

# Compiled Content

## Module 6

**MScFE 670**

**Data Feeds and Technology**

```
... ($this->repo_path = $repo_path; if ($parse_ini['bare']) {$this->repo_path = $repo_path; $this->
($repo_path."/config"); if ($parse_ini['bare']) {$this->repo_path = $repo_path; $this->
path = $repo_path; if ($_init) {$this->run('init');}} else {throw new Exception('"' . $r
* new Exception('"' . $repo_path . '"' is not a directory');}} else {if ($create_new) {if
)) {mkdir($repo_path); $this->repo_path = $repo_path; if ($_init) $this->run('init');}
istent directory');}} else {throw new Exception('"' . $repo_path . '"' does not exist');}}
t" directory) * * @access public * @return string */ public function git_directory_pat
repo_path."/ .git";}}/* * Tests if git is installed * * @access public * @return bool */
> array('pipe', 'w'), 2 => array('pipe', 'w'),); $pipes = array(); $resource = proc_open
t_contents($pipes[1]); $stderr = stream_get_contents($pipes[2]); foreach ($pipes as $pipe
return ($status != 127);}}/* * Run a command in the git repository * * Accepts a shell
.); $pipes = array(); /* Does not exist */
```

## Table of Contents

<b>Module 6: Smart Contract Development for Blockchains: Part 1 .....</b>	<b>3</b>
Unit 1: What is a Smart Contract?.....	4
Unit 2: Deploying Your First Smart Contract.....	8
Unit 3: Deploying Your First Smart Contract on a Live Network.....	18
Unit 4: Solidity .....	22
Bibliography .....	31



## **Module 6: Smart Contract Development for Blockchains: Part 1**

Module 1 begins with a brief history on financial markets and continues by introducing different classifications of financial markets and their key components. Different types of markets elements such as price determination, risk sharing, liquidity, and efficiency are compared. The module ends with a discussion on risk and the financial instruments used to mitigate it.



## Unit 1: What is a Smart Contract?

### Welcome

A common buzz-word thrown around in the blockchain community is “*smart contracts*”. Smart contracts are an important concept in modern blockchain technology. For the most part, they are at the core of many of the new blockchain projects that have been blossoming in recent years. Smart contracts allow blockchains to be programmable and multi-purpose, which extends blockchains’ applicability into diverse and interesting areas outside of just fintech.

In this module, we will focus on Ethereum, which is the world’s first production-ready smart contract platform. Not only was Ethereum the first smart contract platform, it also has the biggest developer community. This means it will likely continue to dominate as the primary smart contract platform for many years to come.

### Smart contracts

Why can’t we all just use Bitcoin? Why do we need another blockchain platform? In 2013, many of the early pioneers of blockchain technology were looking at ways to make Bitcoin more generally useful. To do this, they would often propose to add a new feature at the protocol level – i.e. to the Bitcoin core code itself – but this meant that people were trying to add many disconnected features to Bitcoin.

At this time, Vitalik Buterin, the creator and brain child of Ethereum, was working on a project called *Mastercoin*. This was an ambitious project about

representing non-Bitcoin assets on the Bitcoin blockchain and was pushing the capabilities of Bitcoin at the time. Buterin couldn’t help but feel that the process of adding features to Bitcoin was like adding new knives and screwdrivers to a Swiss Army knife. He thought, “Why can’t I build a tool that will allow me to write code for any tool I could possibly want to have in my blockchain application?”. It is from this that he came up with the idea of Ethereum: a Turing complete, general-purpose blockchain platform that you can build (almost) anything on.

Going back to the Swiss Army knife analogy, Bitcoin is to Ethereum as a Swiss Army knife is to a 3D printer. With a 3D printer, you can create as many different knives and screwdrivers as you wish, and much more!



So, to answer the original question, Bitcoin is limited. Bitcoin has a basic scripting language in it called Bitcoin Script, but it is not Turing complete. This means that Bitcoin Script can only perform simple conditionals, but not general computation. For a computational system or instruction set to be **Turing complete**, it means that the system can simulate any other Turing complete system, which in layman terms is just a general-purpose computer.

When we say that Ethereum is Turing complete, we are actually saying that it is *theoretically* capable of simulating another Turing complete system, such as an Android Phone. It would be extremely slow and expensive, and you would need to find a creative way for the I/O (In/Out interface, such as screen, touch input, and sound) to work, with lots of workarounds. However, no amount of skill, hard work, or creativity would make this possible to do on Bitcoin, hence Bitcoin Script is not Turing complete.

## How does Ethereum achieve smart contract capabilities?

A quick refresher – what is the basic data structure of a blockchain? Yes, a hashed linked list!

A hashed linked list's main strength is that any changes to the data in a given block can easily be detected, since the hash of the data in the block will change, causing a cascading change to all the hashes of each block. This is an important property for blockchains in general – data integrity.

In the case of Bitcoin, what is this data that is stored in the blockchain? It is the state of all of Bitcoin, including the history of all transactions, the accounts involved in those transactions, and some other metadata related to those accounts and transactions. The state of the Ethereum blockchain is somewhat different and intriguing. However, Bitcoin is more than just some data: it is a set of rules and logic about how this data can change over time. At its simplest, this logic dictates that only the owner of a given Bitcoin can send that Bitcoin to another address. Once again, in Ethereum these rules and logic about how the data state in its blockchain can change are different.

Computer code is a sequence of logical steps or instructions, and these computer instructions are just data. This is why you can download an Android application to your phone in the same way that you would download a photo or buy your favorite computer game as a CD. What was one of the things blockchains were good at (looking back to the previous paragraph)? Storing data. So what is preventing you from putting code into the blockchain?

In fact, the rules/logic about how the state of the blockchain can progress over time can be encoded into the blockchain itself. I encourage you to stop and think for a bit before reading on.



Think of why and how putting code in a blockchain is an interesting idea. Data is useful, and so is code, but putting them in the same place is a powerful combination.

Calling them smart contracts is a misnomer in many ways. They are not smart, but instead do exactly what they were told to do. They are also not contracts: they need not have anything to do with the legal concept of contracts. However, the idea and term were introduced by Nick Szabo many years before and the name has stuck.

Vitalik and Vlad Zamfir, another core Ethereum researcher, have joked about this unfortunate confusion caused by the name on Twitter and have suggested “persistent scripts” and “stored procedures” as alternatives.



Figure 1: Vitalik Buterin tweet

## DApps

**DApps** are decentralized applications, in that their architecture has no centralized point of failure. This gives these applications a number of very useful characteristics: they cannot be shut down or censored; the data contained in them is as eternal as the network itself; and end users are given power to control their own identities (as opposed to central authorities managing your identity for you).



In addition to the above-mentioned advantages for the canonical definition of a DApp, there are a few additional advantages in the case of Ethereum (as well as various other networks). Namely, they are trustless; “account” (i.e. your identity in the network) generation is permissionless; and only the logic in, and the state of, the network determines the outcomes of transactions.

With the clear advantages of decentralized applications comes a number of disadvantages, compared to centralized applications. There are high costs to participate in the system; data in the system is public by default, as well as difficult to obfuscate (although future iterations of the technology may make this possible); and any “mistakes” in the system are permanent.





