Cheatography

Writing packages Cheat Sheet

by [deleted] via cheatography.com/29211/cs/8585/

Testing Why

Reliability:
In a web application (not a web site), there are multiple dependencies on any given package or control. This means that simply verifying a control works as intended in one spot in the application is relatively sketchy assurance that it works, especially if making changes when it is already in use.
Collaboration:
Write tests against the behaviors you want a package to have. When someone else changes the behavior, they then know they have broken something that was required for a reason.
Regression:
Something is broken, write a failing test, fix the test, it will never surprise you again.

Testing When

Anytime the public facing behavior of a package changes.
Anytime the internal workings of a package change and your code is riddled with hacks that care about them, or you have poorly written tests that care about them.
Don't get hung up on test-first coding. It would make a well-designed release much more estimable and consistent. But with a typical prototype-becomes-the-package, design as you go approach, re-writing tests before an initial public contract is finished can be a waste of time.

Testing How

Enumerate and describe all of the public behaviors of your package.
Compromises:
Actual unit testing (stubbing out all services) results in sturdy tests that don't need to be re-written when dependencies change. This results in a huge future payoff in reliability and avoiding a cycle where code changes always necessitate test changes (which is what test writing seems to be commonly perceived as).
In a time crunched situation it is possible to best-guess which services will actually change, and write integration or partial integration tests, but they should be marked as such, because it is essentially creating ongoing technical debt for whichever project they are placed in. **It should not be more than a few minutes faster to write a throwaway test vs an actual unit test because there should be no need to create plumbing for any properly designed packages (thanks to sinonjs, systemjs, and the windsor test npm package).**.

With

test-js.git gets installed into the npm test command on the yeoman template automatically. The **npm test** command looks up and runs tests in the **lib** directory. Tests are denoted as:
file.**spec.js** - Loaded with System.import, tested in node and browsers (allows es6)
file.**web.spec.js** - Loaded with System.import, tested in browsers (allows es6)
file.**node.spec.js** - Loaded with node's require(), tested in node (does not allow es6)