A Workshop on

Blockchain Technology

PDSC

Lecture 1

EVM, Smart Contracts and Basics of Solidity

Yesterday, we learnt

- Centralized vs Decentralized vs Distributed
- P2P Systems
- Why did Blockchain come into existence?
- How I let everyone copy my assignments? SMH...
- How transactions happen?
- Public and Private Key
- Building your own blockchain
- Cryptographic Hashes
- Merkel Tree

Today, we will learn

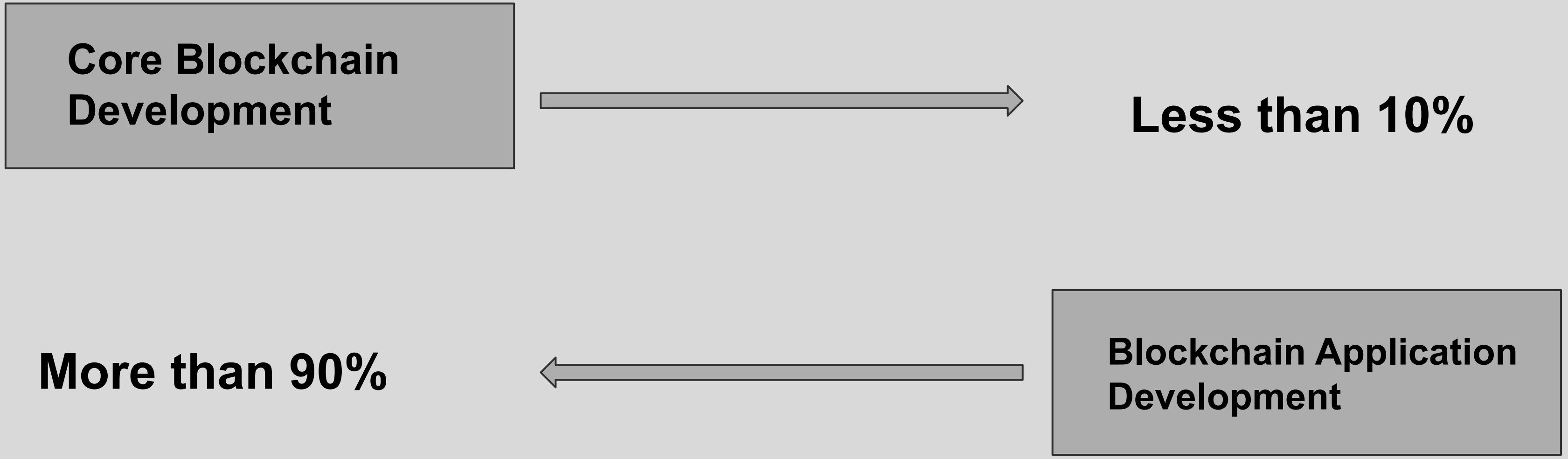
- The dApp Universe
- Blockchain Application Development
- Ethereum Virtual Machine
- Smart Contracts
- Basics of Solidity

The dApp Universe

Lecture 1

EVM, Smart Contracts and Basics of Solidity

Blockchain Development


Core Blockchain Development
Less than 10%
More than 90%
Blockchain Application Development

Decentralized Applications (dApps)

- Applications that run on peer-to-peer network or blockchain network
- Runs on a network rather than a single computer
- P2P network Apps - BitTorrent, Tor
- Blockchain Apps - ChainLink, Brave, Rahat

Blockchain Application Development

- Built on top of an existing blockchain
- The most popular blockchain for application development - Ethereum
- Others - Solana, Cardano, Tezos, Polkadot

Q: Can you build applications on the Bitcoin Blockchain?

Lecture 1

EVM, Smart Contracts and Basics of Solidity

Ethereum Blockchain Development

Lecture 1

EVM, Smart Contracts and Basics of Solidity

Why is Ethereum Blockchain so popular?

