

Code Coverage

YEGOR BUGAYENKO

Lecture #15 out of 24

80 minutes

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as website. Copyright belongs to their respected authors.

Example, Part I

Live Code:

```
1 int fibonacci(int n) {  
2     if (n <= 0) {  
3         return 0;  
4     }  
5     if (n <= 2) {  
6         return 1;  
7     }  
8     return fibonacci(n-1)  
9         + fibonacci(n-2);  
10 }
```

Test Code:

```
1 assert fibonacci(1) == 1;  
2 assert fibonacci(2) == 1;
```

$$C = 3/10 = 30\%$$

Example, Part I

Live Code:

```
1 int fibonacci(int n) {  
2     if (n <= 0) {  
3         return 0;  
4     }  
5     if (n <= 2) {  
6         return 1;  
7     }  
8     return fibonacci(n-1)  
9         + fibonacci(n-2);  
10 }
```

Test Code:

```
1 assert fibonacci(1) == 1;  
2 assert fibonacci(2) == 1;  
3  
4 assert fibonacci(9) == 34;  
5 assert fibonacci(10) == 55;
```

$$C = 5/10 = 50\%$$

+2pt



“Coverage numbers (like many numbers) are dangerous because they’re objective but incomplete. They too often distort sensible action. Using them in isolation is as foolish as hiring based only on GPA.”

— Brian Marick, *How to Misuse Code Coverage*, 1997

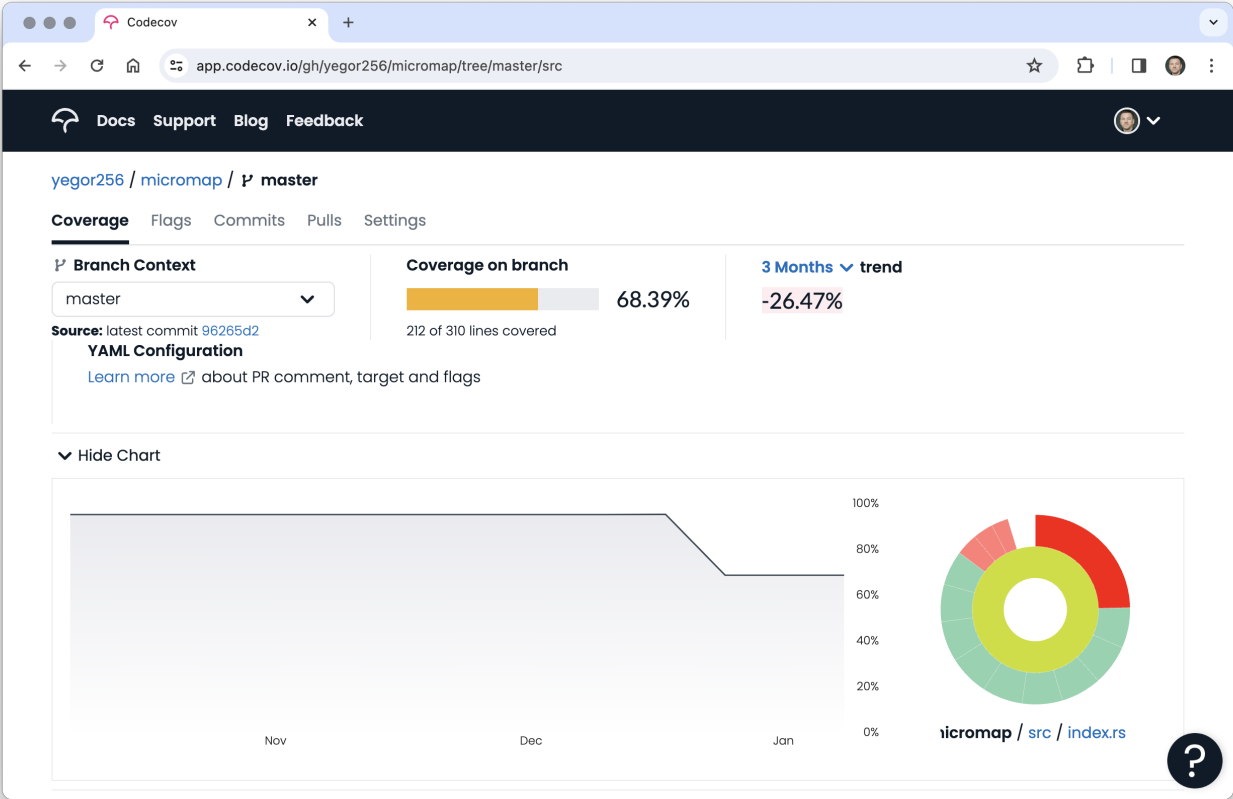
+2pt



“I would be suspicious of anything like 100% — it would smell of someone writing tests to make the coverage numbers happy, but not thinking about what they are doing.”

— Martin Fowler, *Test Coverage*, 1997

Codecov.io



Line Coverage

yegor256 / micromap / master

CoverageFlagsCommitsPullsSettings

micromap / src / iterators.rs

UncoveredPartialCovered

110#[inline]

111#[must_use]

112fn into_iter(self) -> Self::IntoIter {

113 IntoIter {

114 pos: 0,

115 map: ManuallyDrop::new(self),

116 }

117}

118}

119

120impl<K: PartialEq, V, const N: usize> Drop for IntoIter<K, V, N> {

121 fn drop(&mut self) {

122 for i in self.pos..self.map.len {

123 self.map.item_drop(i);

124 }

125 }

126}

127

128impl<'a, K, V> DoubleEndedIterator for Iter<'a, K, V> {

129 fn next_back(&mut self) -> Option<Self::Item> {

130 self.iter.next_back().map(|p| {

131 let p = unsafe { p.assume_init_ref() };

132 (&p.0, &p.1)

133 })

134 }

135}

Tarpaulin for Rust

```
Code Blame 23 lines (23 loc) · 551 Bytes Raw Copy Download Edit View Source
```

```
1  ---
2  name: tarpaulin
3  on:
4    push:
5      branches:
6        - master
7  jobs:
8    tarpaulin:
9      runs-on: ubuntu-22.04
10     steps:
11       - uses: actions/checkout@v4
12       - uses: actions-rs/toolchain@v1
13         with:
14           toolchain: stable
15           override: true
16       - uses: actions-rs/tarpaulin@v0.1
17         with:
18           version: '0.22.0'
19           args: '--all-features --exclude-files src/lib.rs -- --test-threads 1'
20       - uses: codecov/codecov-action@v3
21         with:
22           token: ${ secrets.CODECOV_TOKEN }
23           fail_ci_if_error: true
```


Code Coverage can be calculated by a few tools:

- JaCoCo for Java
- Istanbul for Javascript
- Gcov for C/C++
- Coverage.py for Python
- Simplecov for Ruby
- Tarpaulin for Rust

Read this:

,