## Unit 2: Deploying Your First Smart Contract

### Introduction

In this module, you'll be thrown into the deep end a bit. The purpose is to take you through a complete process of deploying your own smart contract to Ethereum, even before you write your own smart contract code. We will gloss over many of the details, but that doesn't mean you mustn't form questions as you work through this. Questions are good, as they give you a sense of direction and purpose in your learning, particularly with code.

### Getting started

To follow this tutorial, you need to have npm installed on your computer. For help and more details, [ethereum.stackexchange.com](https://ethereum.stackexchange.com) is by far the best place to ask questions and find answers. There is a good chance that someone else has had the same problem as you before.

To install node and npm follow the instructions here: [nodejs.org/en](https://nodejs.org/en). My preferred method of installing on Linux and Mac is with [nvm](https://github.com/nvm-sh/nvm) (node version manager). You will also need a Web3/Ethereum-enabled browser, so I recommend using [MetaMask](https://metamask.io) for this purpose. It is a browser plugin for Chrome and Firefox and will act as your Ethereum wallet. With it installed, your web-browser will be able to interact with the Ethereum blockchain and digitally sign messages and transactions with your private key.

Node is a JavaScript engine (i.e. the thing that runs JavaScript code) and JavaScript is one of the most convenient ways to interact with your DApp. Ethereum implements a protocol and interface or API (Application Programming Interface) which you need to confirm with when interacting with Ethereum. Thus, any language can interact with the Ethereum blockchain, as long as they follow the protocol that is specified for doing so.

Libraries are pieces of code that abstract the complexity of this interaction into something that is easier for the programmer, just as Gmail abstracts the complexity of sending an email from the user. There are many different libraries that one can use to interact with Ethereum. In fact, due to the open source nature of Ethereum, anyone can create their own library or modify and make the libraries of others better.





## Truffle and Ganache

The Ethereum space is continually evolving, and this is one of the things that makes working with Ethereum exciting (although some may say stressful). As such, programmers have to make a considered decision about which programming language and libraries to use when starting a software project. Since they all have tradeoffs, making these decisions is something that takes experience. For the purpose of this course, we will use Truffle ([truffleframework.com](https://truffleframework.com)) since it is mature; has a big user base; is well maintained by one of the most prominent blockchain companies, ConsenSys; and is easy to use.

You will also need to use the terminal (or cmd line for Windows) for this tutorial. All this means is that you will need to copy and paste text that your computer will execute as a command. This is more straightforward on Linux and Mac, but if you have both Windows Powershell and Linux Subsystem for Windows, the procedure will work. Once node/npm is installed, run `npm install -g truffle ganache-cli`, to install Truffle and Ganache. Ganache is an Ethereum blockchain simulator, simulating a single locally running node that you can use to test and develop your blockchain applications. It is important that you get used to using the terminal because many software tools don't have an alternative.

```

→ tutorial npm install -g truffle ganache-cli
/home/jasoons/.nvm/versions/node/v10.13.0/bin/ganache-cli -> /home/jasoons/.nvm/versions/node/v10.13.0/lib/node_modules/ganache-cli/cli.js
/home/jasoons/.nvm/versions/node/v10.13.0/bin/truffle -> /home/jasoons/.nvm/versions/node/v10.13.0/lib/node_modules/truffle/build/cli.bundled.js
+ truffle@5.0.1
+ ganache-cli@6.2.5
added 54 packages from 46 contributors and updated 1 package in 4.379s
→ tutorial
```

*Figure 2: Output on Truffle and Ganache installation*

Truffle comes with truffle boxes, which are minimally preconfigured projects to help you get started. This tutorial will use the simple [React truffle box](#) as a starting point, for the sake of simplicity.

First, we will test and run this app on a local test Ethereum environment (Ganache), after which we will deploy the smart contracts to the Rinkeby test net and run it from there.

Open the terminal in a folder you want to work from and run `truffle unbox react`. This will download the pre-made boilerplate which is a minimal set-up of Truffle and React. Take note that you will have to wait a few minutes for it to download and configure everything.



```
→ tutorial truffle unbox react

✓ Preparing to download
✓ Downloading
✓ Cleaning up temporary files
✓ Setting up box

Unbox successful. Sweet!

Commands:

Compile:          truffle compile
Migrate:          truffle migrate
Test contracts:   truffle test
Test dapp:        cd client && npm test
Run dev server:   cd client && npm run start
Build for production: cd client && npm run build
```

*Figure 3: Output of Truffle unboxing and configuration*

Run `cd client` to move to the *client* directory and `npm start` to fire up the UI.

Then, view it in your browser at <http://localhost:3000>. After a few moments you should see the following error:

### Failed to compile

```
./src/App.js
Module not found: Can't resolve './contracts/SimpleStorage.json' in '/home/jasoons/Documents/code/tutorial/client/src'
```

*Figure 4: Output of compile error*

It is very important to read error messages when trying to code. They can be frustrating, but they are often your only help. This particular error message means that the 'SimpleStorage.json' file is not there, and so this sample DApp doesn't know what the interface of the 'SimpleStorage.sol' smart contract is. To fix this, run `truffle compile` in the parent directory – i.e. do `cd ..` to go out of the client folder. When you run `npm start` again, you should see the following output in the terminal and in your browser respectively.



```
Compiled successfully!  
  
You can now view client in the browser.  
  
Local:      http://localhost:3000/  
On Your Network: http://192.168.1.142:3000/  
  
Note that the development build is not optimized.  
To create a production build, use yarn build.
```

*Figure 5: Output on successful contract compile*

Loading Web3, accounts, and contract...

*Figure 6: Shown in the browser when the page is loaded*

Often when you are stuck, it is useful to check the browser console for errors or hits. To do this, press Ctrl/Cmd + Shift + J and you should see the following warning:

⚠ ATTENTION: In an effort to improve user privacy, MetaMask stopped exposing user accounts `inpage.js:1` to dapps if "privacy mode" is enabled on November 2nd, 2018. Dapps should now call `provider.enable()` in order to view and use accounts. Please see <https://bit.ly/2QQHXvF> for complete information and up-to-date example code.

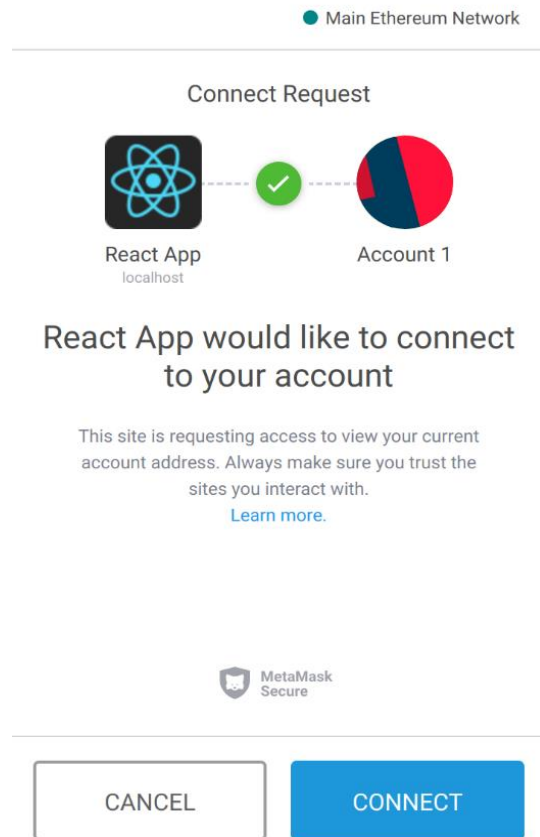
*Figure 7: Browser console log of MetaMask*

## MetaMask

Yes, you read it right – for over 2 years MetaMask would broadcast your Ethereum address to every website you used while the plugin was installed, which rightfully might make some of you uncomfortable. Thankfully, MetaMask have recently implemented EIP (Ethereum Improvement Proposal) 1102, and must explicitly grant access to any website that wants to see your Ethereum address.

