

(https:

ckgee  
ks.co  
m/log  
in/)



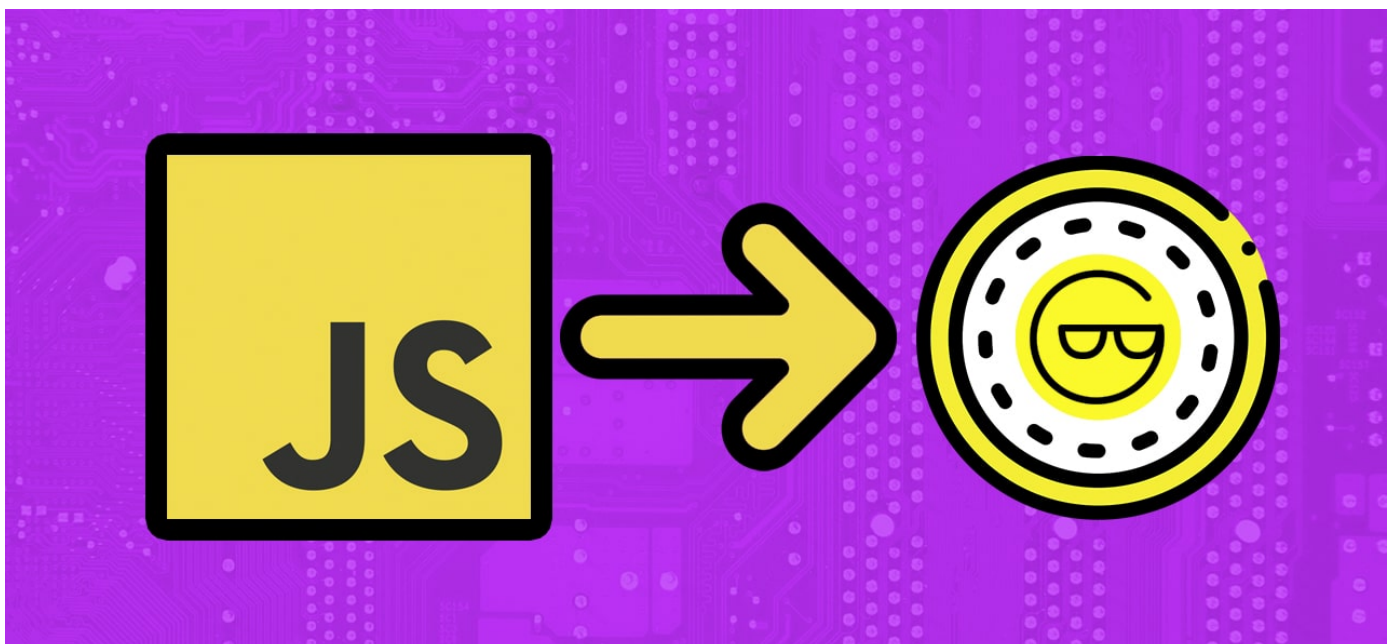
## Blockchain Tutorial | How To Become A Blockchain Developer:

[#Blockchain 101 \(/guides/?tagfilter=true&filter=blockchain-101\)](#)

[#Ethereum \(/guides/?tagfilter=true&filter=ethereum\)](#) [#Hash \(/guides/?tagfilter=true&filter=hash\)](#)

0:00 / 22:00

In this guide, we are going to map out your journey for becoming a Blockchain Developer. To ease things up for you, we are going to divide this guide into various milestones and give you action steps at the end of every section.



How to become a Blockchain Developer?

# HOW TO become a **Blockchain Developer**!

It goes without saying that we are living in the “era of the blockchain (<https://blockgeeks.com/guides/what-is-blockchain-technology/>)”. The impact that it can have on our future is truly scary and magnanimous. So, how can you get a piece of that “blockchain action”? If you are to become a blockchain developer, then there are certain steps that you need to take.

