Resource List @dapp university_youtube

Developer Tools

Here is a list of developer tools, including frameworks, IDEs, and libraries that can be used to start creating blockchain applications (dapps) with Ethereum Smart Contracts:

- Frameworks
    - Truffle Framework (Website)
    - Truffle Framework(Github)
    - Embark Framework (Website)
    - Embark Framework (Github)
    - Dapp
    - Populus
    - Etherlime
    - Cliquebait
- Smart Contract Languages
    - Solidity
    - Vyper
    - Bamboo
    - LLL
- IDEs & Editor Plugins
    - Remix IDE
    - Eth Fiddle
    - Atom Solidity Linter
    - Etheratom
    - Pragma autocomplete-solidity package
    - Pragma language-solidity

    - Superblocks Studio
    - Vim solidity
    - Studio Code Solidity
    - Intellij Solidity Plugin
    - YAKINDU Solidity Tools
- Test Blockchains
    - Ganache
    - Kaleido
    - Local Raiden
    - Private networks deployment scripts
    - Local Ethereum Network
- Test Ether Faucets
    - Rinkeby faucet
    - Kovan faucet
    - Ropsten faucet
- Ethereum Clients
    - Geth
    - Parity
    - Cpp-ethereum
    - Pyethapp
    - Trinity
    - Ethereumjs
    - Ethereumj
    - Harmony
    - Seth

    - Mustekala
    - Exthereum
    - EWF Parity
    - Quorum
- JavaScript APIs For Communicating With Ethereum
    - Web3.js
    - Eth.js
    - Ethers.js
    - Web3Wrapper
    - Ethereumjs
- Backend Ethereum APIs
    - Web3.py
    - Web3.php
    - Web3j
    - Nethereum
    - Ethereum.rb
    - Web3.hs
    - KEthereum
    - Pyethereum
    - Eventeum
- Bootstrapping Tools
    - Truffle boxes
    - Drizzle by Truffle
    - Local Raiden
    - Private networks deployment scripts

    - Local Ethereum Network
    - Kaleido
    - Cheshire
    - Subproviders
    - web3-webpacked
    - Vortex
- Smart Contract Libraries
    - OpenZeppelin
    - cryptofin-solidity
    - Modular Libraries
    - DateTime Library
    - Aragon - DAO protocol
    - 0x - DEX protocol
    - Token Libraries
- Storage
    - IPFS
    - IPFS-Store
    - OrbitDB
    - JS IPFS API
    - Swarm
- Messaging

    Whisper

    Pydevp2p

Ecosystem Tools

Here is a list of ecosystem tools that will help you as a blockchain developer:

- Ethereum Wallets
    - Metamask
    - Gnosis multisig wallet
    - Mist
    - Exodus
- Web Wallets
    - MyEtherWallet
    - MyCrypto
    - Portis
    - Eth lightwallet
    - SpankCard
    - Mnemonic generator
- Mobile Wallets
    - Trust
    - Coinbase Wallet
    - Cipher
    - Status
    - imToken
    - Jaxx
    - WallETH
    - eth-wallet-light
    - Hardware Wallets
        - Trezor

        - Ledger
        - KeepKey
    - Block Explorers
        - Etherscan
        - Etherchain Light
        - POA Explorer
        - QuickBlocks
        - Supermax
        - Alethio EthStats 2.0
    - Gas Price Tools
        - EthGasStation
        - Petrometer
        - CryptoProf

Tutorials/Reference

Here is a list of tutorials and references that will help you get started developing on the blockchain fast:

- The Ultimate Ethereum Dapp Tutorial
- Code Your Own Cryptocurrency on Ethereum
- Intro to Web3.JS
- Build a Dapp in 20 Minutes
- Build a Dapp with IPFS
- Ethereum in 25 Minutes
- Ethereum's Account Model
- Ethereum for Web Developers
- Ethereum Wallets

- The Hitchhiker's Guide to Smart Contracts
- Ethereum Petshop
- CryptoZombies
- Smart Contract Best Practices
- The DAO Hack
- Parity Wallet Hack
- Parity Wallet Hack II
- Ethernaut - Smart Contract Hacking Game
- Hack This Contract Game
- Public/Private Key Cryptography
- RSA
- ECDSA
- Cryptographic Hash Function
- Commitment Schemes
- Merkle Trees
- Merkle Proofs