- Pre-existing development environment
- Our lord and savior: Ethereum Virtual Machine (EVM)
- Built as a Turing Complete Blockchain
- Smart Contracts written in Solidity
- Possibility of adequate monetization

- Also, high gas fees and slower transactions, but let's ignore that for now. :)

Let’s Talk about EVM

Lecture 1

EVM, Smart Contracts and Basics of Solidity

Ethereum Virtual Machine (EVM)

Creates a level of abstraction

Also, improves portability of software and separates applications and hosts

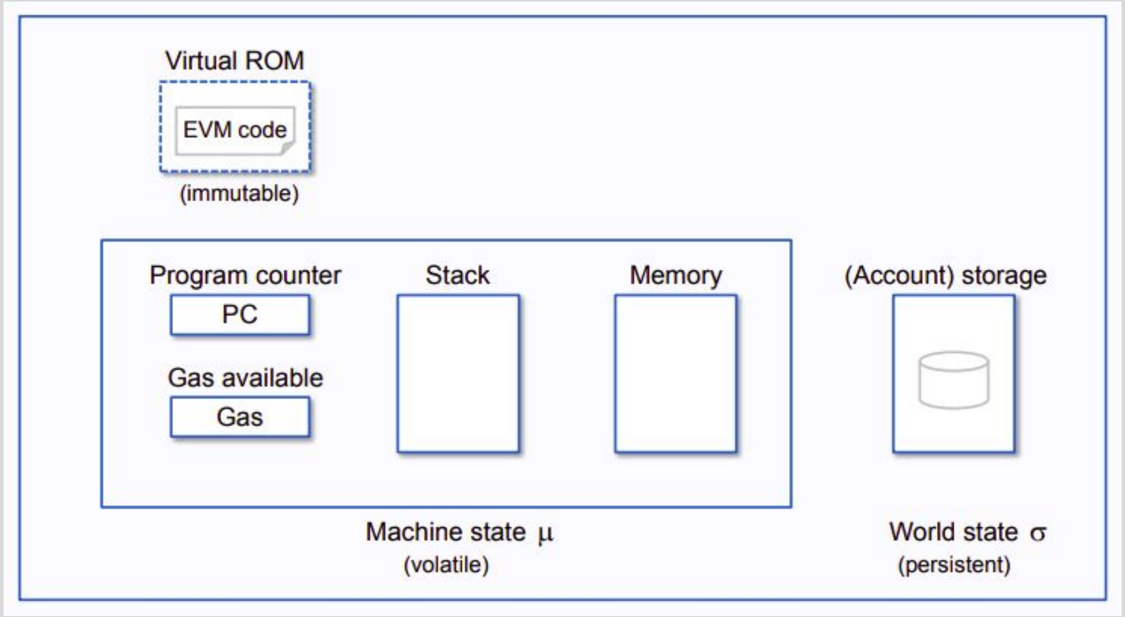
Why learn about EVM?

To understand how smart contracts actually work

Make your own toolkit

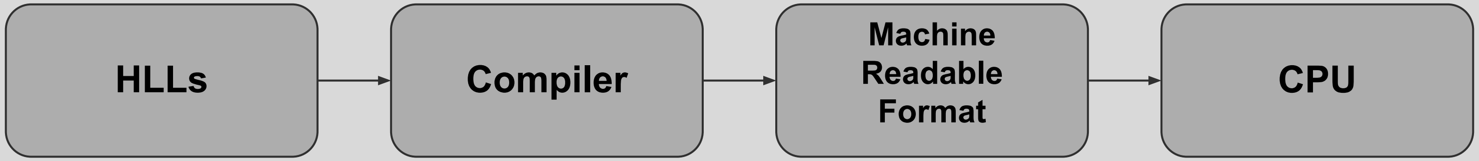
Intellectual Challenge

Ethereum Virtual Machine(EVM)



So, how does EVM work?