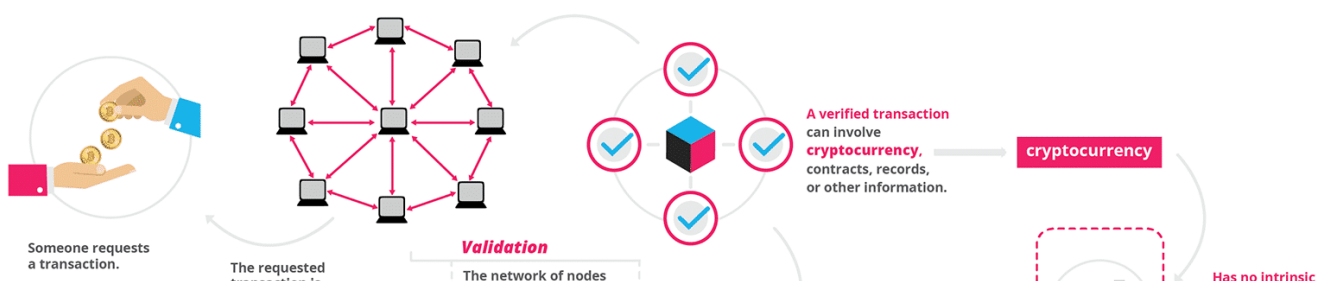
Hopefully, at the end of the guide, you will have the tools required to kick-start your journey. If you are serious about becoming a developer then we need to set some expectations for you. Firstly, it is going to take time and you will need to dedicate your time and resources to your education (you can continue your education with us by taking our [online classes](https://blockgeeks.com/blockchain-courses/) (<https://blockgeeks.com/blockchain-courses/>)). Secondly, do not expect immediate results, becoming a blockchain developer is not a magic pill.

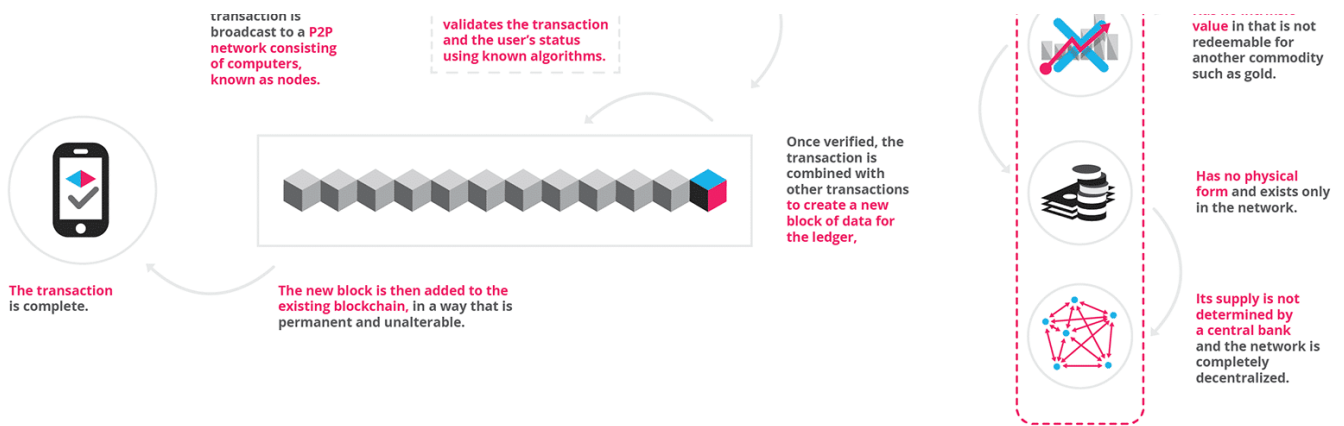
So, having said that, let's start your journey.

## How to Become a Blockchain Developer



## How To Become A Blockchain Developer Tutorial!





## Milestone #1: Understanding The Basics

One of the biggest hurdles with anything as new and revolutionary, such as the blockchain technology, is familiarizing oneself with various concepts integral to the system.

If you are a beginner, then there are certain terms that you need to be familiar with:

Learn Blockchain Technology

- **Blockchain:** The blockchain (<https://blockgeeks.com/guides/what-is-blockchain-technology/>) is a chain of blocks where each block contains data of value without any central supervision. It is cryptographically secure and immutable.
- **Decentralized:** Blockchain is said to be decentralized because there is no central authority supervising anything.
- **Consensus Mechanism:** The mechanism by which a decentralized network comes to a consensus (<https://blockgeeks.com/guides/what-is-cryptoeconomics/>) on certain matters.
- **Miners:** Users who use their computational power to mine for blocks.