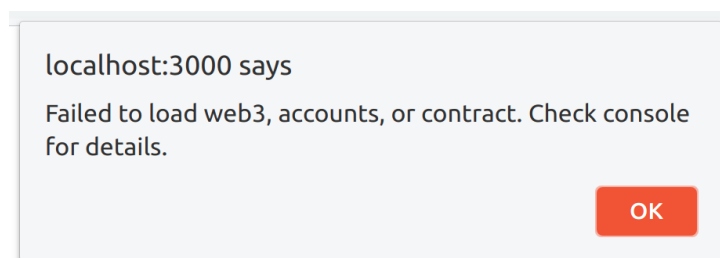
You can accept the prompt from MetaMask:





*Figure 8: MetaMask authorization prompt*

If you get the following error instead, it means that your MetaMask isn't even unlocked, so click on MetaMask and enter your password.



*Figure 9: Locked MetaMask prompt*

Now, you should get the following error:

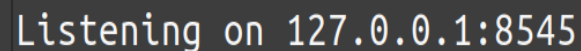
Unhandled Rejection (Error): This contract object doesn't have address set yet, please set an address first.

*Figure 10: Output of unlock error*

This means that your contracts have not been deployed to the current network, although they have been compiled from human readable text into something that the Ethereum network can execute.

To resolve this, we need to start an Ethereum network using Ganache. Run `ganache-cli -m "put your twelve-word passphrase from MetaMask ..."`; where you replace the “put your twelve-word passphrase from MetaMask ...” with your actual MetaMask passphrase. This command starts a blockchain simulation software where the same Ethereum accounts from your MetaMask are “unlocked”, meaning they can be used to sign transactions. There is a GUI for Ganache which you can use, but I’d encourage you to experiment as much as possible with different tools and ways of doing things.

When you run `ganache-cli`, it will say something like:

A dark rectangular box with a light gray border containing the text "Listening on 127.0.0.1:8545" in a light gray monospace font. A small white cursor is visible at the end of the line.

*Figure 11: Output of successful blockchain instance*

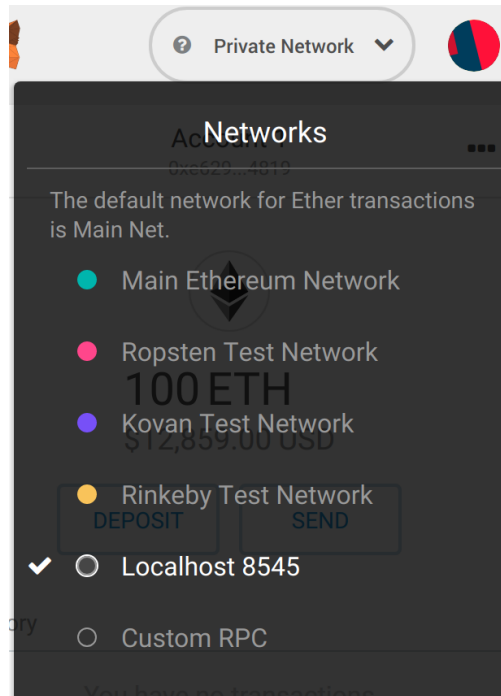
This means that we need to set MetaMask to point to this blockchain instance.

Remember, Ethereum is both a protocol and a platform. If someone says they are using Ethereum to build their product, it could mean they are building it on the public network; or it could mean that they are building it on a network that implements the Ethereum protocol, but that could be privately hosted by a smaller group or consortium.

Thus, MetaMask gives you the ability to switch which network you are connected to. With MetaMask, you always remain in control of your keys, since they remain in your browser and you remain in control of your interactions with the application. This is inherently different to a “web 2.0” website like Facebook, as Facebook theoretically has full control over your interactions with them and their centralized service.

Select the network on localhost (localhost is an alias for 127.0.0.1) port 8545 from the dropdown at the top of MetaMask:





*Figure 12: MetaMask network options*

You will now see that 100.0 Eth has appeared in our account. This is test Ether and has no relation to the Main network.

## Deploy to test Ethereum network

Now, we are ready to deploy the contracts to this test Ethereum network so that the DApp can interact with them. To do this, you can run the command `truffle migrate`, but doing this will give the following error when you run it now:

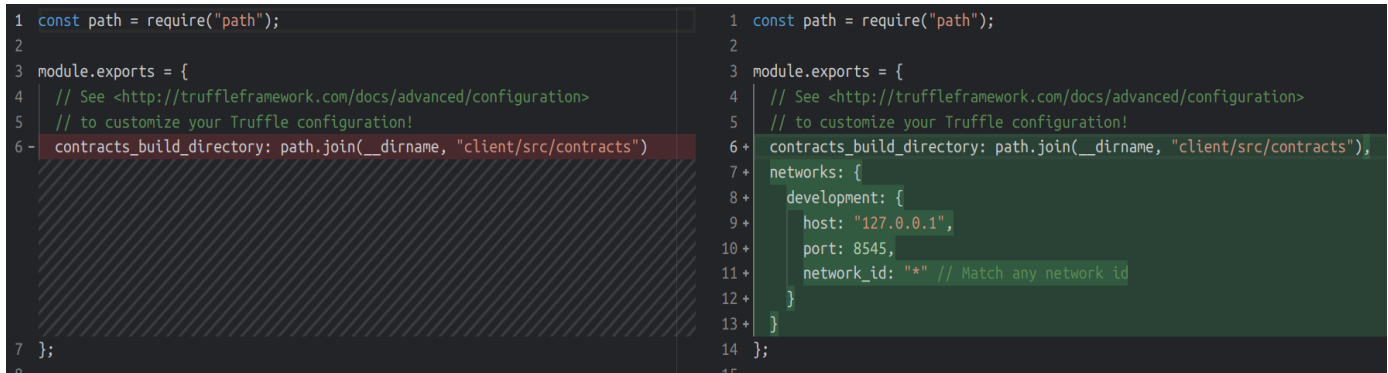
```
→ tutorial truffle migrate
Could not connect to your Ethereum client. Please check that your Ethereum client:
- is running
- is accepting RPC connections (i.e., "--rpc" option is used in geth)
- is accessible over the network
- is properly configured in your Truffle configuration file (truffle.js)
```

*Figure 13: Output of unsuccessful deployment to Ethereum client*

From reading this error, you may well be able to work out what you need to do. Simply, we need to tell Truffle some basic configuration details about this network that you want to deploy the contracts to. Without it, Truffle doesn't know what to do. (See the details of how to do this, and other caveats specific to Windows, in the documentation available at [truffleframework.com/docs/advanced/configuration](https://truffleframework.com/docs/advanced/configuration).)



