when you compile smart contract -> artifact/ABI.json created

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

let contract\_1 = artifacts.require(“contract\_1”);

contract("contract name" ( )=>

{

let i = null ; //i

beforeEach( async ()=>

or

before( async ()=>

{

i = await contract\_1.deployed( );

or

i = await contract\_1.new();

console.log(i.address); //contract address is diff. For Each it() statement

}

it (“testing func\_1 : should return “jatin” value”), async( ) =>

{

let a = await i.func\_1();

assert(a === “jatin”);

};

it (“testing func\_2 : should return 19 value”), async( ) =>

{

let a = await i.func\_1();

console.log(“typeof =”, typeof a) // debugging

assert(a.toNumber() === 19); // bigNumber object -> Number

// parseInt(str\_ ) use for string ->int

// web3.utils.toBN() converting -> bigNumber object

};

it (“testing set(3) : should return 3 value”), async( ) =>

{

await i.set(3);

let a = await i.get ();

assert(a.toNumber() === 3);

};

} );

**avoid using multiple assert** in 1 **it** block(you can use when solidity function return multiple value) because if you write 2 assert condition in 1 it block you will never know which assert condition fails

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**assert**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function f1() public returns(uint,uint) { return(2,3);}

⬇

it (“testing func\_1 : should return “jatin” value”), async( ) =>

{

let a,b =await i.f1();

assert(a==1,”a must be equal to 1”);

assert(b==2,”b must be equal to 2”);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Mocha Library**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

⬇

**before**(function () {

// runs once before the first test in this block

});

**after**(function () {

// runs once after the last test in this block

});

**beforeEach**(function () {

// runs before each test in this block

});

**afterEach**(function () {

// runs after each test in this block

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**msg.sender , msg.value**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

uint a;

function f1(uint i) payable returns(address,uint)

{

a=i;

return (**msg.sender,msg.value**)

}

⬇

it(“ calling f1() function ”,async()=>

{

await i.f1(19,**{from:accounts[4],value: 2 wei}**); // by default from : accounts[0]

)}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**test-helpers**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

uint a;

One\_time\_call\_only

{

require(a<=1,”you can call only one time ”)

\_;

}

function f1() public one\_time\_call\_only {a++;}

⬇

**$ npm i @openzeppelin/test-helpers**

⬇

let{expectRevert} = require(“**@openzeppelin/teest-helpers**”);

contract(“”,()=>

{

it(“ checking f1() function ”,async()=>

{

await i.f1();

await **expectRevert**(f1(),”you can call only one time ”);

// “ ” line must be same

)}

})

always use expectRevert(f1(),””) when function is using **modifier**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Ganache Account**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Connect -> Ganache**

⬇

it(“Ganache Account Printing”,async()=>

{

console.log(“**accounts**”); // **account[0],account[1]**

let b= **web3.eth.getBalance**(account[3]); // get balance of Ganache account

})

These are not pure web3 cmd -> there cmd are from truffle

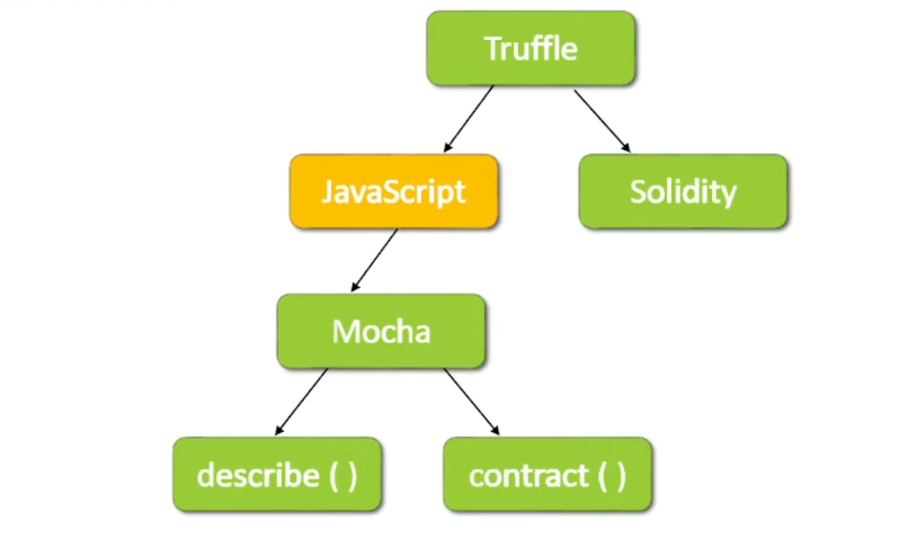
**Error**

It is not a function

**Sol**ution

Check if any statement in top is automatically imported -> then remove those statement

**$ truffle test**



**In describe() use Pure web3 cmds -> little tougher than contract() cmds**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Testing in Hardhat\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

⬇

**$ npm install --save-dev @nomiclabs/hardhat-truffle5 @nomiclabs/hardhat-web3 web3**

⬇

**In hardhat.config.js**

Add Line -> require(“@nomiclabs/hardhat-truffle5”)

⬇

**In Test.js**

let {contract} = requte(“ethers”);

Let {artifacts} = requte(“hardhat”);

let contract\_1 = artifacts.require(“contract\_1”);

contract("contract name" ( )=>

{

let i = null ; // contract instance

beforeEach( async ()=>{i = await contract\_1.deployed( )**;**}

it(“” async()=>

{

**assert.equal**(,“”)

assert.equal(,“”)

})

⬇

npx hardhat test

Resources

https://mochajs.org/

boli lagan ghar ke == bidding (in english)

auction = nelami (in hindi) = a public sale at which items are sold to the person who offers to pay the most money