Train to Become A Blockchain Developer

Start Your Free Trial Today! (<https://courses.blockgeeks.com/enrollment/>)

It could be advisable to learn more about these terms that are widely used in the crypto-sphere. It is highly recommended that you [go through our comprehensive glossary](#)

(<https://blockgeeks.com/guides/blockchain-glossary-from-a-to-z/>)

(<https://blockgeeks.com/guides/blockchain-glossary-from-a-z/>). It is important to learn these basic terms otherwise you will be very lost further on in your education. Now, up next, it is time to educate yourself some more on the technical aspects of the blockchain.

If you are interested in the technical aspects of how to create a fin-tech application on top of the Blockchain then you should definitely learn the ins and outs of [cryptoeconomics](https://blockgeeks.com/guides/what-is-cryptoeconomics/) (<https://blockgeeks.com/guides/what-is-cryptoeconomics/>). Most developers are usually well-versed in the “crypto” part of the equation but their knowledge of the “economics” part is extremely lacking.

This difference in knowledge is extremely apparent when you study some of these [ICOs](https://blockgeeks.com/guides/initial-coin-offering/) (<https://blockgeeks.com/guides/initial-coin-offering/>) floating around. It is very obvious to see that the economics side of their ICO’s are not well thought out.

So, in light of that, it can be a good idea to read up a bit on economics and have a general idea of it. If you want to learn about cryptoeconomics in general, then you [may checkout our article here](https://blockgeeks.com/guides/what-is-cryptoeconomics/) (<https://blockgeeks.com/guides/what-is-cryptoeconomics/>).

If you are intrigued by the cryptography specifically and want to know how signatures work and what public key cryptography means, [then read this](https://blockgeeks.com/guides/cryptocurrencies-cryptography/). (<https://blockgeeks.com/guides/cryptocurrencies-cryptography/>)

After that, it is highly recommended that you understand how bitcoin works. Bitcoin is the most widespread, finest and one of the more elegant applications of the blockchain technology. You can even call it the finest example of what the blockchain technology can achieve purely because of the impact that it has had.

So, it is advisable that you read Satoshi Nakamoto’s whitepaper of bitcoin. You can [find it over here](https://bitcoin.org/bitcoin.pdf). (<https://bitcoin.org/bitcoin.pdf>) Now that completes the first milestone.

**Let’s check out the action steps that you need to take over here:**

- Get acquainted with the various terms and lexicons.
- Read up on the more technical aspects of the blockchain.

- Read the bitcoin whitepaper.

## Milestone #2: Learn How The Process Works

It is pretty surprising to see how many budding “developers” have not had any real world first-hand experience with cryptocurrency at all. How can you possibly innovate and improve upon a platform when you have not used it even once?

It is strongly recommended that you start getting acquainted with the system today.

Go to [Coinbase \(http://bit.ly/2oesV41\)](http://bit.ly/2oesV41) or any other exchange that you are comfortable with or is accessible in your country and buy some coins. You don't need to create an extensive portfolio straightaway, just buy a few coins and see how the whole process works.

It is extremely straightforward. Since you are not going to be buying a lot of coins then simply use a basic online wallet.

These wallets are the easiest to use among all. The creation is super simple because it's basically creating your own account on any of the exchange services. Furthermore, you can access this wallet from any server or any device in the world as long as it is connected to the net. Having said that, there is one big problem when it comes to online wallets. Your private key is going to be saved on another server. This is basically like serving up your key to hackers on a silver platter. Do NOT use online wallets to store huge amounts of your money. Store the bare minimum that you need for exchange purposes.

As you create an extensive portfolio, you must learn how to utilize cold wallets to store your money. [You can learn how to do so here.](#)

[\(https://blockgeeks.com/guides/paper-wallet-guide/\)](https://blockgeeks.com/guides/paper-wallet-guide/) Later on, if you create your ICO then you MUST know how wallets and, in particular, multi-sig wallets work.

We are bringing this section to a close here, the tough part starts from the next milestone.

## Your action steps are here:

- Learn how the exchanges work.
- Get acquainted with wallets.

### Milestone #3: Let's get coding! Blockchain Tutorial

As a blockchain developer, you will face tons of challenges in the back-end. Creating and maintaining a public blockchain is not easy because of a number of reasons.

(Before we continue, a huge shoutout to David Schwartz for his keynote address regarding C++ use in blockchain software development in CPPCON 2016.)

