Triage new issues/PRs on github

This document shows the steps the AngularJS team is using to triage issues. The labels are used later on for planning releases.

Automatic processing

We have tools (e.g. Mary Poppins) that automatically add comments and labels to issues and PRs. The following is done automatically so you don't have to worry about it:

- Label `cla: yes` or `cla: no` for pull requests
- Label `GH: *`
  - `PR` - issue is a PR
  - `issue` - otherwise

Triaging Process

This process based on the idea of minimizing user pain from this blog post.

1. Open the list of non triaged issues

   - Sort by submit date, with the newest issues first
   - You don't have to do issues in order; feel free to pick and choose issues as you please.
   - You can triage older issues as well
   - Triage to your heart's content

2. Assign yourself: Pick an issue that is not assigned to anyone and assign it to you

3. Understandable? - verify if the description of the request is clear.

   - If not, close it according to the instructions below and go to the last step.

4. Duplicate?

   - If you've seen this issue before close it, and go to the last step.
   - Check if there are comments that link to a dupe. If so verify that this is indeed a dupe, close it, and go to the last step.

5. Bugs:

   - Label `Type: Bug`
   - Reproducible? - Steps to reproduce the bug are clear. If they are not, ask for a clarification. If there's no reply after a week, close it.
   - Reproducible on master? - http://code.angularjs.org/snapshot/

6. Non bugs:

   - Label `Type: Feature`, `Type: Chore`, or `Type: Perf`
   - Belongs in core? – Often new features should be implemented as a third-party module rather than an addition to the core. If this doesn't belong, close it, and go to the last step.
   - Label `needs: breaking change` - if needed
   - Label `needs: public api` - if the issue requires introduction of a new public API

7. Label `browser: *` - if the issue **only** affects a certain browser

8. Label `frequency: *` – How often does this issue come up? How many developers does this affect? Chose just one of the following:
    - low - obscure issue affecting a handful of developers
    - moderate - impacts a common usage pattern
    - high - impacts most or all AngularJS apps
9. Label `severity: *` - How bad is the issue? Chose just one of the following:
    - security issue
    - regression
    - memory leak
    - broken expected use - it's hard or impossible for a developer using AngularJS to accomplish something that AngularJS should be able to do
    - confusing - unexpected or inconsistent behavior; hard-to-debug
    - inconvenience - causes ugly/boilerplate code in apps
10. Label `component: *`
    - In rare cases, it's ok to have multiple components.
11. Label `PRs plz!` - These issues are good targets for PRs from the open source community. In addition to applying this label, you must:
    - Leave a comment explaining the problem and solution so someone can easily finish it.
    - Assign the issue to yourself.
    - Give feedback on PRs addressing this issue.
    - You are responsible for mentoring contributors helping with this issue.
12. Label `origin: google` for issues from Google
13. Assign a milestone:
    - Backlog - triaged fixes and features, should be the default choice
    - Current 1.x.y milestone (e.g. 1.3.0-beta-2) - regressions and urgent bugs only
14. Unassign yourself from the issue

Tips

- Label `resolution: *`
    - these tags can be used for labeling a closed issue/PR with a reason why it was closed.
    - Right now there are only a few rejection reasons, but we can add more as needed. Feel free to suggest one to a core team member. We don't use this label for issues that were fixed or PRs that were merged.

Closing an Issue or PR

We're grateful to anyone who takes the time to submit an issue, even if we ultimately decide not to act on it. Be kind and respectful as you close issues. Be sure to follow the code of conduct.

1. Always thank the person who submitted it.
2. If it's a duplicate, link to the older or more descriptive issue that supersedes the one you are closing.
3. Let them know if there's some way for them to follow-up.

    - When the issue is unclear or reproducible, note that you'll reopen it if they can clarify or provide a better example. Mention plunker or fiddle for examples. Watch your notifications and follow-up if they do provide clarification. :)
    - If appropriate, suggest implementing a feature as a third-party module.

If in doubt, ask a core team member what to do. Brian is probably the person to ask. You can mention him in the relevant thread like this: `@btford`.

Example:

> *Thanks for submitting this issue! Unfortunately, we don't think this functionality belongs in core. The good news is that you could easily implement this as a third-party module and publish it to the npm registry.*

Assigning Work

These criteria are then used to calculate a "user pain" score. Work is assigned weekly to core team members starting with the highest pain, descending down to the lowest.

```
pain = severity × frequency
```

severity:

- security issue (6)
- regression (5)
- memory leak (4)
- broken expected use (3)
- confusing (2)
- inconvenience (1)

frequency:

- low (1)
- moderate (2)
- high (3)

Note: Security issues, regressions, and memory leaks should almost always be set to `frequency: high`.

Analytics