Ethereum Official

- Ethereum Website
- Ethereum Whitepaper
- Ethereum Yellow Paper
- Ethereum Wiki
- Ethereum Improvement Proposals
- Solidity Documentation
- Web3.js Documentation

Blogs

Here is a list of blogs that will help sharpen your skills as you learn blockchain development:

- Vitalik Buterin's Blog
- OpenZeppelin
- ConsenSys
- BlockChannel
- Epicenter
- Coindesk
- ETH Research
- Hacking, Distributed
- Unenumerated
- Chris Burniske
- Great Wall of Numbers

Podcasts

Here are some podcasts that you will find helpful as a blockchain developer:

- Dapp University
- BlockChannel
- Unchained Podcast
- Conspiratus Podcast

Dapp Discovery Tools

Here is a list of websites that track blockchain applications (dapps). They will help you discover new dapps and reveal insights about each dapp's usage

- Dapp Radar
- Dapp Volume
- Dapp.com

- State of the Dapps

Books

- Mastering Ethereum
- Building Ethereum Dapps
- Introducing Ethereum and Solidity
- Ethereum Development with Go
- Cryptoeconomics Study
- The business Blockchain

Discussion Forums

Here is a list of discussion forums you can visit to participate in discussion around blockchain development:

- The Ethreum Community Forum
- Reddit - r/ethdev
- Reddit - r/ethereum
- Reddit - r/ethdapps
- r/CryptoCurrency
- Bitcoin Talk
- CryptoPanic News Feed

Newsletters

Here is a list of newsletters and mailers that will keep you up to date with what's happening in the blockchain space:

- Week in Ethereum

Events & Conferences

Here is a list of events and conferences you can attend to learn first-hand from the top blockchain developers and network with other developers:

- DevCon
- TruffleCon
- DappCon
- EthDenver
- EthBerlin
- EthMemphis
- BuildEth
- Hackathon.com for Blockchain

Getting a Job

Here are some great resources that will help you get a job as a blockchain programmer:

- How to Get a Job at a Crypto Startup
- Ethereum Jobs
- AngelList Crypto Startups
- Who's Hiring? (r/ethdev)
- BlockchainJobz
- Be in Crypto
- Blockchain Job Board

All Your Questions Answered

I've compiled a list of questions that I get a lot from brand new blockchain developers. I've answered all of them below. Feel free to reference the table of contents to skip around to questions that interest you.

What Kind of Blockchain Developer?

If you want to become a blockchain developer, you must first figure out what kind of blockchain developer you want to be. I assume that you want to become the kind of developer that builds user-facing applications that are powered by the blockchain. These are called decentralized applications or

"dApps", and they allow businesses and users to leverage the underlying technology of the blockchain.

A blockchain application developer is different from a core blockchain developer, which might work on a blockchain protocol like Bitcoin or Ethereum. Core developers work to improve upon the technologies themselves, like improving Ethereum's consensus algorithm.

This is similar to the choice you must make if you wanted to become a web developer. You would need to decide whether you wanted to build websites and web applications, or if you wanted to work on web protocols like HTTP, improve browsers, etc... Most people who want to become web developers want to build websites or web applications. Similarly, most people who want to become blockchain developers want to build blockchain based applications. I'm going to give you everything you need to know in this article to get started doing just that!

Which Blockchain?

If you're going to build applications that are powered by the blockchain, you need to pick a blockchain! But there are so many choices out there, where do you begin? I like to build blockchain applications on Ethereum because it uses smart contracts which allow developers to write advanced programs on the blockchain (more advanced than Bitcoin). These smart contracts are the building blocks of decentralized blockchain applications. Here are a few more reasons why I choose to build decentralized blockchain applications on Ethereum over other platforms:

- **Great Developer Tools**
- **Fast Growing Developer Community**
- **Wealth of Knowledge Resources**
- **Booming Dapp Ecosystem**
- **Easy to Create Tokens**

What Can You Build On The Blockchain?

Once you've decided to start building decentralized blockchain applications, you might ask, "what can I build on the blockchain?" Good news! I can show you several examples, as I have created several FREE in depth step-by-step tutorials that show you how to get started. Keep reading to find out.

Build Your First Decentralized Application

What's the best way to get started learning to build blockchain applications? Learn by doing! I've created a 2-hour step-by-step video tutorial where I teach you to build your first decentralized blockchain application. You can watch the full video below, and follow the step-by-step walk through with code examples here.

