January 8th, 2015 by Gareth Wilson (2 min read)

Stop More Bugs with our Code Review Checklist

In our blog about effective code reviews, we recommended the use of a checklist. Checklists are a great tool in code reviews – they ensure that reviews are consistently performed throughout your team. They're also a handy way to ensure that common issues are identified and resolved.

Research by the Software Engineering Institute suggests that programmers make 15-20 common mistakes. So by adding such mistakes to a checklist, you can make sure that you spot them whenever they occur and help drive them out over time.

To get you started with a checklist, here's a list of typical items:

Code Review Checklist

General

- Does the code work? Does it perform its intended function, the logic is correct etc.
- Is all the code easily understood?
- Does it conform to your agreed coding conventions? These will usually cover location of braces, variable and function names, line length, indentations, formatting, and comments.
- Is there any redundant or duplicate code?
- Is the code as modular as possible?
- Can any global variables be replaced?
- Is there any commented out code?
- Do loops have a set length and correct termination conditions?
- Can any of the code be replaced with library functions?
- Can any logging or debugging code be removed?

Security

- Are all data inputs checked (for the correct type, length, format, and range) and encoded?
- Where third-party utilities are used, are returning errors being caught?
- Are output values checked and encoded?
- Are invalid parameter values handled?

Documentation

- Do comments exist and describe the intent of the code?
- Are all functions commented?
- Is any unusual behavior or edge-case handling described?
- Is the use and function of third-party libraries documented?

- Are data structures and units of measurement explained?
- Is there any incomplete code? If so, should it be removed or flagged with a suitable marker like 'TODO'?

Testing

- Is the code testable? i.e. don't add too many or hide dependencies, unable to initialize objects, test frameworks can use methods etc.
- Do tests exist and are they comprehensive? i.e. has at least your agreed on code coverage.
- Do unit tests actually test that the code is performing the intended functionality?
- Are arrays checked for 'out-of-bound' errors?
- Could any test code be replaced with the use of an existing API?

You'll also want to add to this checklist any language-specific issues that can cause problems.

The checklist is deliberately not exhaustive of all issues that can arise. You don't want a checklist, which is so long no-one ever uses it. It's better to just cover the common issues.

Optimize Your Checklist

Using the checklist as a starting point, you should optimize it for your specific use-case. A great way to do this is to get your team to note the issues that arise during code reviews for a short time. With this data, you'll be able to identify your team's common mistakes, which you can then build into a custom checklist. Be sure to remove any items that don't come up (you may wish to keep rarely occurring, yet critical items such as security related issues).

Get Buy-in and Keep It Up To Date

As a general rule, any items on the checklist should be specific and, if possible, something you can make a binary decision about. This helps to avoid inconsistency in judgments. It is also a good idea to share the list with your team and get their agreement on its content. Make sure to review the checklist periodically too, to check that each item is still relevant.

Armed with a great checklist, you can raise the number of defects you detect during code reviews. This will help you to drive up coding standards and avoid inconsistent code review quality.

To learn more about code reviews at Fog Creek, check out the following video: