

Blockchain and Smart Contracts

Dr Joshua Ellul

joshua.ellul@um.edu.mt

Department of Computer Science

University of Malta

- Download and Install NodeJS
 - <https://nodejs.org/en/download/>

Compiling

- Many different ways:
 - the online compiler:
 - <https://remix.ethereum.org>
 - Truffle (we'll see soon)
 - Using solc and web3
 - ... more

HelloWorld

```
pragma solidity ^0.4.0;

contract HelloWorld {
    string message;

    function setMessage(string msgin) public {
        message = msgin;
    }

    function getMessage() public constant returns (string) {
        return message;
    }
}
```

remix.ethereum.org

The screenshot displays the Remix IDE interface. On the left, a code editor shows a Solidity contract named `HelloWorld` in `browser/ballot.sol`. The contract includes a `setMessage` function and a `getMessage` function. The right sidebar contains a 'Compile' tab with a 'Start to compile' button and a checked 'Auto compile' checkbox. Below this, a dropdown menu shows the selected contract, and a 'Static Analysis' warning is displayed. At the bottom of the sidebar, a green box indicates the compiled contract name.

```
1 pragma solidity ^0.4.0;
2
3 contract HelloWorld {
4     string message;
5
6     function setMessage(string msgin) public {
7         message = msgin;
8     }
9
10    function getMessage() public constant returns (string) {
11        return message;
12    }
13 }
14
15
16
```

Compile Run Settings Debugger Analysis Support

Start to compile ☒ Auto compile

browser/ballot.sol:HelloWorld Details Publish on Swarm

Static Analysis raised 2 warning(s) that requires your attention. ✕

browser/ballot.sol:HelloWorld ✕

Setting up a Test Environment

- Easiest way to test ethereum and solidity:
 - EthereumJS
 - Runs on NodeJS
 - Install:
 - `npm install -g ethereumjs-testrpc`
 - (sudo?)
 - Run testrpc:
 - `$ testrpc`

Writing your First Contract and Deploying using Truffle

- Install truffle:
 - `npm install -g truffle`
 - (sudo?)
- Create project:
 - `mkdir test_contract`
`cd test_contract`
`truffle init`

Writing your First Contract and Deploying using Truffle (cont)

- Create the contract file, in the **contracts directory** HelloWorld.sol:

```
pragma solidity ^0.4.0;

contract HelloWorld {
    string message;

    function setMessage(string msgin) public {
        message = msgin;
    }

    function getMessage() public constant returns (string) {
        return message;
    }
}
```


Writing your First Contract and Deploying using Truffle (cont)

- Create migrations/2_deploy_contracts.js and modify to look like:

```
var HelloWorld = artifacts.require("./HelloWorld.sol");
module.exports = function(deployer) {
  deployer.deploy(HelloWorld);
};
```

Writing your First Contract and Deploying using Truffle (cont)

- Configure truffle.js to connect to localhost:

```
module.exports = {  
  networks: {  
    development: {  
      host: "localhost",  
      port: 8545,  
      network_id: "*" // Match any network id  
    }  
  }  
};
```

Writing your First Contract and Deploying using Truffle (cont)

- Ensure testrpc is running
- Compile, HelloWorld and then deploy:
truffle compile
truffle migrate

```
Joshuas-MacBook-Air:test_contract joshuaellul$ truffle migrate
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x0511876fbd65e88fa51ee69f5de53942a189eb03aebaf3b8927d4a2a9ccdf321
  Migrations: 0xeb807074bf066ff95b2f8a61b88c9bb2732fd0ad
  Saving successful migration to network...
  ... 0xa98056f37aa1c0263ef93c091455da8fb21668866e04c1858e08cdf420070c5c
[Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying HelloWorld...
  ... 0x3a3420a3c402ce535850438e953006fe43fb0a52400cf4e0ef31527b093b87c9
  HelloWorld: 0xe1e494d6a86fb02c4b976e08455d60013701a281
  Saving successful migration to network...
  ... 0xe3e6e5a38cfc6fc74dd6508384eb6368787061f08acca1c47047eeaf0a74f594
  Saving artifacts...
```