In this tutorial, I show you how to write your first Ethereum smart contract, that holds an election between two candidates. I show you how to write tests against the smart contract, deploy it to the Ethereum blockchain, and develop a client-side application that allows accounts connected to the network to cast votes. I also explore key concepts like "what is a blockchain?", "what is a smart contract?", and "how does a dApp work?"

Code Your Own Cryptocurrency

One of the most popular use cases on Ethereum is to build your own cryptocurrency, which I show you how to do in the 8-hour video tutorial below. I also have a full-length step-by-step article to accompany that video here.

Ethereum is unique because it allows you to create your own cryptocurrency without creating a new blockchain. It allows you to create a token with smart contract code that gets deployed to the blockchain. Ethereum has a standard for creating tokens called "ERC-20", and tokens that implement this standard are called "ERC-20 Tokens". ERC-20 tokens can be transferred from one account to another as payment, and purchased/sold on cryptocurrency exchanges, just like any other cryptocurrency. I explain more how ERC-20 tokens work in the video below.

ERC-20 is simply a standard that specifies how these tokens behave, so that they are compatible with other platforms like cryptocurrency exchanges. Since these tokens are smart contracts, we can use this standard to determine the token's attributes. For example you could create a token called “My Token” with the symbol “MTK” and that there will be 100,000,000 of these tokens in existence. The token smart contract keeps track of the basic token attributes like the name, “My Token", the symbol that you see on a cryptocurrency exchange, and how many total tokens exist. It also keeps track of who owns “My Token” and how much.

You can also sell your Ethereum based cryptocurrency in an initial coin offering or ICO. This is also called a crowdsale. Crowdsales are a way for a company to raise funds for their business by creating their own ERC-20 token that can be purchased with Ether, Ethereum's native cryptocurrency. Whenever a crowdsale takes place, the company gets liquid capital in the form of Ether which was paid by the investors in exchange for the token. You

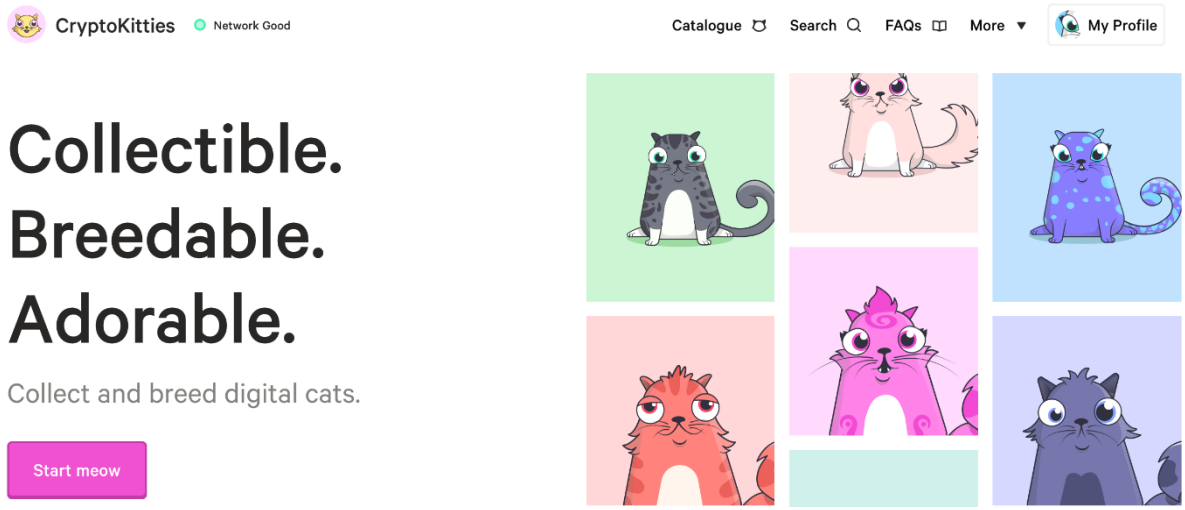
can watch the video below to learn more about how a crowdsale works on the Ethereum blockchain.

In order to participate in a crowd sale, an investor must connect to the Etherum blockchain with an account. This account has a wallet address that can store Ether, as well as the ERC-20 tokens that are purchased in the crowd sale. The investor must visit a crowd sale website that talks to a smart contract. The smart contract governs all of the rules for how the crowd sale works. Whenever an investor purchases tokens on the crowd sale website, they send Ether from their wallet to the smart contract, and the smart contract instantly dispenses the purchased tokens to their wallet. The smart contract sets the price of the token in the crowd sale and governs how the crowd sale behaves.