---

#### • Reason #1: Security

Blockchains, as David Schwartz puts it, should be fortresses. Firstly, the code is public and open for all to see. Anyone can look at the code and check for bugs and vulnerabilities. However, unlike other open code resources, the downside of finding vulnerabilities on blockchain code is massive. Any programmer can hack in and get away with potentially millions and millions of dollars. Because of these legitimate security concerns, development on blockchain is usually very slow.

---

#### • Reason #2: Resource Management

It is important to keep pace with the network. You cannot fall too far behind and not keep up with all the network demands. You should be well equipped to handle remote and local queries.

---

#### • Reason #3: Performance

The blockchain must always perform at its highest possible capabilities but

The blockchain must always perform at its highest possible capacities, but for that to happen the language chosen must be extremely versatile. The thing is that there are certain tasks in the blockchain which are parallelizable whilst there are some tasks which can't be done in parallel.

A good example of "parallelizable" task is digital signature verification. All that you need for signature verification is the key, transaction and the signature. With just three data you can conduct verifications in a parallelized manner.

However, not all the functions on a blockchain should be done that way. Think of transaction execution itself. Multiple transactions can't be executed in parallel; it needs to be done one at a time to avoid errors like double spends. Some languages are good at parallel operations while some are good in non-parallel operations.

---

- **Reason #4: Isolation**

What is deterministic behavior?

If  $A + B = C$ , then no matter what the circumstances,  $A+B$  will always be equal to  $C$ . That is called deterministic behavior.

Hash functions are deterministic, meaning  $A$ 's hash will always be  $H(A)$ .

So, in blockchain development, all transaction operations must be deterministic. You cannot have a transaction that behaves one way and then behaves another way the next day. Similarly, you cannot have smart contracts that work in two different ways on two different machines.

The only solution to this is isolation. Basically, you isolate your smart contracts and transactions from non-deterministic elements.

There are some languages which fulfill most of these needs. If you are a blockchain developer, then you definitely need to have [some basic knowledge \(https://blockgeeks.com/guides/blockchain-coding/\)](https://blockgeeks.com/guides/blockchain-coding/) of C++ and JavaScript.

While C++ may seem a little outdated, the truth is that it wonderfully satisfies all the functionalities that we have described above. In fact, Satoshi Nakamoto wrote the Bitcoin source code in C++.

Along with HTML and CSS it is one of the three core technologies in World Wide Web Content Production. Javascript is usually used to create highly interactive web pages.

So, now we will see how to create a very simple blockchain using Javascript.

**Huge shoutout to savjee.be for the content below.**

How do we make a block? What does a simple block consist of? In our simple cryptocurrency that we are going to make (Let's call it "BlockGeeksCoin"), each block will have the following pieces of information:

- **Index:** To know the block number.
- **Timestamp:** To know the time of creation.
- **Data:** The data inside the block.
- **Previous Hash:** The hash of the previous block.
- **Hash:** [The Hash \(https://blockgeeks.com/guides/what-is-hashing/\)](https://blockgeeks.com/guides/what-is-hashing/) of the current block.

Before we continue. You need to understand certain terms that we are going to use in our program:

- **This:** The "this" keyword is invoked inside a function and enables you to access the values inside a specific object that calls that particular function.
- **Constructor:** A constructor is a special function which can help create and initialize an object within a class. Each class is restricted to only one constructor.

**Now that that's done, let's start making our block.**

Creating the Block



```
const SHA256 = require("crypto-js/sha256");

class Block

{
  constructor(index, timestamp, data, previousHash = "")
  {
    this.index = index;
    this.previousHash = previousHash;
    this.timestamp = timestamp;
    this.data = data;
    this.hash = this.calculateHash();
  }
  calculateHash()
  {
    return SHA256(this.index + this.previousHash + this.timestamp
  }
}
```

## Code Analysis

Ok, so this right here is out a block. So, in the first line of the code, we called the crypto-js library because the sha256 hash function is not available in JavaScript.

Next. we invoked a constructor inside the class to call for objects which will

Next, we invoked a constructor inside the class to call for objects which will have certain values. The thing that probably catches your eye is the calculateHash() function. Let's see what exactly is it doing.

In a block, we take all the contents and hash them to get the hash of that particular block. We are using the JSON.stringify function to turn the data of the block into a string to hash it.