Writing your First Contract and Deploying using Truffle (cont)

- testrpc output (contract created?)

```
Transaction: 0x0511876fbd65e88fa51ee69f5de53942a189eb03aebaf3b8927d4a2a9ccdf321
Contract created: 0xeb807074bf066ff95b2f8a61b88c9bb2732fd0ad
Gas usage: 269543
Block Number: 1
```

Writing your First Contract and Deploying using Truffle (cont)

- Test the contract:
truffle console

- Call getMessage:

```
[truffle(development)> HelloWorld.deployed().then(instance => instance.getMessage.call())
```

- Set the message:

```
[truffle(development)> HelloWorld.deployed().then(instance => instance.setMessage.sendTransaction('hello world'))  
'0x6080086ecf31c4e53de1c82ddce56386ba2e8c8eefc3dcc9becf8586e7cf4e33'
```

- Check again:

```
[truffle(development)> HelloWorld.deployed().then(instance => instance.getMessage.call())  
'hello world'  
_
```

Writing Blockchain Edge Code: web3

- web3 is a JavaScript library for communicating with Ethereum nodes
 - I.e. this is off-blockchain code
- Installing web3 in your project directory:
 - `npm install web3@0.20.1`

Using web3

- Getting web3 instance:

```
var Web3 = require('web3');  
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));
```

- Check the web3 API (also check to see if anything has changed):

```
console.log(web3.version.api);
```

Using web3 (cont)

- Execute a method on your contract

```
var Web3 = require('web3');
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

//useful to check if something does not work
console.log(web3.version.api);

var contract = "0xe1e494d6a86fb02c4b976e08455d60013701a281";

var abi = [{"constant":false,"inputs":[{"name":"msgin","type":"string"}],"name":"setMessage","output":
var MyContract = web3.eth.contract(abi);
var myContractInstance = MyContract.at(contract);

console.log( myContractInstance.getMessage() );
```

Run using node:
node <YourJSFile>

Find your contract address

Find your ABI

Finding your Contract Address

```
Deploying Migrations...
... 0x394ec166f490f8232b318d1a320e86a6599a17c9419102f3cf912abb6e1bc03b
Migrations: 0x63dd6442649104eaa59b809c16095f3b3a960082
Saving successful migration to network...
... 0x1f74085065e30213d0a62a310692d4167a8e60a71764086e3b57a84dbe08e4a7
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying HelloWorld...
  ... 0x2df1835f440b9b2c0a0ee6cf5432c3d553685b344675b10a4685de3ea4dba4f7
  HelloWorld: 0x05a2ac304736d178b382b34d6f7e280c31f04c41
Saving successful migration to network...
... 0x64a7bdf32526dc21a51ba0c0284c74b01d29f1dbe2f71e1fbee905fc6bea811d
Saving artifacts...
Joshuas-MacBook-Air:test_contract joshuaellul$
```