Crowdsales can take on all kinds of shapes and sizes. They can have multiple tiers or phases, like Pre ICO, ICO, and ICO Bonus phase. Each of these tiers can happen at different points of time and can behave differently. They can also have white lists to restrict which investors can purchase tokens. They can also have a reserved amount of tokens that are not sold in the crowd sale. These reserves are usually set aside for specific members of each company like founders and advisors. These reserves can be a fixed amount of tokens or a percentage. Whenever a crowd sale ends, it can be finalized by an administrator. Whenever this happens, all of the reserved tokens will be distributed to the appropriate accounts and the crowd sale will officially be over.

Blockchain Games

Games are one of the most popular use cases on the Ethereum blockchain. You might have heard of the blockchain game CryptoKitties, which was famous for slowing down the Ethereum network because of high transaction volume in December 2017.



CryptoKitties is a game based upon Ethereum that lets you collect and breed digital cats. It does this with an Ethereum token standard called ERC-721, which allows you to create Ethereum tokens with a unique identity. CryptoKitties uses this technology to create unique kitty tokens that have cute kitty graphics associated with each token.

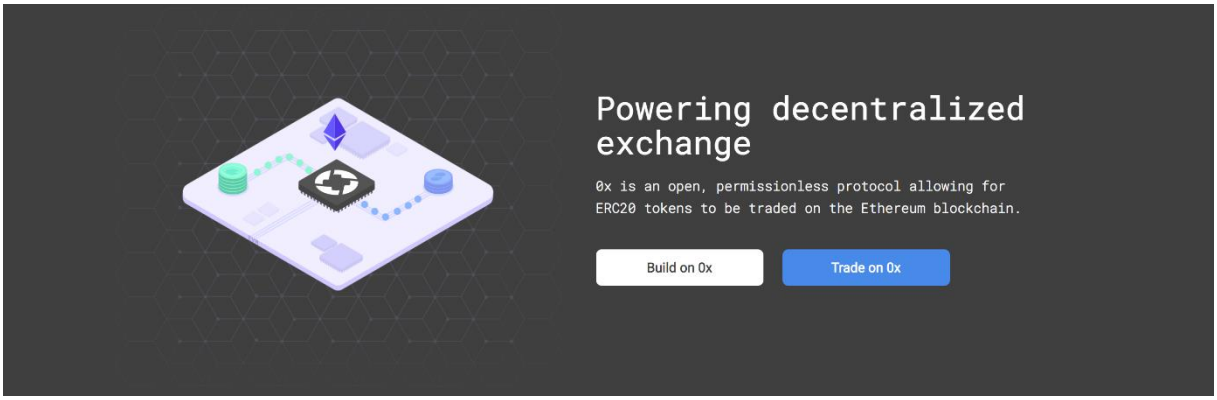
These tokens are called non-fungible tokens, which means that each token has a unique value, and cannot simply be replaced by another token. This is different from the ERC-20 token standard that I mentioned in the "Code Your Own Cryptocurrency" section. ERC-20 tokens are used for payment because they can be swapped for any other ERC-20 token of the same value. They're completely replaceable, or fungible. ERC-721 tokens, like CryptoKitties, are not necessarily replaceable with any other ERC-721 token. Each individual token's value is appraised on a case-by-case basis, not in relation to the whole set. This is kind of like a collectible baseball card.

Blockchain Applications (Dapps)

One of the most exciting things about the booming blockchain technology ecosystem is that many companies are trying to solve challenging problems right now. That's also one of the most exciting reasons to learn blockchain programming: to figure out what kinds of problems that you can solve as a developer, and to find new ways to provide value to lots of potential customers. There are several other leading blockchain applications that are exploring what's possible with the Ethereum network. Here are a few that you should know about.

ETHLend is a decentralized lending service that issues loans on the blockchain with secure, peer-to-peer lending with smart contracts. It removes banks and financial institutions from the traditional loan process by allowing borrowers and lenders to set the terms of the loan. It also allows these parties to bypass the middleman by trading with one another directly. Essentially, the lender and the borrower can create loan contracts from anywhere in the world on their own terms.