Ok, so we have the block ready and good to go. Now let's connect the blocks together into a blockchain.

## Creating the blockchain: Becoming A Blockchain Developer

```
class Blockchain

{

//Section 1 Genesis block creation

  constructor()

  {

this.chain = [this.createGenesisBlock()];

  }

  createGenesisBlock()

  {

    return new Block(0, "01/01/2017", "Genesis block", "0");

  }

//section 2 adding new blocks

  getLatestBlock()
```

```
{  
  
return this.chain[this.chain.length - 1];  
  
}  
  
addBlock(newBlock) {  
  
    newBlock.previousHash = this.getLatestBlock().hash;  
  
    newBlock.hash = newBlock.calculateHash();  
  
    this.chain.push(newBlock);  
  
}  
  
//section 3 validating the chain  
  
isChainValid()  
  
{  
  
for (let i = 1; i < this.chain.length; i++)  
  
{  
  
    const currentBlock = this.chain[i];  
  
    const previousBlock = this.chain[i - 1];  
  
    if (currentBlock.hash !== currentBlock.calculateHash()) {  
  
        return false;  
  
    }  
  
    if (currentBlock.previousHash !== previousBlock.hash)  
  
    {
```

```
return false;

}

}

return true;

}

}
```

## Code Analysis

Ok, so a lot of things are going on in the chain above, let's break it down into sections.

### RELATED GUIDES

---

Intro to ERC-721: The  
CryptoKitty Token

[\(https://blockgeeks.com/guides/erc-721-cryptokitty-token/\)](https://blockgeeks.com/guides/erc-721-cryptokitty-token/)

What are Ethereum  
Nodes And Sharding?

[\(https://blockgeeks.com/guides/what-are-ethereum-nodes-and-sharding/\)](https://blockgeeks.com/guides/what-are-ethereum-nodes-and-sharding/)

What is Monero? Most Comprehensive  
Guide

[\(https://blockgeeks.com/guides/monero/\)](https://blockgeeks.com/guides/monero/)

---

---

- **Section 1: The Genesis Block**

What is the genesis block?

The genesis block is the first block of the blockchain, and the reason why it is special is that while every block points to the block previous to it, the genesis block doesn't point at anything. So, the moment a new chain is created, the genesis block is invoked immediately.

Also, you can see a "createGenesisBlock()" function wherein we have given the data of the block manually:

```
createGenesisBlock()

{
  return new Block(0, "01/01/2017", "Genesis block", "0");
}
```

---

- **Section 2: Adding The Blocks**

Firstly, we will need to know what the last block in the blockchain currently is. For that we use the getLatestBlock() function.

```
getLatestBlock()

{
  return this.chain[this.chain.length - 1];
}
```

Now that we have determined the latest block, let's see how

```
addBlock(newBlock) {
  newBlock.previousHash = this.getLatestBlock().hash;
  newBlock.hash = newBlock.calculateHash();
  this.chain.push(newBlock);
}
```

So, what is happening here? How are we adding the blocks? How are we checking if the given block is valid or not?

Remember the contents of a block? A block has the hash of the previous block right?

So, what we are going to do here is simple. Compare the previous hash value of the new block with the hash value of the latest block.



Image Courtesy: Lauri Hartikka medium article

If these two values match, then this means that the new block is legit and it gets added to the blockchain.

### • Section 3: Validating the Chain

Now, we need to check that nobody has been messing with our blockchain and that everything is stable.

We are using the "for" loop to go from the block 1 to the last block. Genesis block is block 0.

```
for (let i = 1; i < this.chain.length; i++)
```

```
{
```

```
  const currentBlock = this.chain[i];
```

```
  const previousBlock = this.chain[i - 1];
```

In this part of the code we are defining two terms, current

```
if (currentBlock.hash !== currentBlock.calculateHash()) {
```

```
  return false;
```

```
}  
  
if (currentBlock.previousHash !== previousBlock.hash)  
{  
  
    return false;  
  
}  
  
}  
  
return true;  
  
}
```

If the "previousHash" of the current block is not equal to the "Hash" of the previous block, then this function will return False, else it will return True.