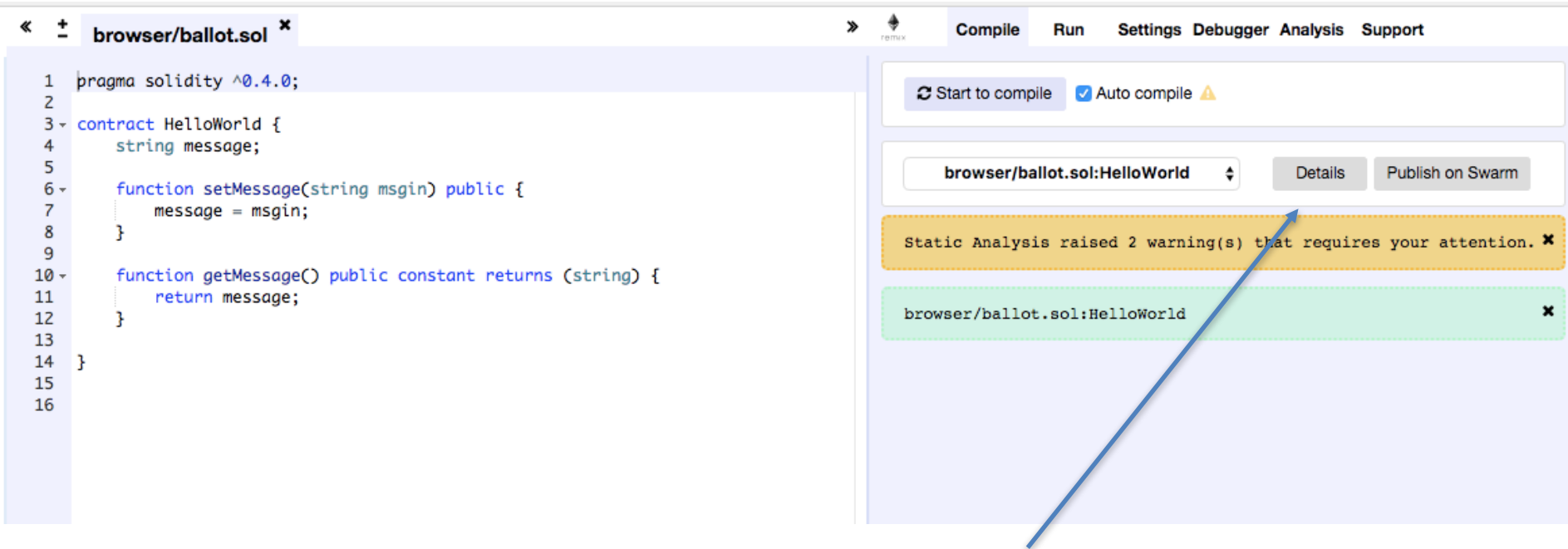


This is it

Finding your ABI

In a browser go to: remix.ethereum.org

Paste your code:



The screenshot shows the Remix IDE interface. On the left, a code editor displays Solidity code for a contract named 'HelloWorld'. The code includes a pragma statement for Solidity version ^0.4.0, a string variable 'message', a 'setMessage' function, and a 'getMessage' function. On the right, the 'Compile' tab is active. It shows a dropdown menu with 'browser/ballot.sol:HelloWorld' selected, and buttons for 'Details' and 'Publish on Swarm'. Below the dropdown, a yellow warning box states 'Static Analysis raised 2 warning(s) that requires your attention.' and a green box shows the contract name 'browser/ballot.sol:HelloWorld'. A blue arrow points from the text 'Press details' to the 'Details' button.

```
1 pragma solidity ^0.4.0;
2
3 contract HelloWorld {
4     string message;
5
6     function setMessage(string msgin) public {
7         message = msgin;
8     }
9
10    function getMessage() public constant returns (string) {
11        return message;
12    }
13 }
14
15
16
```

Compile Run Settings Debugger Analysis Support

Start to compile Auto compile

browser/ballot.sol:HelloWorld Details Publish on Swarm



Static Analysis raised 2 warning(s) that requires your attention.

browser/ballot.sol:HelloWorld



Press details

Finding your ABI

```
« + browser/ba
1 pragma solidity
2
3 contract HelloWorld
4     string message;
5
6     function setMessage(string memory _message) public {
7         message = _message;
8     }
9
10    function getMessage() public view returns (string memory) {
11        return message;
12    }
13
14 }
15
16
```

NAME  



browser/ballot.sol:HelloWorld

METADATA  

- ▶ compiler:
 - language: Solidity
- ▶ output:
- ▶ settings:
- ▶ sources:
 - version: 1

BYTECODE  

6060604052341561000f57600080fd5b6102e38061001e6000396000f300...