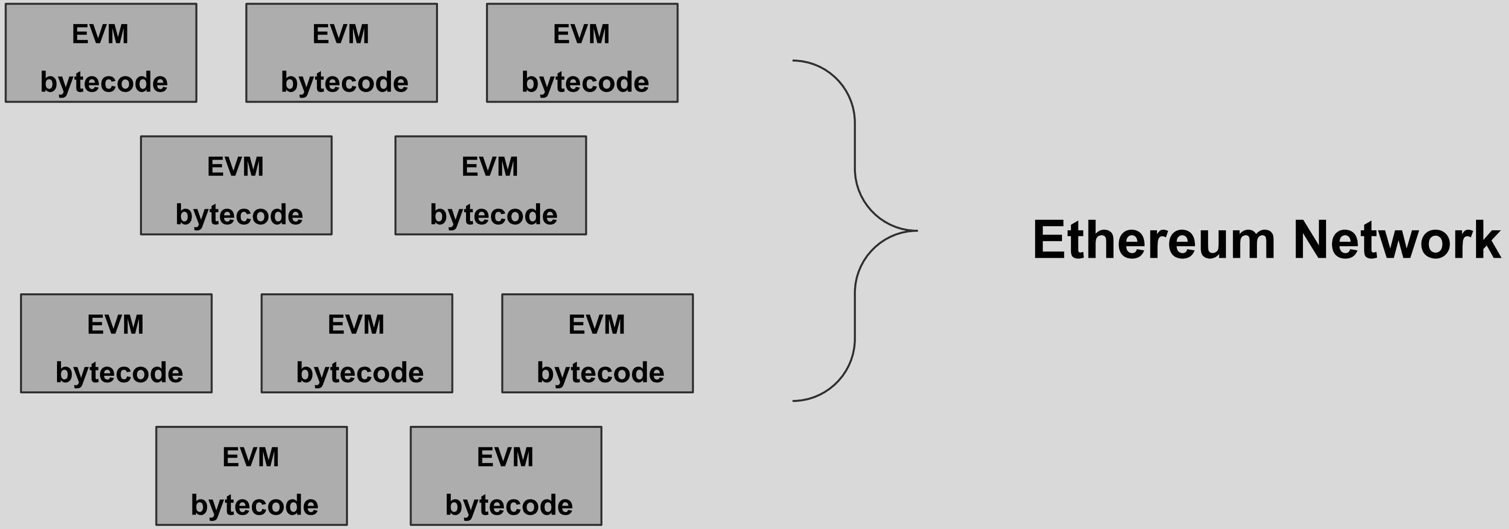
- We know how a computer understands high level languages



Ethereum Virtual Machine (EVM)

- EVM is analogous to CPU
- Not physical, ofc
- It runs with Ethereum Protocol implementations (like Geth)
- Executes the bytecode obtained from solidity smart contracts

Ethereum Virtual Machine (EVM)



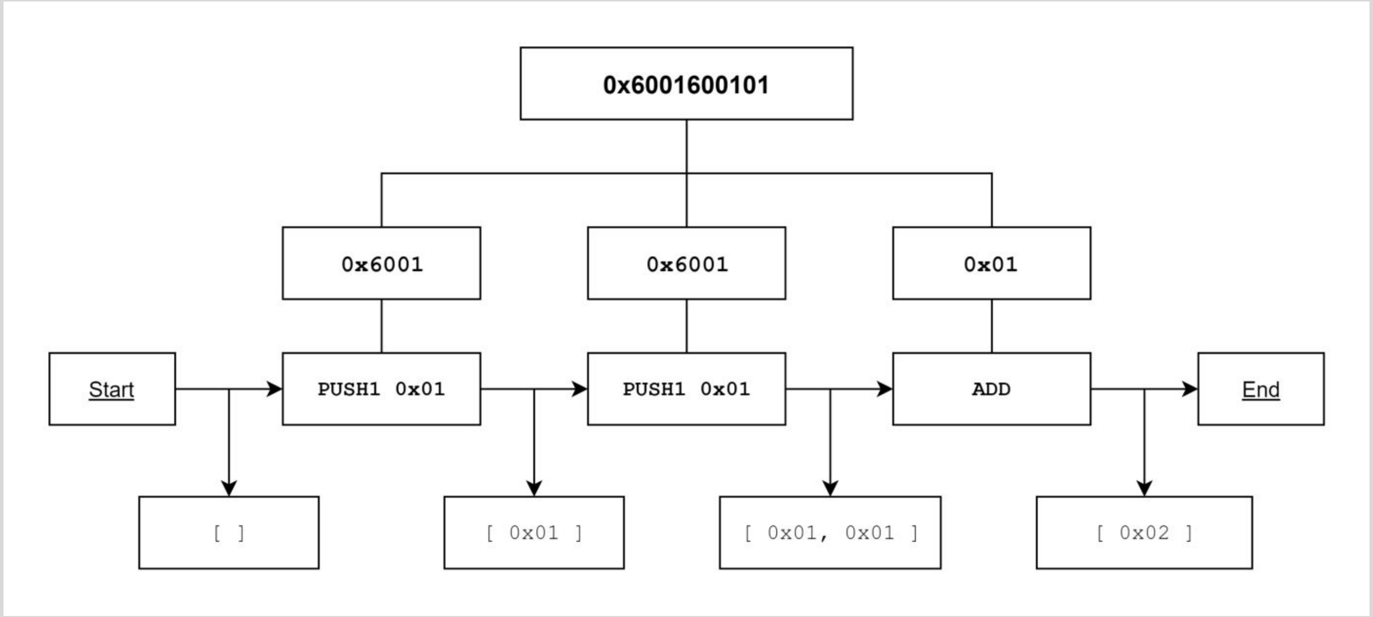
So, how does EVM work?

- Solidity is compiled to bytecode
- Bytecode is distributed to different machines
- Hence, all of them have a copy after deployment
- When smart contract is called, all machines pull up the copy and execute the bytecode

Opcode and ByteCode

- Solidity Compiler compiles solidity code to machine level instructions
- Opcodes - for executing specific tasks
- Stack Manipulation, Arithmetic/Logic/Bitwise Operation, Halting Opcodes
- Bytecodes - for efficiently storing opcodes

Opcode and ByteCode



Memory and Storage

- EVM uses a 256-bit register stack
- Only 16 items can be accessed or manipulated at once
- Because of these constraints, complicated opcodes use contract memory
- Memory is not persistent, hence will not be saved

- Using storage for data persistency. Also no transaction fee!!
- So, why not always use storage?
- Writing to storage is very expensive!

So, how does EVM work?

- When smart contract is called, all machines pull up the copy and execute the bytecode
- This results in state change!
- What is a state?

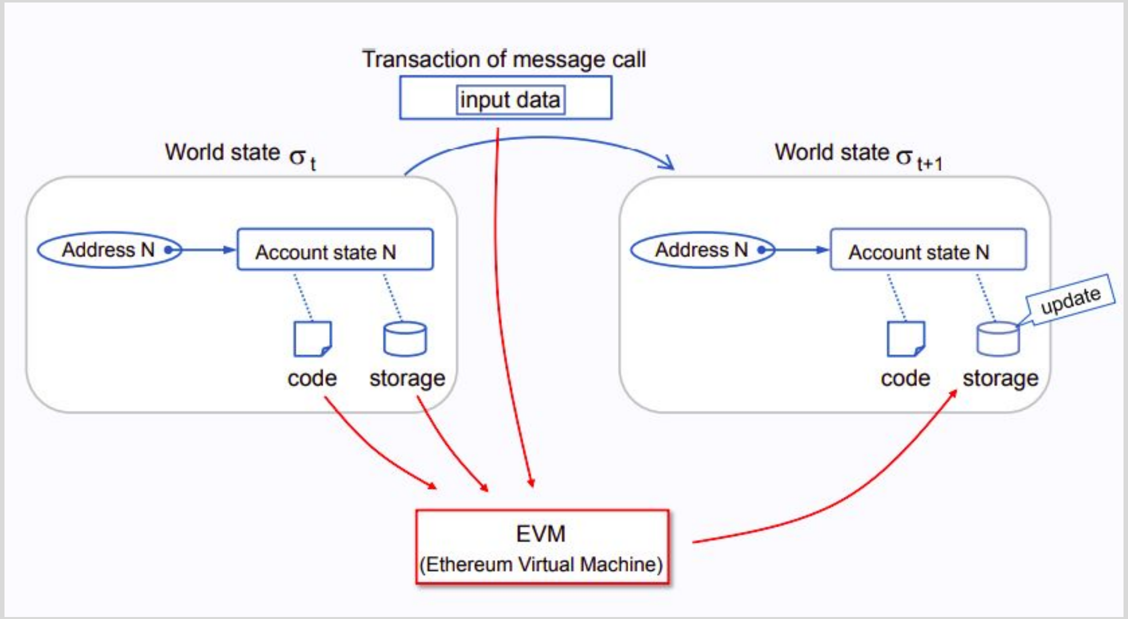
State in Blockchain

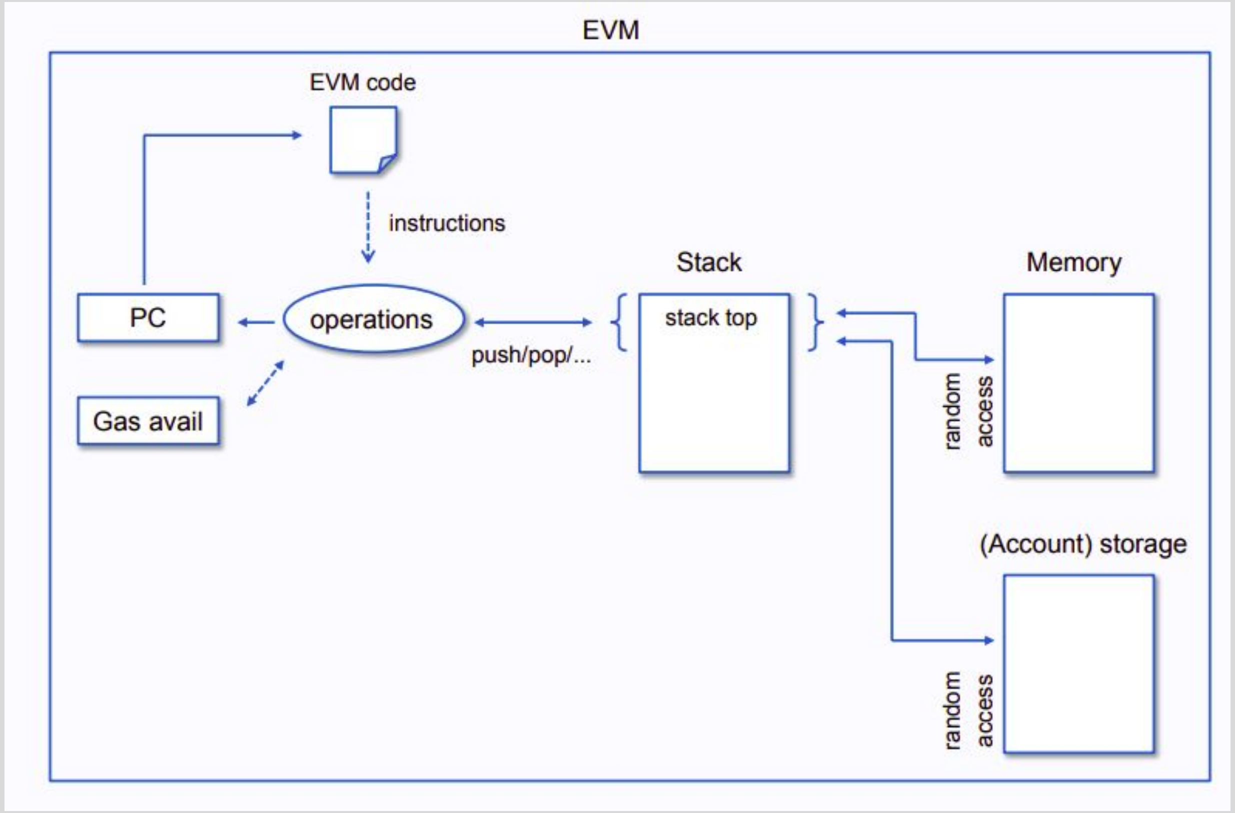
- The documentation says

  "In the context of Ethereum, the state is an enormous data structure called a modified Merkle Patricia Trie, which keeps all accounts linked by hashes and reducible to a single root hash stored on the blockchain."

