# Code Coverage

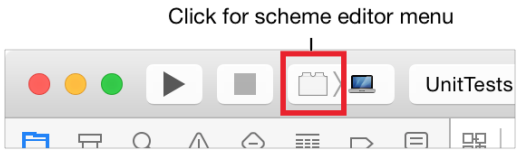
Code coverage is a feature in Xcode 7 that enables you to visualize and measure how much of your code is being exercised by tests. With code coverage, you can determine whether your tests are doing the job you intended.

## Enable Code Coverage

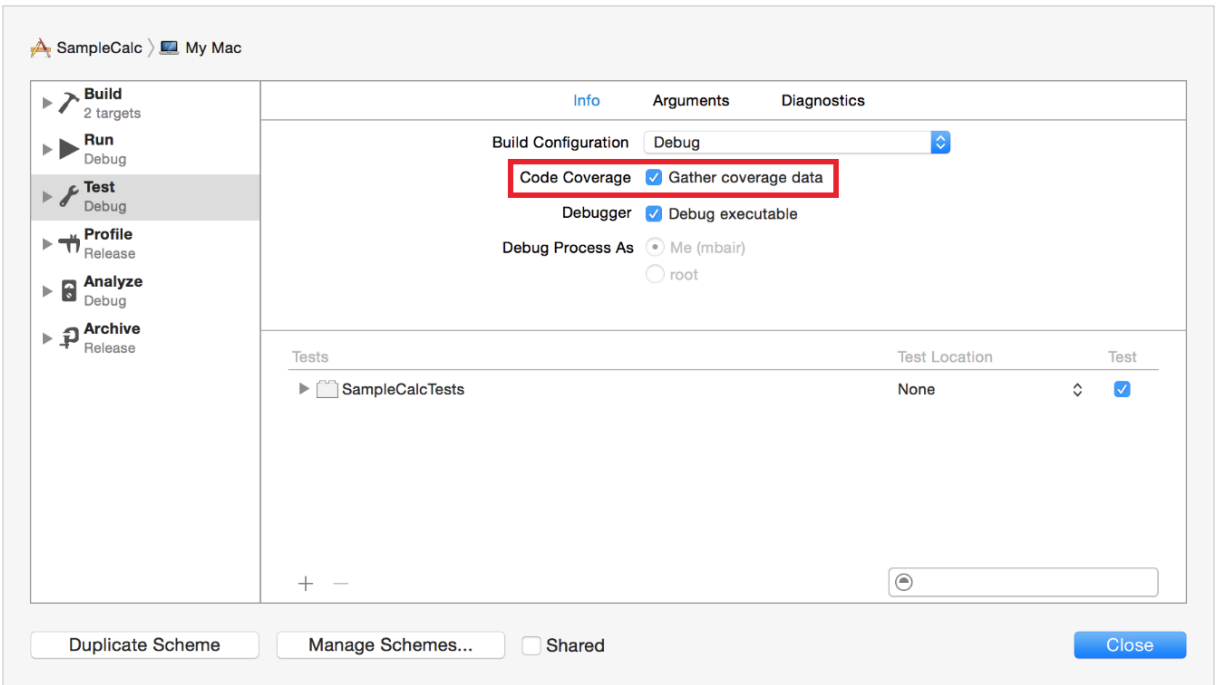
Code coverage in Xcode is a testing option supported by LLVM. When you enable code coverage, LLVM instruments the code to gather coverage data based on the frequency that methods and functions are called. The code coverage option can collect data to report on tests of correctness and of performance, whether unit tests or UI tests.

You enable code coverage by editing the scheme's Test action.

1. Select Edit Scheme from the scheme editor menu.




2. Select the Test action.
3. Enable the Code Coverage checkbox to gather coverage data.




4. Click Close.

**Note:** Code coverage data collection incurs a performance penalty. Whether the penalty is significant or not, it should affect execution of the code in a linear fashion so performance results remain comparable from test run to test run when it is enabled. However, you should consider whether to have code coverage enabled when you are critically evaluating the performance of routines in your tests.