ETHLend provides value that other centralized solutions don't. For example, it solves the issue of trust because the decentralized platform removes the need for the lender and the borrow to trust one another. It's transparent because the Ethereum blockchain has a public ledger where anyone can verify the validity of transactions. It's also accessible because by using the Ethereum network, lenders and borrowers can make transactions unconstrained form anywhere in the world. All they need is an account address!

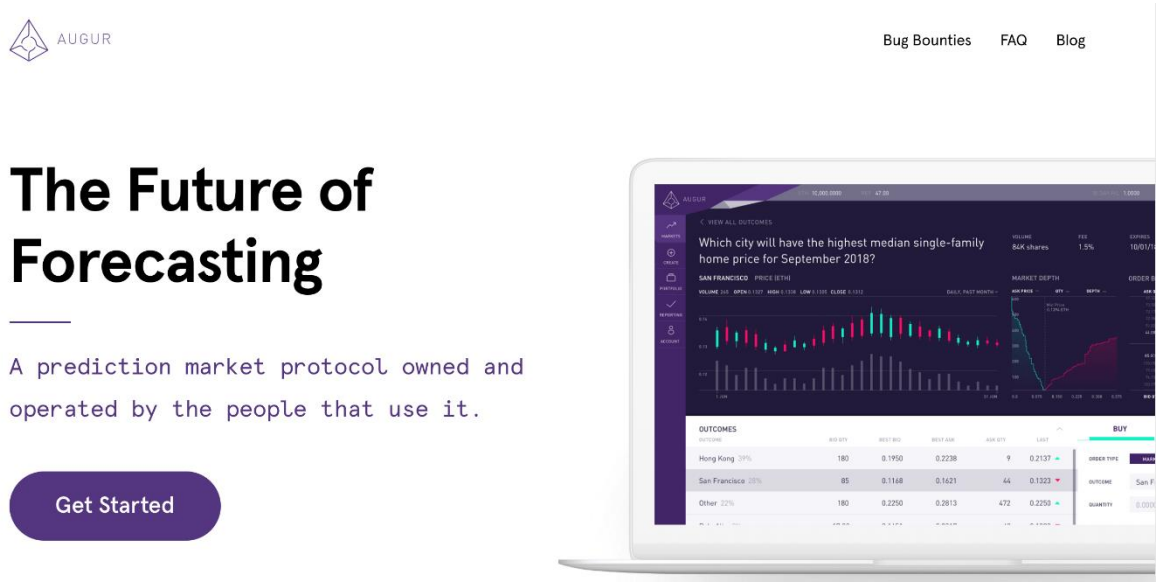


0x is a protocol that facilitates decentralized exchange of ERC-20 tokens on the Etherem blockchain. It brings all of the benefits of the Ethereum blockchain to process of exchanging cryptocurrency, like trust, transparency, and open access.

If you've ever purchased cryptocurrency before, you might have used a website like Coinbase. This is a popular exchange, but it is **centralized**. All of the code and the data run on a central server. If you leave cryptocurrency on

Coinbase, and the servers are compromised in any way, then hackers can potentially gain access to your cryptocurrency and steal funds.

A decentralized exchange is different. It allows you to trade Ethereum tokens directly with other parties with smart contracts without leaving funds on a centralized exchange. This is a huge boost for security when you're trading cryptocurrency because the funds aren't required to leave your wallet until the orders are fulfilled. The funds are transferred directly between the buyer and the seller without risk of leaving them on a server in the meantime.



Augur is a decentralized prediction market that crowd sources data in order to forecast outcomes of specific events. By doing this, data that is collected by the crowd is calculated to produce the most realistic possibility, and therefore the most probable, outcome of the event. Whenever a prediction is accurate, it is rewarded by the network. Whenever it is wrong, it is penalized. It uses this incentive structure to foster the most accurate reporting possible.

Augur's main goal is to revolutionize prediction markets in order to change the way that people verify truthful information on the web. It does this by verifying data from large groups of people, rather than having to trust a small group of centralized "experts".

Augur uses its own tradeable token called Reputation (REP) in order to give users the ability to report or weigh in on the outcome of events. Users can earn more tokens when they provide truthful data. Users can also earn tokens by submitting correct predictions. Anyone who reports against the correct prediction untruthfully will lose their tokens and lose out on the possible winnings.

CryptoZomibes

Cryptozombies is a really fun way to start programming in Solidity, the language used to write smart contracts on Ethereum. It is a "gamified" experience similar to other educational programming websites like Code Academy or Code School. It gives you an accessible overview of the programming language with small lessons with code challenges that you can complete in your browser. You don't even have to set up any development tools locally. Throughout these lessons, you create a blockchain collectible game, much like Cryptokitties.