The main thing you need to do is change the 'truffle-config.js' file to look like the following:



```
1 const path = require("path");
2
3 module.exports = {
4   // See <http://truffleframework.com/docs/advanced/configuration>
5   // to customize your Truffle configuration!
6 -  contracts_build_directory: path.join(__dirname, "client/src/contracts")
7 };
8
```

```
1 const path = require("path");
2
3 module.exports = {
4   // See <http://truffleframework.com/docs/advanced/configuration>
5   // to customize your Truffle configuration!
6 +  contracts_build_directory: path.join(__dirname, "client/src/contracts"),
7 +  networks: {
8 +    development: {
9 +      host: "127.0.0.1",
10 +      port: 8545,
11 +      network_id: "*" // Match any network id
12 +    }
13 +  }
14 };
15
```

*Figure 14: Update Truffle-config.js file*

Re-running the migrate command should now work fine:



```

Starting migrations...
=====
> Network name:      'development'
> Network id:        1547306493772
> Block gas limit: 6721975

1_initial_migration.js
=====

  Deploying 'Migrations'
  -----
  > transaction hash:      0xdc65a7e59fe27dfd65f7bf0de6ec70bf4420476047a192e9d42405fbdb8312d4
  > Blocks: 0              Seconds: 0
  > contract address:      0x29818AEF86B3136a8EC8AE542B7c61DbdE5c26Af
  > account:               0xe629E846b7E26Eb6AD3fc8043FFfc1Bbb14b4819
  > balance:               99.99430184
  > gas used:              284908
  > gas price:             20 gwei
  > value sent:            0 ETH
  > total cost:            0.00569816 ETH

  > Saving migration to chain.
  > Saving artifacts
  -----
  > Total cost:            0.00569816 ETH

2_deploy_contracts.js
=====

  Deploying 'SimpleStorage'
  -----
  > transaction hash:      0x4e106e9e720ebf56765a4736d92b7a1306d2dbe2e68ba7cfc1e3e2a52a2c2649
  > Blocks: 0              Seconds: 0
  > contract address:      0xA2e28634942fBc3cb7a6C64746d0f1B85e6b7A97
  > account:               0xe629E846b7E26Eb6AD3fc8043FFfc1Bbb14b4819
  > balance:               99.99114582
  > gas used:              115767
  > gas price:             20 gwei
  > value sent:            0 ETH
  > total cost:            0.00231534 ETH

  > Saving migration to chain.
  > Saving artifacts
  -----
  > Total cost:            0.00231534 ETH

Summary
=====
> Total deployments:      2
> Final cost:             0.0080135 ETH

```

*Figure 15: Output of successful deployment to Ethereum client*

Now, if you reload the website, you should get a notification from MetaMask asking you to sign a transaction. Normally, you should be very careful when signing random transactions and ensure



that it is what you want to do. Remember that actions on the blockchain are generally irreversible, but this time you can trust me and just click accept.

Congratulations, you have just signed your first Ethereum transaction inside a DApp! This changed the value of a number on the blockchain from 0 to 5. While simple, it is a good step on your journey into the future of open finance.

## Additional insights

Homework task: how do you make it change to another number besides 5? The hint is written in the DApp itself. Try change it to 6.

Another strange thing is that when you reload the website the value resets to zero. Let's try to fix that. We need to load the value of the variable from the blockchain when the DApp loads. To do this make the following changes to the 'client/src/App.js' file:



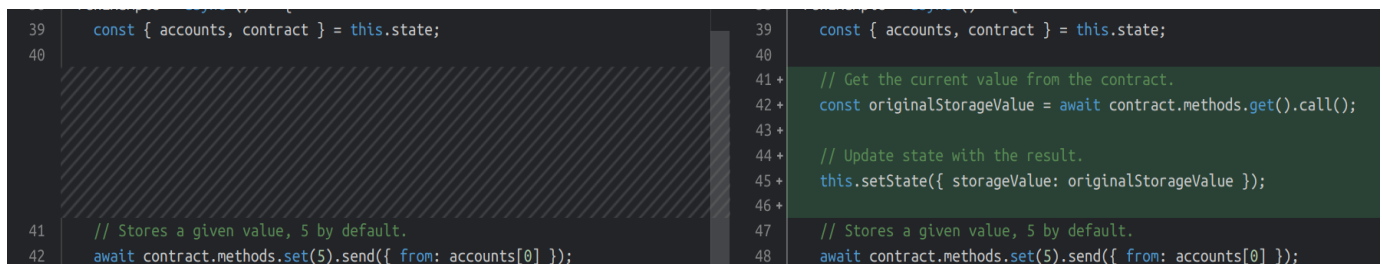
```

7 class App extends Component {
8 - state = { storageValue: 0, web3: null, accounts: null, contract: null };
9
7 class App extends Component {
8 + state = { storageValue: null, web3: null, accounts: null, contract: null };
9

```

Figure 16: Updated App.js file

Here you don't set the initial 'storageValue' but rather get this value from the blockchain (in the next code snippet).



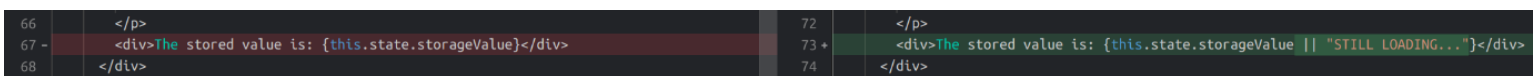
```

39 const { accounts, contract } = this.state;
40
41 // Stores a given value, 5 by default.
42 await contract.methods.set(5).send({ from: accounts[0] });
39 const { accounts, contract } = this.state;
40
41 + // Get the current value from the contract.
42 + const originalStorageValue = await contract.methods.get().call();
43 +
44 + // Update state with the result.
45 + this.setState({ storageValue: originalStorageValue });
46 +
47 // Stores a given value, 5 by default.
48 await contract.methods.set(5).send({ from: accounts[0] });

```

Figure 17: Updated App.js file

Notice the similarity of these lines with lines 44-48 in the original code. (Hint: they do the exact same thing.)



```

66 </p>
67 - <div>The stored value is: {this.state.storageValue}</div>
68 </div>
72 </p>
73 + <div>The stored value is: {this.state.storageValue || "STILL LOADING..."}</div>
74 </div>

```

Figure 18: Updated HTML file

The above code changes what the website will display when the 'state.storageValue' variable is null. When its value is set, it will display that value.



## Unit 3: Deploying Your First Smart Contract on a Live Network

### Introduction

The code interacts with Ethereum via the Web3 provider. In this example, the Web3 provider is injected into the page by MetaMask, but the Web3 provider can be designed to be provided by anything, and so sign the message by different means. For example, the mobile wallet called Status.im could also load this same web DApp. As long as it injects a provider with the same interface, the app can work. It also means that the actual signing of the transactions can happen on something like a hardware wallet for additional security, but nothing in the app needs to change (as long as the same Web3 provider interface is injected).

### Rinkeby test network

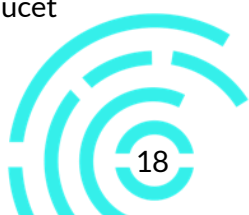
Now that we have deployed our contract to our local running test net, we should try get our code running on a public network. We will use the Rinkeby test network for this purpose.