- **For our convenience, we say, "State is how our blockchain looks at present"**
- Execution of smart contracts brings about transactions
- Transactions bring state change

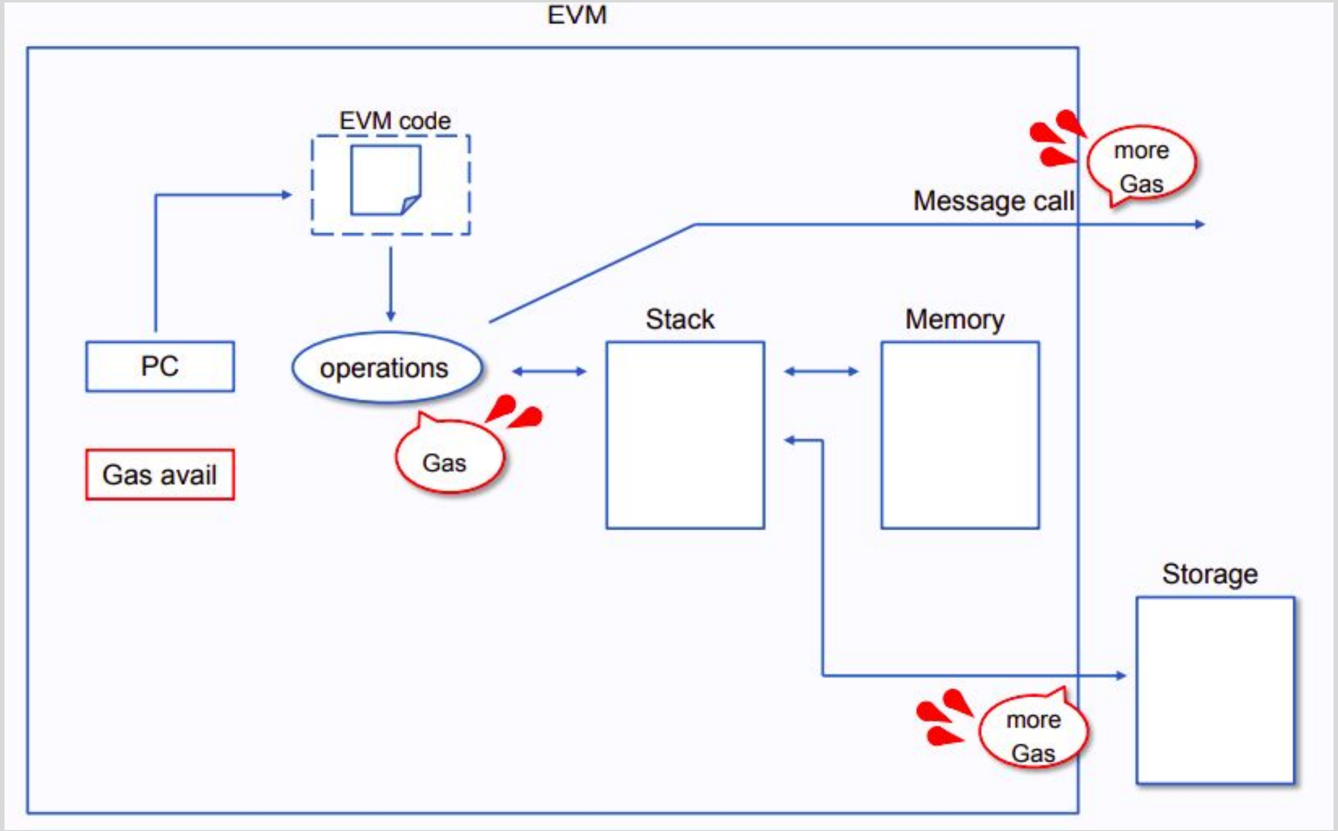
State Change




EVM
EVM code
instructions
PC
operations
push/pop/...
Gas avail
Stack
stack top
Memory
random access
(Account) storage
random access