INTERFACE - ABI  

- ▶ 0:
- ▶ 1:

Click on copy Interface - ABI icon

Settings Debugger Analysis Su

to compile ⚠

HelloWorld  Details

2 warning(s) that requires

HelloWorld

Using web3 (cont)

- Execute a method on your contract

```
var Web3 = require('web3');
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

//useful to check if something does not work
console.log(web3.version.api);

var contract = "0xe1e494d6a86fb02c4b976e08455d60013701a281";

var abi = [{"constant":false,"inputs":[{"name":"msgin","type":"string"}],"name":"setMessage","output":
var MyContract = web3.eth.contract(abi);
var myContractInstance = MyContract.at(contract);

console.log( myContractInstance.getMessage() );
```

Run using node:
node <YourJSFile>

Paste your contract address in quotes

Paste your abi, as is such that the line reads:
var abi = <PASTED ABI>;

Using web3 (cont)

- Get the message, set the message, then get the message

```
var Web3 = require('web3');
var web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

//useful to check if something does not work
console.log(web3.version.api);

var contract = "0xd00c4570323e10544ffc0a4f9d8badbd334e6692";

var abi = [{"constant":false,"inputs":[{"name":"msgin","type":"string"}],"name":"setMessage"}];
var MyContract = web3.eth.contract(abi);
var myContractInstance = MyContract.at(contract);

console.log( myContractInstance.getMessage() );
myContractInstance.setMessage('hello from JS', {from: web3.eth.accounts[0]});
console.log( myContractInstance.getMessage() );
```



Note, when updating state, we need to pass in the account that is executing the transaction

Anatomy of a Smart Contract

```
pragma solidity ^0.4.0;

contract Overview {
    uint256 someField;

    //constructor
    function Overview() public {
        someField = 0;
    }

    function publicStateChange(uint256 inputState) public {
        someField = inputState;
    }

    function publicConstant() public constant returns (uint256) {
        return someField;
    }

    function acceptsEther() public payable {
        someField += msg.value;
    }
}
```

Lab

- Go through the 'Solidity in Depth' tutorial starting from:
<http://solidity.readthedocs.io/en/develop/solidity-in-depth.html>
- All the way to the Cheat Sheet:
<http://solidity.readthedocs.io/en/develop/miscellaneous.html#cheatsheet>

Assignment

- Use case: A company's sole managing director wants to allow for the shareholders to make (binary) decisions, which he will propose to the share holders. For the sake of keeping decisions secret, until the director deems fit, the director would like to be the only one who is aware of how shareholders have voted until the director decides to let the shareholders know the outcome for a particular decision.

Assignment

- 1. The director will be the one to upload the contract. He should thereafter be recognised as the director because he was the one to upload the contract.
- 2. The director would like the ability to upload any number of questions (which require a true or false response to). The director will upload each question, one at a time.
 - Whenever the director uploads a decision, shareholders would need to be notified. An event can be used for this.

Assignment

- 3. The director would like the ability to add and remove shareholders from being able to vote and being able to see results for approved decisions at any point.
- 4. Each shareholder may only vote for each decision once.
- 5. The director would like to be able to see the number of votes in favour and against any decision at any point in time. The director would also like to receive updates as soon as a shareholder places a vote.
- 6. Shareholders should not be able to see the results, until the director has decided that they can.

Assignment

- 7. The director would like to allow for shareholders to check at any point what is the current question being asked of them.
- (more on next slide)

Assignment

- 8. A shareholder does not have time to vote, and is not sure if the director is taking note of who is voting. The shareholder also does not want the director to claim that the shareholder never took an interest in the company. The shareholder decides to write a blockchain edge node that will:
 - Check the contract (given the address and the ABI) for the latest question that is pending (this will be done every 5 minutes).
 - If the current question has not been answered, then the edge node should automatically place a random binary vote for the current question (given that the shareholder knows their own identifier)