There are a number of different networks with different properties. *Rinkeby* runs on the **proof-of-authority** (POA) consensus algorithm, which means that there are a predefined set of nodes validating and processing transactions. This makes this network stable and predictable – provided these nodes stay online. *Kovan* test network is similar to Rinkeby, as it also uses a POA consensus algorithm, but Rinkeby is run by the Geth (Go Ethereum) client (code/software), whereas Kovan is run by the Parity client.

Another notable test network is *Ropsten*, which is the most similar to the live Ethereum network. You can also develop on Ropsten since it is most similar to the main net, but this means that it is using **proof-of-work** (POW). Since there is no economic

incentive – i.e. because Ether on the test net has no value – it has a low hashrate. It is a popular target for budding hackers who want to cause issues since this means it doesn't have much security.

Set MetaMask to use the 'Rinkeby Test Network'. If your balance is zero, go to this [faucet](#) to get your hands on some test Ether for your primary Ethereum address – i.e. your first address in MetaMask. A **faucet** is a website that allows users to earn small amounts of a cryptocurrency by completing small tasks, rather than purchasing the currency directly. Depending on the faucet



used, these tasks may involve completing captchas, viewing ads, or publicly posting your wallet address to a social media account. Faucets lower the barrier of entry to becoming a cryptocurrency user, as they offer a low-risk way of getting some cryptocurrency without the need to purchase it. One such faucet that dispenses Ether for use on the Rinkeby network is [faucet.rinkeby.io](https://faucet.rinkeby.io)

To do this, you will need to post your Ethereum address on a public channel like Facebook, Twitter or Google+. Deploying contracts to Ethereum is an expensive process – it can be many dollars for complex contracts on a busy day when the network is congested, and you are unlikely to get it right the first time. Therefore, I highly recommend practicing with test Ether. Doing things with real money is scary too. Networks like the live POA network are much cheaper to use, so this is an option if you want to test on a live network. The process is exactly the same.

## Connecting to the Rinkeby test network

We will be using Infura as a hosted node for the sake of convenience. Go ahead and create an Infura account at [infura.io/signup](https://infura.io/signup), and record the key that you receive. Infura handles many of the transactions and interactions with the Ethereum network and are often criticized for being centralized. Although this is true, they are extremely convenient and completely opt-in to use. You can use a normal Ethereum node, if

you want, or run your own node – there is nothing stopping anyone from doing that. We will briefly cover how to use a locally running Ethereum node in the next section. The important thing to remember is that Ethereum is an open and permissionless system, so the ability for anyone to participate with a node in the network is an important one.

Next, run `npm i -s truffle-hdwallet-provider` to install a provider that will connect you to the chosen network, sign your transactions, and submit these transactions to the network. This “provider” is used by Web3 to submit transactions to the Ethereum network and, in this case, sign these transactions. To use the provider, add the following to your ‘truffle.js’ file. We are using a `network_id` of 4 for Rinkeby, but other network IDs that may be of interest are 1, 3, 42 for mainnet, Ropsten and Kovan respectively.



```

1 const path = require("path");
2
3 module.exports = {
4   // See <http://truffleframework.com/docs/advanced/configuration>
5   // to customize your Truffle configuration!
6   contracts_build_directory: path.join(__dirname, "client/src/contracts"),
7   networks: {
8     development: {
9       host: "127.0.0.1",
10      port: 8545,
11      network_id: "*" // Match any network id
12    }
13  }
14 };

```

```

1 const path = require("path");
2 const HDWalletProvider = require("truffle-hdwallet-provider");
3 const mnemonic = "your twelve word passphrase from metamask ...";
4 const infuraAccessToken = '<<your_infura_access_token>>';
5
6 module.exports = {
7   // See <http://truffleframework.com/docs/advanced/configuration>
8   // to customize your Truffle configuration!
9   contracts_build_directory: path.join(__dirname, "client/src/contracts"),
10  networks: {
11    development: {
12      host: "127.0.0.1",
13      port: 8545,
14      network_id: "*" // Match any network id
15    },
16    rinkeby: {
17      provider: function() {
18        return new HDWalletProvider(mnemonic, "https://rinkeby.infura.io/" + infuraAccessToken);
19      },
20      network_id: 4
21    }
22  }
23 };

```

Figure 19: Updated Truffle.js

## Deploy to the Rinkeby test network

Now you are ready to run the migrations again, so use the `truffle migrate --network rinkeby` command. This will take a bit longer than last when we used Ganache, since we need to wait for the blocks to be mined by the network and the average blocktime on the Rinkeby Network is 15 seconds.

```

Running migration: 1_initial_migration.js
Deploying Migrations...
... 0x51085771dfc78b31bc784546f2113a4fd408b8b478611f32c546c807810ec7d6
Migrations: 0xd5491a69b87db56c770e69541d72595063a6cf58
Saving successful migration to network...
... 0x547bcae81ecfdf9876e021fd910ebcccd8e7c0c1459631a43b3688e6936e465
Saving artifacts...
Running migration: 2_deploy_contracts.js
Deploying SimpleStorage...
... 0xbdf619456e2a48065c176304b806b9d334bfd8b5cd389d477ac8d5007f9ba9e5
SimpleStorage: 0xb7e0f4bb4043d7011941c20b265242a5423917a9

```

Figure 20: Output of successful deployment to Rinkeby test net

You can then copy the hash from the transaction, which in this case is `0xbdf619456e2a48065c176304b806b9d334bfd8b5cd389d477ac8d5007f9ba9e5`, and search for it on [rinkeby.etherscan.io](https://rinkeby.etherscan.io) to confirm and view the details of your transaction.

Open the user interface again and confirm the transaction to set the storage and confirm that the transaction went through on [Etherscan](https://etherscan.io).

Congratulations; you have deployed your first contract to a public network!

## Challenge

If you want to try something slightly more difficult, you can try deploying your contract with this modified [provider engine](#) that uses a private key directly rather than a mnemonic phrase. This is directly modified from the original [truffle-hdwallet-provider](#).

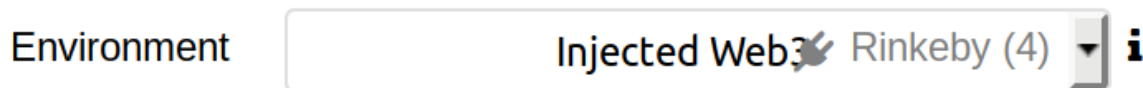
### *Using a locally running mode*

You can also deploy directly to a local node with an unlocked account. Please be extremely careful with this method – if your Ethereum node is opened up to the public, they can execute a transaction with your unlocked account (read: “steal all your Ethereum”). This can be done with Parity or Geth running locally. Please refer to their respective documentation to set them up.

Once your node is running correctly, you can deploy your contract in exactly the same way as you did with Ganache, since it also just provides you with a set of unlocked accounts.

### *Another fun way to deploy your contracts to public networks*

You can use [remix.ethereum.org](https://remix.ethereum.org) and paste your code there. Alternatively, you can use [remixd](#), choose your environment (Injected Web3 is referring to MetaMask, as seen in the image below), and click deploy. If you use this method, you will need to manually save and link the contract address to Truffle if you want to use Truffle for your UI.



