Unit tests

- Software development process
- Units(smallest testable parts of application) are tested/validated for proper operation
- Component of test driven development
    - Build product by means of continuous testing and revision
- It involves testing only vital characteristics for information under test
- Earlier a problem is identified, fewer compound errors will occur

For more information:

https://searchsoftwarequality.techtarget.com/definition/unit-testing

Truffle tests:

- Test in truffle are built over ChaiJS.
- http://www.chaijs.com/api/assert/

Important points about testing solidity contracts:

1: Solidity does not support logging functions like console.log () or print

This make it harder to debug and troubleshoot. (Remember you can always use Remix)

2: Smart contract is executed by EVM so we need Ethereum node to execute contract

There are three different ways to execute your contract:

1: Main ethereum Network:

Costs money, takes around 5 mins to deploy and execute. Sometimes dangerous if contract has bugs.

2:Ethereum Testnet:

Free, but still slow and its public.

3: Ethereum Network Simulator(testrpc):

Fast, private and free.

Since we have already installed truffle. We install testprcp

$npm i -g ethereumjs-testrpc

Make directory for your project:

$mkdir voting-project

Initiate truffle

$truffle init

Go to contracts directory and create Voting.sol file:

Copy and paste the voting contract you wrote during last lab into Voting.sol file and save it.

Now you have Voting.sol contract ready.

Lets create test file for Voting.sol contract

You can do it either by:

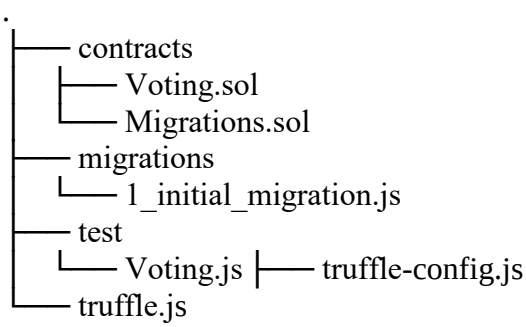
$ truffle create test Voting

Or

Manually create file and write your tests:

$ nano test/Voting

voting-project directory structure will look like:



Open test/Voting.js with your favorite editor and let's write test:

Before we start lets know about promises:

A Promise object represents completion (or failure) of an asynchronous operation, and its resulting value. In other words, a promise is returned object to which we attach callbacks, instead of passing callbacks to a function.

For more information about Promise visit : https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-promise-27fc71e77261

Voting = artifacts.require('../contracts/Voting.sol');

```
contract('Voting', function() {

  // it("should assert true", function(done) {

  //   var voting = Voting.deployed();

  //   assert.isTrue(true);

  //   done();

  // });

   const voting = Voting.deployed(['a','b']);

 it("should add users during deployment", function(){

   return voting.then(instance=>{

     //console.log(instance)

   return  instance.validCandidate.call("a").then(result=>{

   //  console.log(result);

 //expect(instance.validCandidate('a')).to.be.a(true);
```

```
        assert.isTrue(result);



      });

    });

  });

it("Should add the vote to the respective user", function(){

  return voting.then(instance1=>{

    //console.log(instance1)

    return instance1.voteForCandidate("a").then(tx=>{

    //  console.log(tx)

      return instance 1.totalVotesFor.call("a").then(r =>{

        //console.log(r)

        assert.equal(r,1,"One vote is added to the candidate");

 });
```

```
  });

});

});

});
```

After writing start testrpc in another terminal:

$testrpc

Now on the first terminal run your test:

$truffle test /test/voting.js