use typical Git flows
    - Use a standardized branch naming scheme

COMMIT BEST PRACTICES

- Commit often; commit early
- Use concise, accurate, readable comments
- Name your branches and stick with the scheme across repos
  - feature branches, develop, release, master, hotfix - check out Git Flow
- Rebase-then-merge vs. rebase-then-rebase vs. merge-then-merge: decide on an approach and stick with it
- Use a proper Pull Request/code review process and be vigilant about maintaining it
- Don’t forget to delete branches once merged

BEST PRACTICES FOR SASS (AND OTHER PRECOMPILED FILES)

1. Edit the .scss file

2. Commit JUST the .scss file with a concise, accurate message

3. Commit the .css file in a separate commit; depending on whether or not everyone on your team has project build tools, you may not check the compiled .css in at all

REMEMBER: *Don't edit the compiled .css files! Only edit source files.*

PROTIP: *Build tools are a great opportunity to unify a team's machine imaging and installation process to ensure everyone has the same tools and is on the same page.*

GITHUB RELEASES & RELEASE CYCLE STAGES

- Use Milestones with some sort of incremental naming scheme (dates, sprint numbers, Github Issue ranges)

- Associate Issues with specific Milestones to track completion

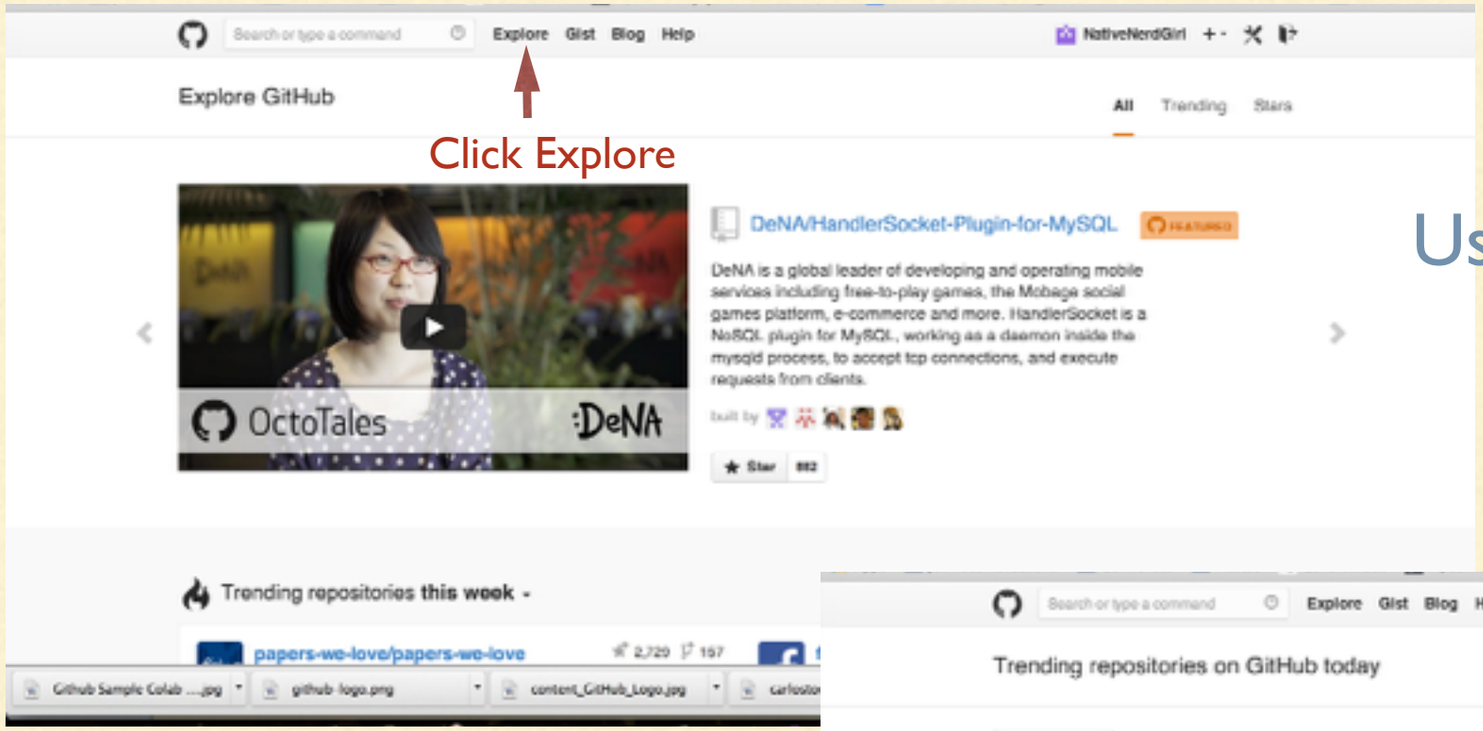
- Associate tagged releases with specific Milestones