Using the blockchain

Now, we are going to finally use the blockchain to create our BlockGeeksCoin.

```
let BlockGeeksCoin = new Blockchain();  
  
BlockGeeksCoin.addBlock(new Block(1, "20/07/2017", { amount:  
BlockGeeksCoin.addBlock(new Block(2, "20/07/2017", { amount:
```

And that's it!

So what happened here?

We created a new cryptocurrency based on the blockchain and named it BlockGeeksCoin. By invoking this new object, I activated the constructor, which in turn created the Genesis block automatically.

We simply added two more blocks to it and gave them some data.

It is that simple.

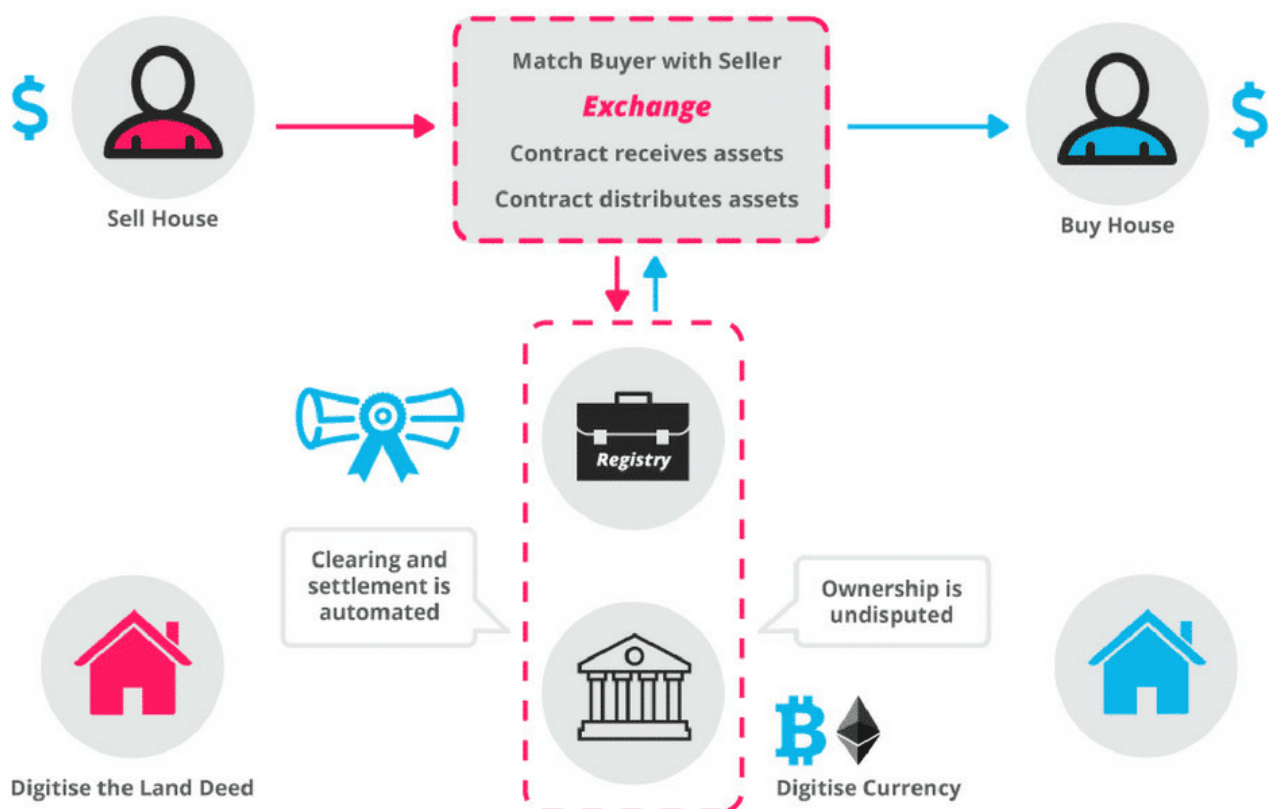
(Thank you savjee.be for the amazing and simple explanation.)

That's it for this milestone. Let's look at the action steps. It is very simple but it definitely isn't easy:

- Get educated in one of the many blockchain friendly languages like C++, Javascript, C#, Go etc.

Milestone #4: Get Educated On Smart Contracts

## How Smart Contracts Works



How do you define a smart contract?