The first lesson in the series has 15 chapters. You can aim to complete it in about one hour. The website contains a code editor where you can complete the challenges at the end of each chapter. Once you complete it, you can check your answer against the solution to see if you implemented it correctly. If not, you'll have the opportunity to fix your code before moving on to the next chapter.

If you're new to programming, I would recommend acquiring some programming experience before trying CryptoZombies. If you want to get started programming, you can try one of the sites I mentioned before, like Code Academy or Code School. I also recommend Free Code Camp for beginners!

Which Programming Languages Do I Learn?

One of the most obvious questions to ask when learning blockchain programming is, "what programming languages should I learn"? Keep reading as I answer those questions in this section!

Solidity

Solidity is the main programming language used when writing decentralized applications on the Ethereum blockchain. It allows you to create smart contracts that get compiled and run on the Ethereum virtual machine, which executes the contracts on the decentralized network without requiring centralized authorities or trusted parties.

Solidity is a statically typed, contract oriented programming language that looks a lot like JavaScript. So if you come from a web development background, Solidity should look somewhat familiar to you. It is a Turing Complete programming language, which means that it has all the major features of other useful programming languages like data structures, loops, variables, functions, etc...

Similar to classes in object oriented programming, Ethereum uses smart contracts to encapsulate all the code and data that gets run on Ethereum. Smart contracts are the building blocks of decentralized applications. Just like classes is OOP, each smart contract contains state variables, functions, and common data structures.

Because all of the code on the Ethereum blockchain is immutable (it cannot change), this means that you must be very careful when writing code that gets deployed to production. You must adopt a different programming paradigm that prioritizes caution end security. You cannot rush code out to production because the cost of bugs could be very high. You cannot simply "fix" a bug inside a smart contract because the code cannot be changed. Aside from advanced strategies for upgrading smart contracts (which always come with compromise), the only thing you can do to upgrade a smart contract is deploy a new copy of it. When you do this, however, all of the contract's state, potentially containing user data, will be lost forever.

Let's take a look at some examples of smart contracts. Here is the election smart contract that I teach you to build step-by-step in my 2 hour dapp tutorial:

```
pragma solidity ^0.4.2;

contract Election {
    // Model a Candidate
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }

    // Store accounts that have voted
    mapping(address => bool) public voters;
    // Read/write candidates
    mapping(uint => Candidate) public candidates;
    // Store Candidates Count
    uint public candidatesCount;

    function Election () public {
        addCandidate("Candidate 1");
        addCandidate("Candidate 2");
    }
```

```
    function addCandidate (string _name) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(candidatesCount, _name,
0);
    }

    function vote (uint _candidateId) public {
        // require that they haven't voted before
        require(!voters[msg.sender]);

        // require a valid candidate
        require(_candidateId > 0 && _candidateId <= candidatesCount);

        // record that voter has voted
        voters[msg.sender] = true;

        // update candidate vote Count
        candidates[_candidateId].voteCount ++;
    }
}
```

Let me explain some of this code. It starts by declaring the solidity version with the `pragma solidity` statement. Next, it declares the smart contract with the "contract" keyword, followed by the contract name. Then, it declares a state variable that will store the value of the candidate name. State variables allow us to write data to the blockchain. The variable is declared as a string, and its visibility has been set to `public` which will allow external consumers to read its value.

Next the contract has a constructor function that will get called whenever it is deployed to the blockchain. This is where it will set the value of the candidate state variable that will get stored to the blockchain upon migration. Notice that the constructor function has the same name as the smart contract. This is how Solidity knows that the function is a constructor.

Next, you can see the `vote` function. The core functionality of this function is to increase the candidate's vote count by reading the Candidate struct out of the "candidates" mapping and increasing the "voteCount" by 1 with the increment operator (++). Let's look at a few other things that it does:

97. It accepts one argument. This is an unsigned integer with the candidate's id.
98. Its visibility is public because we want an external account to call it.
99. It adds the account that voted to the voters mapping that we just created. This will allow us to keep track that the voter has voted in the election. We access the account that's calling this function with the global variable "msg.sender" provided by Solidity.
100. It implements require statements that will stop execution if the conditions are not met. First require that the voter hasn't voted before. We do this by reading the account address with "msg.sender" from the

mapping. If it's there, the account has already voted. Next, it requires that the candidate id is valid. The candidate id must be greater than zero and less than or equal to the total candidate count.

