# How Nick Does Code Review

I'm going to be in Odessa and on PTO until the middle of October, so it doesn't make sense to block on me for moving PRs along. However, if Chris or Dylan feel like it really would be nice if I was around to look at something, then they can work through this checklist. If not, that's fine too.

Sometimes the answers to these questions are obvious, and sometimes I leave a comment asking for help from the pull requestor. Not every entry is applicable to every CR.

## Design

1. What problem is this trying to solve?
2. Why is this the right time to solve that problem?
3. Does this provably solve that problem?
4. Is this the right tradeoff of elegance of design v. pragmatism?
5. What other approaches to solving this problem did we consider?
6. Does this introduce new responsibility to a module? Is that what we want?
7. Does this add new coupling between modules? Is that what we want?
8. Does the code rely on any implementation details of its dependencies?
9. Is this a breaking change to the module's API? Is now the right time for that?
10. If there are performance optimizations, are they premature?
11. If we are adding new parameters or ways to configure something, do the defaults make sense?
12. Is all state as tightly scoped as it can be? (If there is global state, why?)

13. Are we handling various expected error cases appropriately? (eg http failures)
14. If we are using sync APIs, why?

## Communication

1. Are variables well-named?
    a. If there are units of measure, ensure that they are present in the name
    b. By convention, certain types like promises or html elements tend to have names that explicitly mention their types
2. Are all comments necessary?
3. Do we need to inform any users that this change is occurring?
4. Is there anything weird or confusing that lacks a comment?
5. If something explicitly sucks but we're doing it anyway for expediency, ensure that it has a comment explaining the situation
6. If the code may throw a user-facing error, will it make sense to the user?
7. If we are introducing a new way for the user to interact with our code, does that interaction make sense?
8. Does this change require updating the docs or adding new docs?
9. Is there additional logging that would be helpful?
10. Are existing logs at the right log level?

## "Static" Analysis

1. Is lodash being used effectively?
    a. If there's a `for` loop, what's the great reason justifying it?
2. Ensure that there are no commented-out blocks of dead code
3. Are there any common pitfalls?
4. Does the code match the style guide?
5. Is the code compliant with our generally functional style?
    a. If it's excessively imperative, what's the great reason justifying that?
    b. Where possible, ensure that the output of a method is a pure function of its input
    c. Recursion is often cleaner than iteration
6. If `this` is being used, why?
7. Are we using const as much as possible?
8. Ensure that var is never used
9. Ensure that no PII is logged
10. If we are taking input from users, ensure that it is sanitized
11. Ensure that methods from third party code are bound where appropriate.
    a. Unsafe:
        i. var x = thirdPartyModule.x;
    b. Safe;
        i. var x = thirdPartyModule.x.bind(thirdPartyModule);

## Tests

1. Do they exist?
2. If you break the code that is being tested, do the tests fail as you would expect?

3. Are we testing the right thing? Or are we just testing an implementation detail?
4. Can you manually see that it works?