PR Review

Tools

A better way to do a code-review of a PR is to do it in your IDE. Here are two scripts which allow you to perform the review and create local changes which can be appended to the PR.

1. Loading PR

Run this command to load the changes into your local repository where your IDE is running.

```
$ ./scripts/github/review-pr 24623
```

This will result in output:

```
Already on 'master'
Your branch is up to date with 'origin/master'.
Fetching pull request #24623 with 1 SHA(s) into branch range:
pr/24623_base..pr/24623_top
================================================================================
cef93a51b (pr/24623_top) ci: scripts to review PRs locally
================================================================================
Switched to a new branch 'pr/24623'
On branch pr/24623
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        docs/PR_REVIEW.md
        scripts/github/push-pr
        scripts/github/review-pr

nothing added to commit but untracked files present (use "git add" to track)
```

Note that the script created `pr/24623_top` and `pr/24623_base` branches which denote SHAs where the PR start and end.

```
cef93a51b (pr/24623_top) ci: scripts to review PRs locally
637805a0c (pr/24623_base) docs: update `lowercase` pipe example in "AngularJS to
Angular" guide (#24588)
```

Knowing `pr/24623_top` and `pr/24623_base` makes it convenient to refer to different SHAs in PR when rebasing or resetting.

2. Review PR

Because the script has reset the `HEAD` of the PR the changes show up as unstaged files.

```
$ git status
On branch pr/24623
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

```
        docs/PR_REVIEW.md
        scripts/github/push-pr
        scripts/github/review-pr

nothing added to commit but untracked files present (use "git add" to track)
```

Use your IDE to review the untracked files as needed. A good trick is to use your IDE to stage the files which were already reviewed. When all files are staged the review is done.

3. Creating Edits

At any point you can edit any line in the repository. The idea is to create edits locally and push them to the PR later. This is useful because it is often times easier to make minor changes locally than to request the PR author to change and repush through a comment (often times the comment is larger than the change.)

Example of a local edit.

```
echo "# here is a change" >> docs/PR_REVIEW.md
```

4. Creating a Commit From Local Edits

Since the HEAD has been reset to `pr/24623_base` so that changes show up in `git status` we have to reverse the reset to only see our local changes. To do that reset the `HEAD` to `pr/24623_top`.

```
$ git reset pr/24623_top
```

Doing so will remove all PR changes and only leave your local modifications which you have done. You can verify by running `git status` and `git diff` to see only your changes (PR changes have been removed.)

```
$ git status
On branch pr/24623
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   docs/PR_REVIEW.md

no changes added to commit (use "git add" and/or "git commit -a")
```

```
$ git diff
diff --git a/docs/PR_REVIEW.md b/docs/PR_REVIEW.md
index 184b5aeca..83517fbe0 100644
--- a/docs/PR_REVIEW.md
+++ b/docs/PR_REVIEW.md
@@ -8,4 +8,4 @@ A better way to do code review of the PR is to do it in your IDE. Here
are two s
existing text
-
\ No newline at end of file
+# here is a change
```

Next step is to turn your local changes into a `fixup!` commit. Run `git commit --all --fixup HEAD` to create a `fixup!` commit.

NOTE: If you added new files they must be added using `git add .` or they will not be picked up by the `git commit --all` flag.

```
$ git commit --all --fixup HEAD
[pr/24623 45ae87ce4] fixup! ci: scripts to review PRs locally
 1 file changed, 1 insertion(+), 1 deletion(-)
```

You can verify that the `fixup!` commit with your local modifications was created.

```
$ git log --oneline
45ae87ce4 (HEAD -> pr/24623) fixup! ci: scripts to review PRs locally
cef93a51b (pr/24623_top) ci: scripts to review PRs locally
```

5. Pushing local edits back to the PR

The last step is to push your local changes back into the PR. Use `./scripts/github/push-pr` script for that.

```
$ ./scripts/github/push-pr
Assuming PR #24623
>>> git push  git@github.com:mhevery/angular.git HEAD:review_pr_script
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 392 bytes | 392.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To github.com:mhevery/angular.git
   cef93a51b..45ae87ce4  HEAD -> review_pr_script
```

NOTE: Notice that we did not have to specify the PR number since the script can guess it from the branch name.

If you visit https://github.com/angular/angular/pull/24623/commits you will see that your `fixup!` commit has been added to the PR. This greatly simplifies the work for many minor changes to the PR.