According to Wikipedia ([https://en.wikipedia.org/wiki/Smart\\_contract](https://en.wikipedia.org/wiki/Smart_contract)), a smart contract (<https://blockgeeks.com/guides/smart-contracts/is-a>)



smart contract (<https://blockgeeks.com/guides/smart-contracts/>) is a computer protocol intended to facilitate, verify, or enforce the negotiation or performance of a contract". While it was first proposed by American cryptographer Nick Szabo in 1996, [Ethereum](https://blockgeeks.com/guides/ethereum/) (<https://blockgeeks.com/guides/ethereum/>) is often credited with popularizing the concept and making it mainstream.

You can learn more about smart contracts in [our in-depth guide here](https://blockgeeks.com/guides/smart-contracts/). (<https://blockgeeks.com/guides/smart-contracts/>)

So, what are the desirable properties that we want in our smart contract?

Anything that runs on a blockchain needs to be immutable and must have the ability to run through multiple nodes without compromising on its integrity. As a result of which, smart contract functionality needs to be three things:

- Deterministic.
  - Terminable.
  - Isolated.
- 

## Feature #1: Deterministic

A program is deterministic if it gives the same output to a given input every single time. Eg. If  $3+1 = 4$  then  $3+1$  will ALWAYS be 4 (assuming the same base). So when a program gives the same output to the same set of inputs in different computers, the program is called deterministic.

**There are various moments when a program can act in an un-deterministic manner:**

- **Calling un-deterministic system functions:** When a programmer calls an un-deterministic function in their program.
- **Un-deterministic data resources:** If a program acquires data during runtime and that data source is un-deterministic then the program becomes un-deterministic. Eg. Suppose a program that acquires the top 10 google searches of a particular query. The list may keep changing.
- **Dynamic Calls:** When a program calls the second program it is called dynamic calling. Since the call target is determined only during execution it

dynamic calling. Since the call target is determined only during execution, it is un-deterministic in nature.

---

## Feature #2: Terminable

In mathematical logic, we have an error called “halting problem”. Basically, it states that there is an inability to know whether or not a given program can execute its function in a time limit. In 1936, Alan Turing deduced, using Cantor’s Diagonal Problem, that there is no way to know whether a given program can finish in a time limit or not.

This is obviously a problem with smart contracts because, contracts by definition, must be capable of termination within a given time limit. There are some measures taken to ensure that there is a way to externally “kill” the contract and to not enter into an endless loop which will drain resources:

- **Turing Incompleteness:** A Turing Incomplete blockchain will have limited functionality and not be capable of making jumps and/or loops. Hence they can’t enter an endless loop.
  - **Step and Fee Meter:** A program can simply keep track of the number “steps” it has taken, i.e. the number of instructions it has executed, and then terminate once a particular step count has been executed. Another method is the Fee meter. Here the contracts are executed with a pre-paid fee. Every instruction execution requires a particular amount of fee. If the fee spent exceeds the pre-paid fee then the contract is terminated.
  - **Timer:** Here a pre-determined timer is kept. If the contract execution exceeds the time-limit then it is externally aborted.
- 

## Feature #3: Isolated

In a blockchain, anyone and everyone can upload a smart contract. However

in a blockchain, anyone and everyone can upload a smart contract. However, because of this the contracts may, knowingly and unknowingly contain virus and bugs. If the contract is not isolated, this may hamper the whole system. Hence, it is critical for a contract to be kept isolated in a sandbox to save the entire ecosystem from any negative effects.

Now that we have seen these features, it is important to know how they are executed. Usually the smart contracts are run using one of the two systems:

- Virtual Machines: Ethereum uses this.
- Docker: Fabric uses this.

Let's compare these two and determine which makes for a better ecosystem. For simplicity's sake, we are going to compare Ethereum (Virtual Machine) to Fabric (Docker).

	Virtual Machines	Docker
<b>Deterministic</b>	The contracts have no un-deterministic functions and the data is limited to on-chain information only. However, it executes dynamic calls which can be nondeterministic in nature. Thankfully the data accessible is deterministic	Because of the design of the docker, the system is reliant on users to create contracts which are deterministic. That is not really the best of solutions.
<b>Terminable</b>	Ethereum uses the "Fee-meter" for termination. Every step in the contracts costs "gas" and once the gas cost exceeds the pre-paid fee, the contract is killed.	Fabric uses the timer. However, since the timer can change from node to node because each node has its own computational power there is a risk to the consensus process.
<b>Isolated</b>	Has good isolation properties.	Is namespace-reliant and not capable of proper isolation

If you are interested in Ethereum development specifically then it is important that you learn solidity as well.