*Figure 21: Remix environment configuration*

I hope you had fun deploying your first DApp to the world! That code will stay there for as long as the network is running, which should be a very long time.

## Unit 4: Solidity

### Introduction

In this section, we will edit the Solidity code in our example to learn some basics about the Solidity programming language. We will use Remix ([remix.ethereum.org](https://remix.ethereum.org)) to edit our code for convenience. It is an online integrated development environment (IDE) and has a convenient out-of-the-box way to interact with, debug, and develop smart contracts. You can work entirely in Remix if you want, but it will be a nice challenge for you to extend this further and integrate this code back into the project; work on improving the interface for it; and turn it into a real DApp.

Learning to program is like learning a language – you don't learn unless you try to talk. The internet has many resources to guide you right from being a beginner to becoming an advanced user. Building your own projects is fun! If you enjoyed playing with something like Lego when you were younger, then there is a good chance that you will enjoy programming.

### Getting started

To sync the code in your local folder with the code in the browser in Remix, a tool called remixd ([www.npmjs.com/package/remixd](https://www.npmjs.com/package/remixd)) is very useful. It avoids the need for back and forth copy-and-paste between browser and editor.

It is also useful to get some of the best plugins for your editor of choice. Some recommendations of add-ons that might be useful for developing Ethereum can be found here: [github.com/bkrem/awesome-solidity#editor-plugins](https://github.com/bkrem/awesome-solidity#editor-plugins).

You can also choose which Ethereum environment you want to connect to within Remix. The “JavaScript VM” (VM = Virtual Machine) is very convenient, fast, and

is managed in the browser. “Injected Web3” is what is injected by MetaMask (which is convenient since you get the full power of MetaMask). “Web3 Provider” is any external endpoint that you specify that you can connect to.





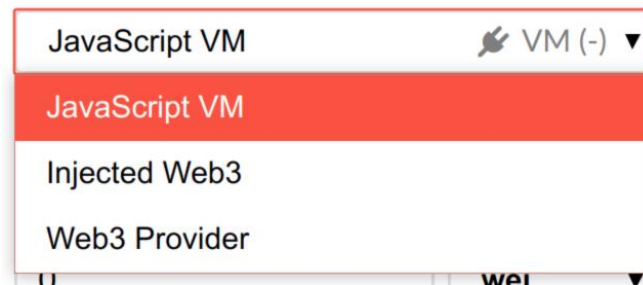


Figure 22: Remix environment options

## Editing Solidity code

### *Adding a constructor*

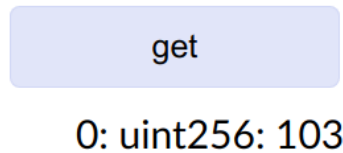
Let's try to add a constructor to our smart contract. A constructor is a way to give the contract extra data at initialization. We will make our constructor take in an initial value.

```

1 pragma solidity ^0.5.0;
2
3 contract SimpleStorage {
4     uint storedData;
5
6     function set(uint x) public {
7         storedData = x;
8     }
9
10    function get() public view returns (uint) {
11        return storedData;
12    }
13 }
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2
```

*Figure 24: Remix deployment options*

Then, test the value on the deployed contract with the 'get' function:

*Figure 25: Output for Remix deployment test*

Next, we can remove the 'get' function from this contract by making the variable 'public'. In Ethereum, everything is public, so anyone can read any application or contract state directly from the blockchain. However, you can allow code and other contracts to access the data by exposing those variables publicly. By making a variable 'public', you are making it easy for other contracts (or humans) to read the value, and thus there is no more need for our 'get' function. How would you update the DApp from the previous example so that it can still read the stored value? Experiment and Google around until you have found the answer.

*Figure 26: Removed get function*

### *Adding access control*

Currently, there is no access control. This means that anyone with any account can change any number. Let's make this application give each user their own number. To do this, we will use a

mapping, which is like a dictionary look-up of values, and use the Ethereum address of the user as the key. To get this Ethereum address in code, use the 'msg.sender' variable.

<pre> 1 pragma solidity ^0.5.0; 2 3 contract SimpleStorage { 4 -   uint public storedData; 5 6   constructor(uint initialValue) public { 7 -     storedData = initialValue; 8   } 9 10  function set(uint x) public { 11 -    storedData = x; 12  } 13 } 14 </pre>	<pre> 1 pragma solidity ^0.5.0; 2 3 contract SimpleStorage { 4 +   mapping(address =&gt; uint) public storedData; 5 6   constructor(uint initialValue) public { 7 +     storedData[msg.sender] = initialValue; 8   } 9 10  function set(uint x) public { 11 +    storedData[msg.sender] = x; 12  } 13 } 14 </pre>
--	---

Figure 27: Added mapping

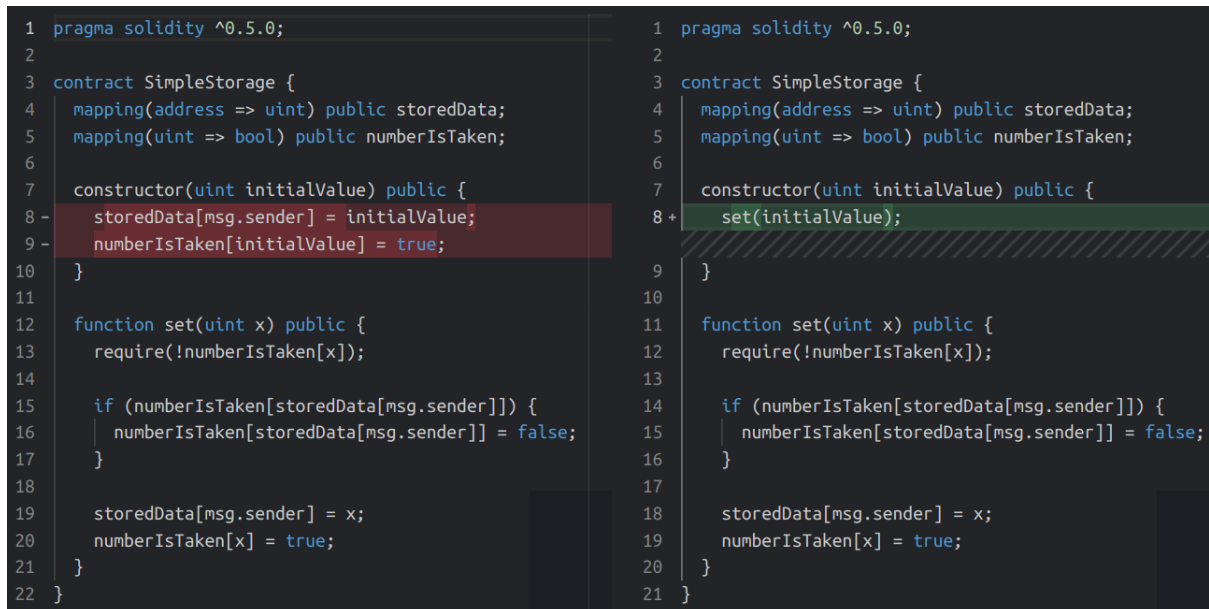
What if we want to make sure that there are no duplicate numbers? We can use another mapping for this. Please make sure every line of this makes sense before moving on. Ask yourself, what would happen if I removed this line from the code? What would people be able to do now that they weren't able to do before?

