the time has come to **Ditch your Beloved Taskrunner**

Drew Fyock

@drewfyock

Taskrunners

Your build system should empower, not impede

- Aim to alleviate complex tasks via automation
  - minification, compilation, unit testing, linting, etc
- **Grunt** – "the original"
  - config > code
  - wasn't updated for almost 2 years
  - Just hit 1.0.0 on Apr 4 – now is the perfect time to switch!
- **Gulp –** streams are fun!
  - Necessitates some degree of Streams mastery
  - Task sequencing is more difficult (to be addressed in v4.0)
- **Broccoli** – who actually uses it? ...and why?
  - Still "beta", i.e. <1.0

So what’s the problem?

- Require that tasks fit into their paradigms and configurations
- Have their own syntaxes, quirks and gotchas
- ***Add*** to code and build complexity
- Make you focus on fixing tooling rather than writing code
- **Even if accept those shortcomings, you still should NOT use these taskrunners because:**
  - Dependence upon plugin authors
  - Frustrating debugging
  - Disjointed documentation

Plugins

- Taskrunners rely on plugins that wrap a core command line tool
- Create another layer of abstraction away from the core tool
  - More potential for bad things to happen
  - Delay in support of fixes and features
  - Not all functionality of original tool is always supported

Debugging

- Is the base tool broken?
  - Possible, but less likely
- Is the plugin broken?
  - If a plugin fails, it might not pass the error from the core tool correctly
- Is my configuration broken?
  - Syntax, streams?
- Am I using incompatible versions?
  - Typically very difficult to determine, unless noted in the (lackluster) documentation

Documentation

- Core tool docs are **always** better than the associated plugin docs
- Even if plugin docs are decent, they almost always still require reference back to core tool for “advanced” features

There's gotta be a better way!

(hint: NPM Scripts)

- Already part of your process when using node
- Any command that you are already running at the command prompt can be moved into your *package.json* file
- NPM ecosystem is huge and very active
- Build process in your *package.json* file is less overhead
  - Only one file to keep updated as opposed to multiple configuration files for your build process
- Automatically adds *node_modules/.bin* to the PATH provided to scripts
  - Less global installs! (*npm install –g …)*

Let’s look at some **code**

https://github.com/fyockm/npm-scripts

Scripty

- Because no one should be shell-scripting inside a JSON file.
- https://github.com/testdouble/scripty#scripty

References

- http://blog.npmjs.org/post/127671403050/testing-and-deploying-with-ordered-npm-run-scripts
- http://substack.net/task_automation_with_npm_run
- https://css-tricks.com/why-npm-scripts/
- http://blog.keithcirkel.co.uk/why-we-should-stop-using-grunt/
- http://www.ctomczyk.pl/why-i-switched-to-only-nodejs-npm-and-stopped-using-grunt/767/
- https://medium.com/@dabit3/introduction-to-using-npm-as-a-build-tool-b41076f488b0#.aeaiqwnwo
- https://medium.freecodecamp.com/why-i-left-gulp-and-grunt-for-npm-scripts-3d6853dd22b8#.y4dfjg42n