That's a quick overview of how that smart contract works! Hopefully that gives you a taste of the Solidity programming language. Again, you can follow along step-by-step as I show you how to build that smart contract in my 2 hour dapp tutorial.

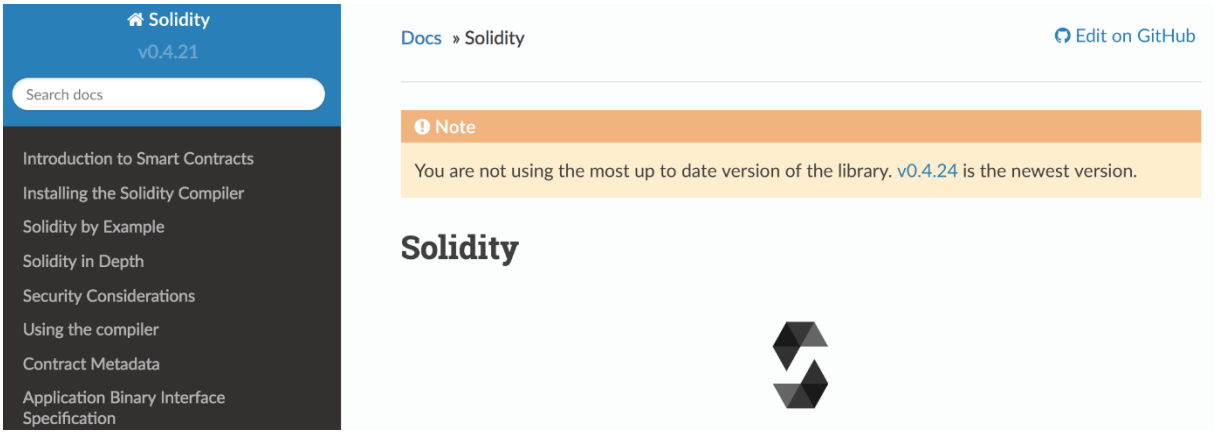
Let's look at the ERC-20 token smart contract that I teach you to build in my 8-hour tutorial where I teach you to code your own cryptocurrency on Ethereum:

```
pragma solidity ^0.4.2;

contract DappToken {
    string  public name = "DApp Token";
    string  public symbol = "DAPP";
    string  public standard = "DApp Token v1.0";
    uint256 public totalSupply;

    event Transfer(
        address indexed _from,
        address indexed _to,
        uint256 _value
    );

    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint256 _value
    );

    mapping(address => uint256) public balanceOf;
    mapping(address => mapping(address => uint256)) public allowance;

    function DappToken (uint256 _initialSupply) public {
        balanceOf[msg.sender] = _initialSupply;
        totalSupply = _initialSupply;
    }

    function transfer(address _to, uint256 _value) public returns (bool
success) {
        require(balanceOf[msg.sender] >= _value);

        balanceOf[msg.sender] -= _value;
        balanceOf[_to] += _value;

        Transfer(msg.sender, _to, _value);

        return true;
    }

    function approve(address _spender, uint256 _value) public returns
(bool success) {
```

```
        allowance[msg.sender][_spender] = _value;

        Approval(msg.sender, _spender, _value);

        return true;
    }

    function transferFrom(address _from, address _to, uint256 _value)
public returns (bool success) {
        require(_value <= balanceOf[_from]);
        require(_value <= allowance[_from][msg.sender]);

        balanceOf[_from] -= _value;
        balanceOf[_to] += _value;

        allowance[_from][msg.sender] -= _value;

        Transfer(_from, _to, _value);

        return true;
    }
}
```

Let's take a look at what this smart contract does, and how it implements the ERC-20 standard:

- It stores the token name `string public name = "DApp Token"`.
- It stores the token symbol for cryptocurrency exchanges `string public symbol = "DAPP"`.
- It stores the total supply of tokens in existence `uint256 public totalSupply`.
- It uses a Solidity mapping to store the balance of each account that owns tokens `mapping(address => uint256) public balanceOf`.
- It implements a `transfer` function to allow users to send tokens to another account.
- It implements an `approve` function that allows another account to spend tokens, like on a cryptocurrency exchange. This updates the `allowance`mapping to see how much the account is allowed to spend.
- It implements a `transferFrom` that allows another account to transfer tokens.



