GENERAL ASSEMBLY

TEST-DRIVEN DEVELOPMENT

Zeke Abuhoff

Lead iOS Instructor, General Assembly

LEARNING OBJECTIVES

+ Define test-driven development
+ Define code coverage
+ Define continuous integration

TDD

Test-driven development, often referred to as **TDD**, is a philosophy of software development.

It posits that writing tests after writing production code relegates the tests to an afterthought; it makes them seem unimportant.

To make your software not merely possessing tests, but rather driven by those tests, you must write the tests first. The tests start out failing, then correctly written production code makes them pass.

TDD

PRO	CON
• Tests are definitely present. • Tests actually describe desired behavior. • Expectations of the software are clear before production code is written.	• Assertions don’t start out with references to the code they’ll need. • Design of the app is influenced by the production process; not everything is clear at first. • Not every workflow prioritizes automated tests.

CODE COVERAGE

Whenever we're writing unit tests, we want the tests to be comprehensive. If some code is validated by tests while other code is not, we only know that part of the app is working as expected.

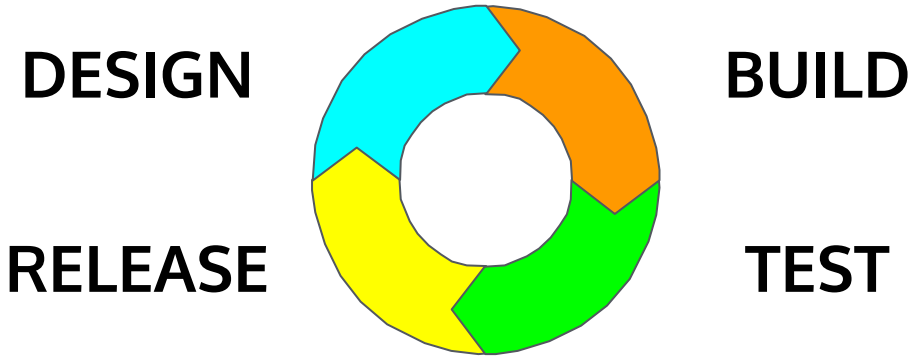
The measure of code validated by tests is called code coverage.

```
func purchase(item: Item) throws {
    if paymentInfo != nil {
        pointOfSaleClient.purchaseItem(id: item.id)
    } else {
        throw PurchaseError.NoPaymentInfo
    }
}
```

Currently tested code path

Code path missing from tests

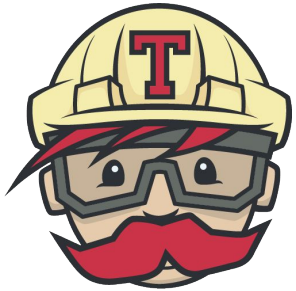
CONTINUOUS INTEGRATION



In a world where the release process is continuous, our testing should be continuous. Every change should be tested, no matter how small.

The process of maintaining fully tested code is known as **continuous integration**, or CI for short.

CONTINUOUS INTEGRATION



Different companies use different CI systems. The details of the systems vary, but the basic design remains the same:

1) Changes are deployed via git.
2) Tests run on remote machines, either virtual or real.
3) The team can check if recent changes broke tests.