For anyone who wants learn how to make [DAPPs](https://blockgeeks.com/guides/dapps/) (<https://blockgeeks.com/guides/dapps/>) (Decentralized Applications) or get into the ICO game, learning Solidity is an absolute must. We already have a detailed guide to it which [you can read here](#).

[\(https://blockgeeks.com/guides/how-to-learn-solidity/\)](https://blockgeeks.com/guides/how-to-learn-solidity/) However, here we are going to give you a basic overview. Solidity was developed by Gavin Wood, Christian Reitwiessner, Alex Beregszaszi, Yoichi Hirai and several former Ethereum core contributors to enable writing smart contracts on blockchain platforms such as Ethereum.

**Solidity** (<https://blockgeeks.com/guides/how-to-learn-solidity/>) is a purposefully slimmed down, loosely-typed language with a syntax very similar to ECMAScript (Javascript). There are some key points to remember from the Ethereum Design Rationale document, namely that we are working on a stack-and-memory model with a 32-byte instruction word size, the EVM (Ethereum Virtual Machine) gives us access to the program "stack" which is like a register space where we can also stick memory addresses to make the Program Counter loop/jump (for sequential program control), an expandable temporary "memory" and a more permanent "storage" which is actually written into the permanent blockchain, and most importantly, the EVM requires total determinism within the smart contracts.

If you are interested in learning solidity then you can check our in-depth [class here. \(https://courses.blockgeeks.com/\)](https://courses.blockgeeks.com/)

**So, let's see the action steps now:**

- Understand how smart contracts work.
- (Optional for Ethereum developers) [Learn Solidity. \(https://courses.blockgeeks.com/\)](https://courses.blockgeeks.com/)

**Milestone #5: Be In The Mix**

One of the most important things that you can do as a budding developer is to constantly stay in the mix.

Go and join the Reddit forums, Github pages, and StackExchange and connect with other developers and always be on the lookout for any news regarding the technology.

Along with that, it will be helpful for you to know what people look for in blockchain developer. What qualities are companies looking for when they are looking to hire? [You can find that information here.](#)

[\(https://blockgeeks.com/how-to-hire-a-good-blockchain-developer/\)](https://blockgeeks.com/how-to-hire-a-good-blockchain-developer/)

This information can be very useful in you fine-tuning your skills enough to appeal to the companies.

## Blockchain Developer: Conclusion

So, this is a rough roadmap for you and your journey to becoming a blockchain developer. This alone won't be enough, of course, you will need to show your own initiative and always be in the mix.

[If you are looking for a resource of information on blockchain development then click here. \(https://blockgeeks.com/\)](#)

We wish you all the best on your journey!

---

Like what you read? Give us one like or share it to your friends

96

9

Discussion

Comments

---

You must be logged in ([https://blockgeeks.com/wp-login.php?redirect\\_to=https%3A%2F%2Fblockgeeks.com%2Fguides%2Fblockchain-developer%2F](https://blockgeeks.com/wp-login.php?redirect_to=https%3A%2F%2Fblockgeeks.com%2Fguides%2Fblockchain-developer%2F)) to post a comment.

---

Courses


(<https://courses.blockgeeks.com>)


Blockchain Jobs  
([/talent/](https://courses.blockgeeks.com/talent/))

Affiliates

(<https://courses.blockgeeks.com/affiliates/>)

	Support	
	(https://blockgeeks.fr	Hackathons
	eshdesk.com/support	(https://bountyone.io
Terms (/terms)	/home)	/hackathons)
Partnership		
(https://blockgeeks.c	Privacy Policy	Faq
om/blockchain-	(https://blockgeeks.c	(https://courses.block
partnership/)	om/policy/)	geeks.com/faq/)
	Guest Posts	
	(https://blockgeeks.c	Scholarships
Free Webinars	om/guest-post-	(https://blockgeeks.c
(/webinars)	submissions/)	om/scholarships/)
Events		
(https://blockgeeks.c		
om/events/)		





<https://blockgeeks.com>

© 2019 Blockgeeks

/B  
YA  
kf  
Q)