- The release cycle generally looks like: development -> internal QA (alpha) -> client review (beta) -> prod release

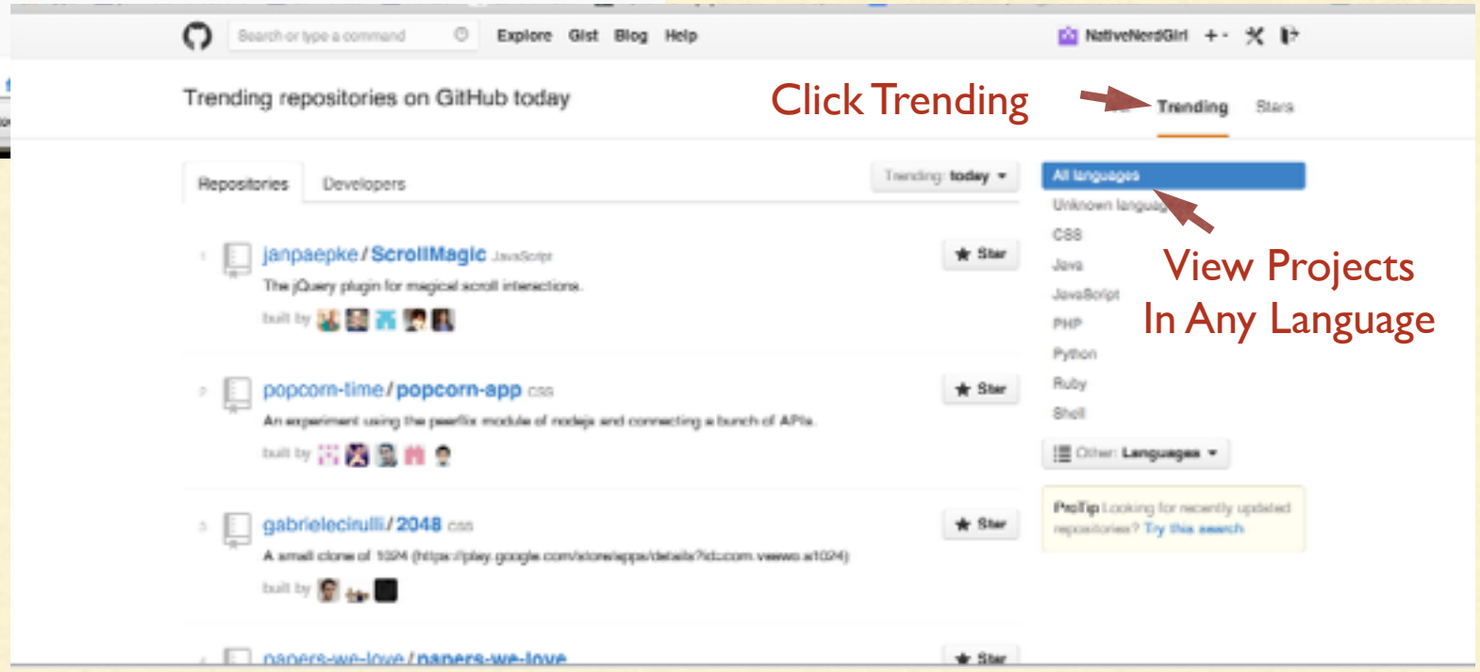
REMINDERS

- Branches should have descriptive names
- Development only happens on feature/hotfix branches
- Utilize Git Flow or another scheme for establishing naming conventions and general workflow
- Tag releases with well-annotated notes
- Commit early and often with good commit messages
- DON’T FORGET ABOUT DOCUMENTATION!

DO YOU NEED TO LEARN NEW CODE?



Use Trending repos to help learn a new language



REFERENCE MATERIALS FOR YOU

- **Pull Requests & Good Practices:**

http://codeinthehole.com/writing/pull-requests-and-other-good-practices-for-teams-using-github/

http://gitready.com/

http://git-scm.com/book

- **Another Tool For Open Source Contributing**

http://www.ohloh.net/

- **Open Source Rails Contribution guide**

https://github.com/railsjedi/opensourcerails

http://guides.rubyonrails.org/contributing_to_ruby_on_rails.html

- http://readwrite.com/2013/11/18/github-tom-preston-warner#awesm=~oy5LCDewTdx442

- https://github.com/bbatsov/rubocop

- http://www.slideshare.net/ZeroTurnaround/brent-beerjordanmcculloughlevelupyourgitandgithubexperience-24613144

- https://teamtreehouse.com/forum/choosing-a-github-username

- **Quick Guide/Refresher for Git**

https://github.com/github/hub

- **Ruby Style Guides**

https://github.com/bbatsov/ruby-style-guide (more in depth)

https://github.com/styleguide/ruby (quicker overview)

- **Hash Look Up Guide & Tricks**

https://blog.engineyard.com/2013/hash-lookup-in-ruby-why-is-it-so-fast