<pre> 1 pragma solidity ^0.5.0; 2 3 contract SimpleStorage { 4   mapping(address =&gt; uint) public storedData; 5 6   constructor(uint initialValue) public { 7     storedData[msg.sender] = initialValue; 8   } 9 10  function set(uint x) public { 11 12 13  } 14 </pre>	<pre> 1 pragma solidity ^0.5.0; 2 3 contract SimpleStorage { 4   mapping(address =&gt; uint) public storedData; 5 +   mapping(uint =&gt; bool) public numberIsTaken; 6 7   constructor(uint initialValue) public { 8     storedData[msg.sender] = initialValue; 9 +     numberIsTaken[initialValue] = true; 10  } 11 12  function set(uint x) public { 13 +    require(!numberIsTaken[x]); 14 + 15 +    if (numberIsTaken[storedData[msg.sender]]) { 16 +      numberIsTaken[storedData[msg.sender]] = false; 17 +    } 18 + 19     storedData[msg.sender] = x; 20 +     numberIsTaken[x] = true; 21  } 22 } 23 </pre>
--	--

Figure 28: Updated mapping to avoid collisions

## Cleaning up your code

There is some duplication in this code. A common acronym in software development is DRY (don't repeat yourself). Let's remove this duplication by using function abstraction. When writing code, it is important to always look over your code and perform refactoring. **Refactoring** is cleaning up your code by making it simpler, easier to read, or more efficient, without changing the underlying functionality. If you don't refactor constantly, you run the risk of creating messy and disorganized code.



```

1 pragma solidity ^0.5.0;
2
3 contract SimpleStorage {
4     mapping(address => uint) public storedData;
5     mapping(uint => bool) public numberIsTaken;
6
7     constructor(uint initialValue) public {
8         storedData[msg.sender] = initialValue;
9         numberIsTaken[initialValue] = true;
10    }
11
12    function set(uint x) public {
13        require(!numberIsTaken[x]);
14
15        if (numberIsTaken[storedData[msg.sender]]) {
16            numberIsTaken[storedData[msg.sender]] = false;
17        }
18
19        storedData[msg.sender] = x;
20        numberIsTaken[x] = true;
21    }
22 }
  
```

```

1 pragma solidity ^0.5.0;
2
3 contract SimpleStorage {
4     mapping(address => uint) public storedData;
5     mapping(uint => bool) public numberIsTaken;
6
7     constructor(uint initialValue) public {
8         set(initialValue);
9     }
10
11    function set(uint x) public {
12        require(!numberIsTaken[x]);
13
14        if (numberIsTaken[storedData[msg.sender]]) {
15            numberIsTaken[storedData[msg.sender]] = false;
16        }
17
18        storedData[msg.sender] = x;
19        numberIsTaken[x] = true;
20    }
21 }
  
```

Figure 29: Refactored constructor function

Currently, users of this contract can choose any 'uint' which can take a value from 0 to  $2^{256}$ , which is a very big range. To be precise,  $2^{256}$  is 115792089237316195423570985008687907853269984665640564039457584007913129639936, which is 115 quattuorvigintillion. (That's a 78-digit number!) Let's set a more constrictive range of numbers.



```

1 pragma solidity ^0.5.0;
2
3 contract SimpleStorage {
4     mapping(address => uint) public storedData;
5     mapping(uint => bool) public numberIsTaken;
6
7     constructor(uint initialValue) public {
8         storedData[msg.sender] = initialValue;
9         numberIsTaken[initialValue] = true;
10    }
11
12    function set(uint x) public {
13        require(!numberIsTaken[x]);
14
15        if (numberIsTaken[storedData[msg.sender]]) {
16            numberIsTaken[storedData[msg.sender]] = false;
17        }
18
19        storedData[msg.sender] = x;
20        numberIsTaken[x] = true;
21    }
22 }

```

```

1 pragma solidity ^0.5.0;
2
3 contract SimpleStorage {
4     mapping(address => uint) public storedData;
5     mapping(uint => bool) public numberIsTaken;
6     uint public rangeMin;
7     uint public rangeMax;
8
9     constructor(
10         uint initialValue,
11         uint _rangeMin,
12         uint _rangeMax
13     ) public {
14         set(initialValue);
15
16         rangeMin = _rangeMin;
17         rangeMax = _rangeMax;
18     }
19
20     function set(uint x) public {
21         require(!numberIsTaken[x]);
22         require(x > rangeMin && x < rangeMax);
23
24         if (numberIsTaken[storedData[msg.sender]]) {
25             numberIsTaken[storedData[msg.sender]] = false;
26         }
27
28         storedData[msg.sender] = x;
29         numberIsTaken[x] = true;
30     }
31 }

```

*Figure 30: Added range limits*

For help following the process, please see the video included in the Additional Resources section for a practical demonstration of how this code is implemented.

Let's give this code some meaning. Let's make it a lottery of sorts, where you have to pay to hold a number. Then we must set the price of a ticket in the constructor and allow the 'set' function to accept payment. I will include some more logic that you should think about in your own time. Ask yourself, what are the tradeoffs? How would you implement this differently? Why? When you think of different ways, put in the effort and implement them – that is the only way to learn to code. You can't learn to code if you never code, sadly.



<pre> 1 pragma solidity ^0.5.0; 2 3 contract SimpleStorage { 4     mapping(address =&gt; uint) public storedData; 5     mapping(uint =&gt; bool) public numberIsTaken; 6     uint public rangeMin; 7     uint public rangeMax; 8 9     constructor( 10         uint initialValue, 11         uint _rangeMin, 12         uint _rangeMax 13     ) public { 14         set(initialValue); 15 16         rangeMin = _rangeMin; 17         rangeMax = _rangeMax; 18     } 19 20     function set(uint x) public { 21 22         require(!numberIsTaken[x]); 23         require(x &gt; rangeMin &amp;&amp; x &lt; rangeMax); 24 25         if (numberIsTaken[storedData[msg.sender]]) { 26             numberIsTaken[storedData[msg.sender]] = false; 27         } 28 29         storedData[msg.sender] = x; 30         numberIsTaken[x] = true; 31     } 32 } </pre>	<pre> 1 pragma solidity ^0.5.0; 2 3 contract SimpleStorage { 4     mapping(address =&gt; uint) public storedData; 5     mapping(uint =&gt; bool) public numberIsTaken; 6     uint public rangeMin; 7     uint public rangeMax; 8     uint public ticketCost; 9 10    constructor( 11        uint _ticketCost, 12        uint _rangeMin, 13        uint _rangeMax 14    ) public { 15        ticketCost = _ticketCost; 16 17        rangeMin = _rangeMin; 18        rangeMax = _rangeMax; 19    } 20 21    function set(uint x) public payable { 22        require( 23            storedData[msg.sender] != 0    24            msg.value &gt;= ticketCost 25        ); 26 27        require(!numberIsTaken[x]); 28        require(x &gt; rangeMin &amp;&amp; x &lt; rangeMax); 29 30        if (numberIsTaken[storedData[msg.sender]]) { 31            numberIsTaken[storedData[msg.sender]] = false; 32        } 33 34        storedData[msg.sender] = x; 35        numberIsTaken[x] = true; 36    } 37 } </pre>
---	--

Figure 31: Added ticket

Here, if the person sends too much Ether, the money is taken as a donation. Is that good? Can you think how to implement the 'set' function such that it returns any extra Ether back to the user. Another challenge for you – can you change this code so that it accepts an ERC20 token instead of Ether?