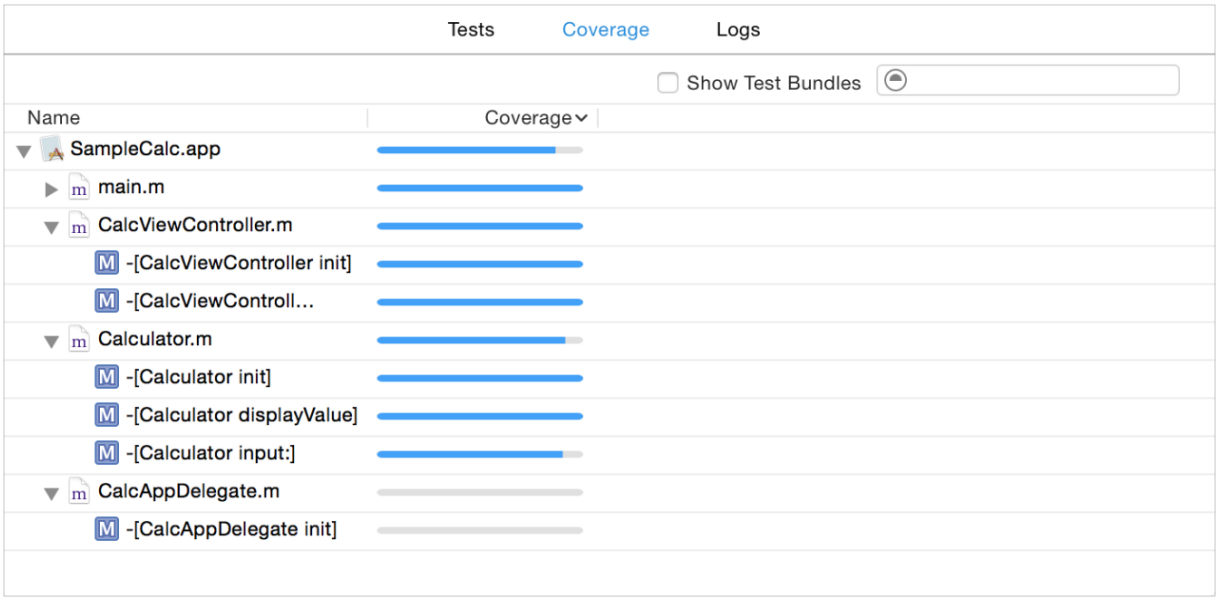
# How Code Coverage Fits into Testing

Code coverage is a tool to measure the value of your tests. It answers the questions

- What code is actually being run when you run your tests?
- How many tests are enough tests?

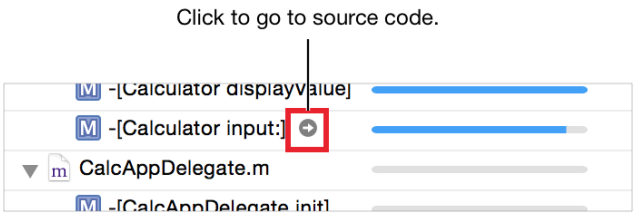
  In other words, have you architected enough tests to ensure that all of your code is being checked for correctness and performance?
- What parts of your code are not being tested?

After a test run is completed, Xcode takes the LLVM coverage data and uses it to create a coverage report in the Reports navigator, seen in the Coverage pane. It shows summary information about the test run, a listing of source files and functions within the files, and coverage percentage for each.