While you're learning Solidity, it is very helpful to reference the documentation which can be found here.

JavaScript

In addition to learning the Solidity programming language, it is essential to learn JavaScript to become a blockchain developer. If you come from a web development background, great! You probably already have some familiarity with JavaScript. If you don't, do not worry. You can learn the basics of JavaScript fairly quickly, which will give you enough to become productive when developing for the blockchain.

Let's take a look at how JavaScript is used when developing decentralized blockchain applications.

1. Smart Contract Testing

JavaScript is the main language used to test Ethereum smart contracts. These tests are written to simulate client-side interactions with the smart contract. When running tests, the smart contracts are compiled and deployed to a JavaScript runtime environment, and test are executed against JavaScript representations of the compiled smart contracts. Most commonly used is the the Mocha testing framework and the Chai assertion library when writing tests. With both of these tools, we can craft robust tests to ensure that our smart contract code is production ready.

Here is an example of the kinds of the tests from the smart contracts that I teach you to build in my 2 hour dapp tutorial:

```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;

  it("initializes with two candidates", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
```

```
    }).then(function(count) {
      assert.equal(count, 2);
    });
  });

  it("it initializes the candidates with the correct values", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      return electionInstance.candidates(1);
    }).then(function(candidate) {
      assert.equal(candidate[0], 1, "contains the correct id");
      assert.equal(candidate[1], "Candidate 1", "contains the correct
name");
      assert.equal(candidate[2], 0, "contains the correct votes count");
      return electionInstance.candidates(2);
    }).then(function(candidate) {
      assert.equal(candidate[0], 2, "contains the correct id");
      assert.equal(candidate[1], "Candidate 2", "contains the correct
name");
      assert.equal(candidate[2], 0, "contains the correct votes count");
    });
  });
});
```

You can see that this basic test checks that the smart contract was deployed and initialized correctly. It asserts that the election started with two candidates, and that each candidate has the correct `name`, `id`, and `vote count`. The testing syntax is straight-forward, readable, and adds high value by ensuring that the smart contract behaves the way that we expect. Testing smart contracts is so vital because once the smart contract code is deployed to the Ethereum blockchain, it cannot be changed!

2. Talking to The Blockchain

Ethereum allows you to talk to its blockchain by communicating directly with an Ethereum node, on behalf of the entire network. It does this with something called the Web3 library. The most popular implementation of this library is Web3.js.

Essentially, Web3.Js is the main JavaScript library for interacting with the Etherum blockchain because it allows you to read data from it, and write data to it. It allows you to query data about the blockchain, like reading data the blocks themselves and inspecting transactions on Ethereum. It also allows you to read data from smart contacts. It also allows you to write data to the blockchain by sending transactions, creating smart contracts, writing data to smart contracts, and calling smart contract functions.

I've created an 8-video tutorial series that shows you how to get started with Web3.js. You can start with the first video above, and also follow along with the accompanying article here.

3. Blockchain Applications

As I've already mentioned, smart contracts are the building blocks of decentralized applications. Once you've created a smart contract, you have effectively created a decentralized application that can be interacted with through any interface. You could call its functions through a variety of 77methods, but most end users want a client-side interface that allows them to use the smart contracts you've created. The easiest way to do this is to create client-side web applications that communicate directly with your smart contracts. These applications are written in JavaScript.

The best way to turn a client-side web application into a blockchain-enabled web application is to use the Web3.js library I mentioned previously. This will allow your JavaScript application to communicate with Ethereum smart contracts.

In addition to knowing JavaScript, it's essential to know a modern JavaScript web development framework. If you don't have one already, I highly recommend using React JS, as it is widely used among the web development community as well as the Ethereum blockchain development community.

/* other resources

https://medium.com/@Francesco_AI/the-convergence-of-ai-and-blockchain-whats-the-deal-60c618e3accc
https://www.weforum.org/agenda/2017/06/3-ways-blockchain-can-accelerate-financial-inclusion/
https://bravenewcoin.com/news/blockchains-for-artificial-intelligence/
https://accubits.com/white_papers/possibilities-integrating-ai-blockchain/
https://medium.com/@prysmeconomics/blockchain-ai-10-outcomes-119d95532a99
https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/
https://news.crowdvalley.com/news/state-of-blockchain-and-artificial-intelligence-ai-in-fintech