Can you see any major bug (as in a serious bug that could cause money to be lost forever) in the code? Take a moment and think about it!

If you noticed that there was no way to withdraw the money that is sent to the contract, congratulations! This contract is like a black hole, as it can only accept funds. Once they are in the contract, there is no way to possibly move them.



<pre> 32 33     storedData[msg.sender] = x; 34     numberIsTaken[x] = true; 35 } 36 37 </pre>	<pre> 32 33     storedData[msg.sender] = x; 34     numberIsTaken[x] = true; 35 } 36 + 37 +     function withdrawal() public { 38 +         msg.sender.transfer(address(this).balance); 39 +     } 40 } 41 </pre>
---	--

Figure 32: Added withdrawal function

## Adding a withdrawal function

Now we have a withdrawal function, but are there any restrictions on who can call it? Let's make it so that only the "owner" of the smart contract can call the withdrawal function, where the "owner" is the creator of the smart contract.

<pre> 1  pragma solidity ^0.5.0; 2 3  contract SimpleStorage { 4      mapping(address =&gt; uint) public storedData; 5      mapping(uint =&gt; bool) public numberIsTaken; 6      uint public rangeMin; 7      uint public rangeMax; 8      uint public ticketCost; 9 10     constructor( 11         uint _ticketCost, 12         uint _rangeMin, 13         uint _rangeMax 14     ) public { 15         ticketCost = _ticketCost; 16 17         rangeMin = _rangeMin; 18         rangeMax = _rangeMax; 19     } 20 21     function set(uint x) public payable { 22         require( 23             storedData[msg.sender] != 0    24             msg.value &gt;= ticketCost 25         ); 26         require(!numberIsTaken[x]); 27         require(x &gt; rangeMin &amp;&amp; x &lt; rangeMax); 28 29         if (numberIsTaken[storedData[msg.sender]]) { 30             numberIsTaken[storedData[msg.sender]] = false; 31         } 32 33         storedData[msg.sender] = x; 34         numberIsTaken[x] = true; 35     } 36 37     function withdrawal() public { 38         msg.sender.transfer(address(this).balance); 39     } </pre>	<pre> 1  pragma solidity ^0.5.0; 2 3  contract SimpleStorage { 4      mapping(address =&gt; uint) public storedData; 5      mapping(uint =&gt; bool) public numberIsTaken; 6      uint public rangeMin; 7      uint public rangeMax; 8      uint public ticketCost; 9 +     address public owner; 10 11     constructor( 12         uint _ticketCost, 13         uint _rangeMin, 14         uint _rangeMax 15     ) public { 16         ticketCost = _ticketCost; 17 18         rangeMin = _rangeMin; 19         rangeMax = _rangeMax; 20 + 21 +         owner = msg.sender; 22     } 23 24     function set(uint x) public payable { 25         require( 26             storedData[msg.sender] != 0    27             msg.value &gt;= ticketCost 28         ); 29         require(!numberIsTaken[x]); 30         require(x &gt; rangeMin &amp;&amp; x &lt; rangeMax); 31 32         if (numberIsTaken[storedData[msg.sender]]) { 33             numberIsTaken[storedData[msg.sender]] = false; 34         } 35 36         storedData[msg.sender] = x; 37         numberIsTaken[x] = true; 38     } 39 40     function withdrawal() public { 41 +         require(msg.sender == owner); 42         msg.sender.transfer(address(this).balance); 43     } </pre>
--	--



*Figure 33: Updated withdrawal function*

Another way of doing this could be to allow anyone to call the withdrawal function, but instead fix the transfer function to send to the “owner” only. The contract then remains secure, since only the owner can receive the funds, but someone else may be kind enough to withdraw them for the “owner”.

## Challenges

This smart contract is still incomplete. Features around a deadline, for example, would be useful. Please try and set a closing time in the instructor and enforce that no one can set a value after that deadline. These docs may be useful: [solidity.readthedocs.io/en/v0.5.3/units-and-global-variables.html#time-units](https://solidity.readthedocs.io/en/v0.5.3/units-and-global-variables.html#time-units)

Another thing which you can try to implement as a challenge is to make the code return the user’s funds if not enough tickets for the lottery are bought. In other words, conditions for what happens if the lottery fails.

One last thing that is important, since the compiler for the code is always evolving, is the Pragma version. This is stated at the top of the code as the version of the Solidity compiler to use, since sometimes different versions of the Solidity compiler differ in significant ways, giving strange or unexpected results. In smart contracts, we want to avoid the possibility of unexpected behavior at all costs, so it is worth your while to be pedantic about this.



## Bibliography

“bkrem/awesome-solidity,” *GitHub*. [Online]. Available: <https://github.com/bkrem/awesome-solidity#editor-plugins>.

“Ethereum Stack Exchange,” *Ethereum Stack Exchange*. [Online]. Available: <https://ethereum.stackexchange.com/>

“MetaMask Ethereum Browser Extension,” *MetaMask*. [Online]. Available: <https://metamask.io/>

“Node Version Manager - POSIX-compliant bash script to manage multiple active node.js versions,” *GitHub*. [Online]. Available: <https://github.com/creationix/nvm>.

“Private Key ProviderEngine for Ethereum,” *Gist*. [Online]. Available: <https://gist.github.com/JasoonS/11fca1a98f1eb41a79987a041541e8c8>.

“remixd,” *npm*. [Online]. Available: <https://www.npmjs.com/package/remixd>

“Rinkeby Authenticated Faucet,” *Rinkeby*. [Online]. Available: <https://faucet.rinkeby.io/>.

“trufflesuite/truffle-hdwallet-provider,” *GitHub*. [Online]. Available: <https://github.com/trufflesuite/truffle-hdwallet-provider/blob/master/index.js>.

Ethereum Revision, “Units and Globally Available Variables” *Solidity v0.5.3 documentation*. [Online]. Available: <https://solidity.readthedocs.io/en/v0.5.3/units-and-global-variables.html#time-units>.

Ethereum, “ethereum/remixd,” *GitHub*, [Online]. Available: <https://github.com/ethereum/remixd>.

Etherscan, *Rinkeby Testnet Explorer* [Online]. Available: <https://rinkeby.etherscan.io/>

Node.js Foundation, *Node.js* [Online]. Available: <https://nodejs.org/en/>.

*Remix - Ethereum IDE* [Online]. Available: <https://remix.ethereum.org/>

Truffle Blockchain Group, 2019 “React,” *Truffle Suite*. [Online]. Available: <https://truffleframework.com/boxes/react>

Truffle Blockchain Group, 2019 “Sweet Tools for Smart Contracts,” *Truffle Suite*. [Online]. Available: <https://truffleframework.com/>



Truffle Blockchain Group, 2019 “Truffle | Configuration | Documentation,” *Truffle Suite*. [Online]. Available: <http://truffleframework.com/docs/advanced/configuration>