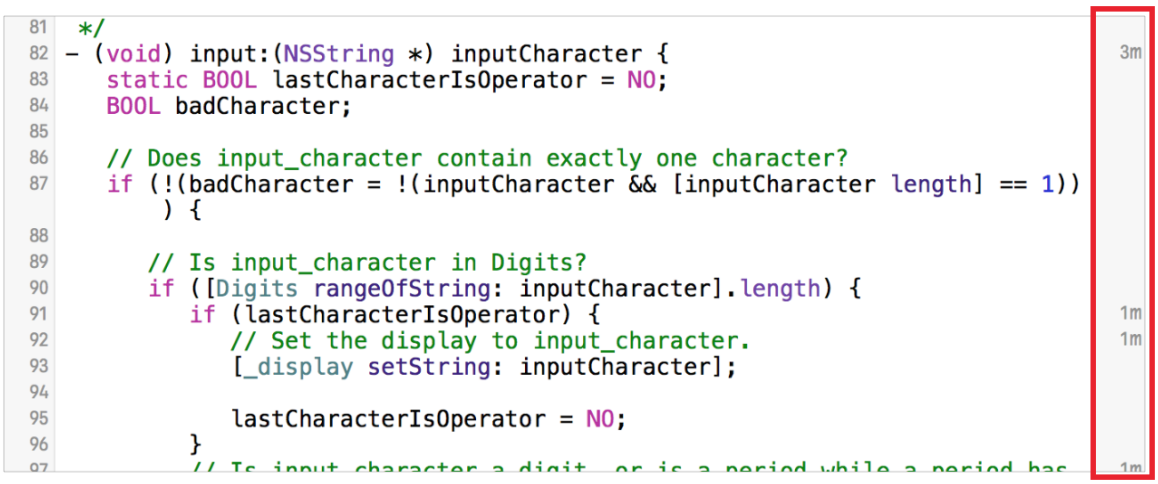



The source editor shows counts for each line of code in the file and highlights code that was not executed. It highlights areas of code that need coverage rather than areas that are already covered.

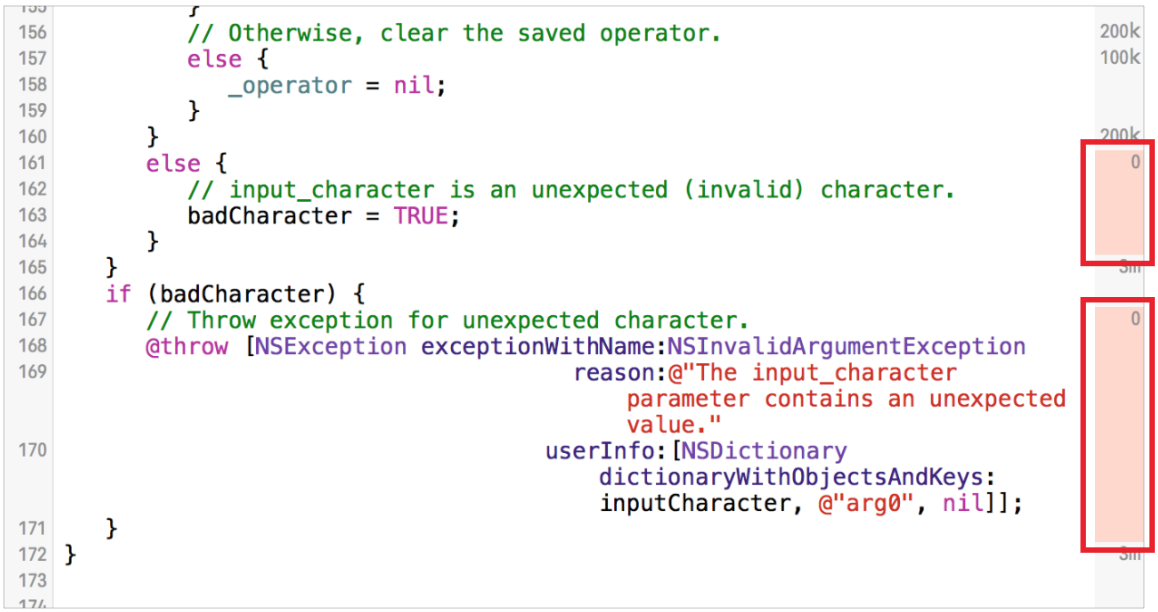
For example, holding the pointer over the `-[Calculator input:]` method in the coverage report above shows a button that will take you to the annotated source code.




The coverage annotation is drawn on the right and shows the count for how many times a particular part of the code was hit during the test. For example:




The `input:` method, by the counts above, was called frequently in the tests. However, there were sections of the method that were not called. This is clearly marked in the source editor, as below:




This report data and display suggests an opportunity to write a test that includes unexpected or invalid characters to be sure that the error handling works the way you intended.

---