Gas Cost

- All programmable computation in Ethereum is subject to fees (denominated in gas).
- Computationally expensive instructions charge a higher gas fee than simple, straightforward instructions.
- Two main factors for determining the price of a transaction: gas limit and gas price.
- Transaction fee = gas limit x gas price
- Gas price measured in Gwei, 1 Gwei = 0.000000001 ETH


EVM
EVM code
more Gas
Message call
PC
operations
Stack
Memory
Gas
Gas avail
Storage
more Gas

Smart Contracts

Lecture 1

EVM, Smart Contracts and Basics of Solidity

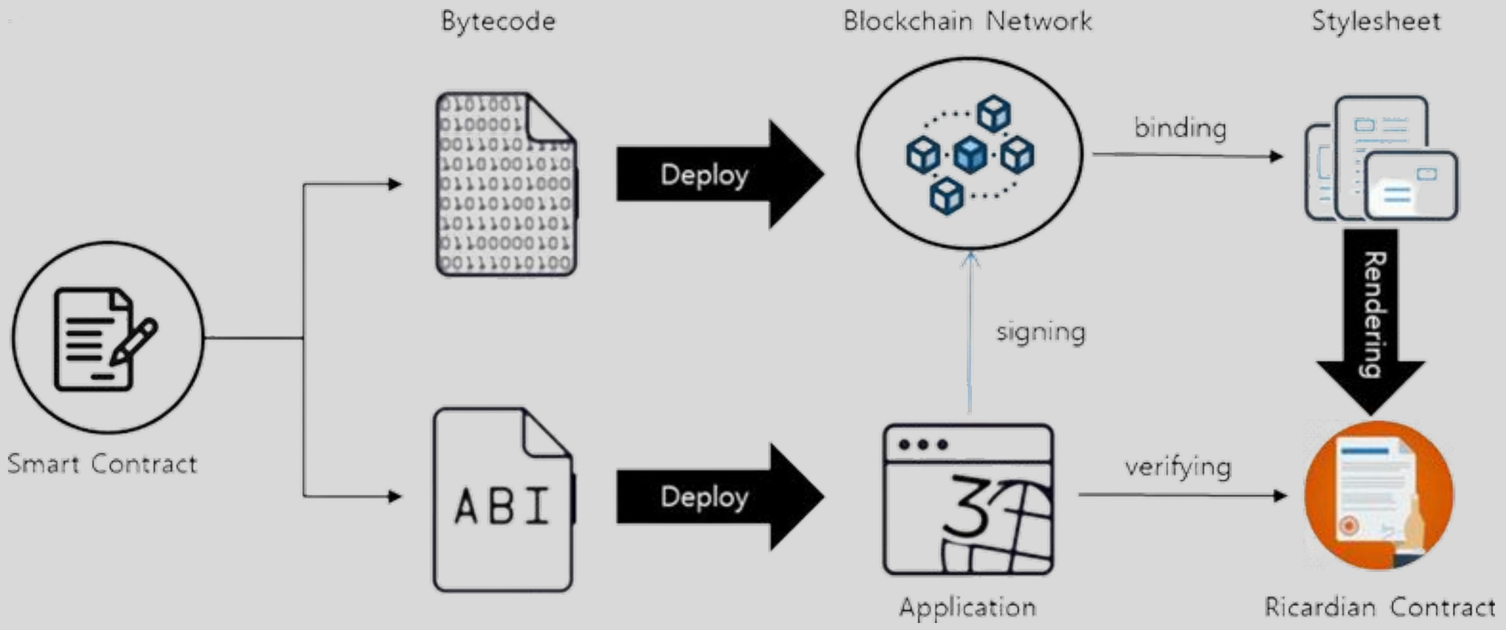
Smart Contracts

- **IBM says**

  Smart Contracts are simply programs stored on a blockchain that run when predetermined conditions are met.

  They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss.

Smart Contracts



Smart Contract Development

- Solidity
- Vyper
- Rust
- Javascript
- Yul

Smart Contract Example 1

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
  }
```

Smart Contract Example 2

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.7;

contract MyContract {

    constructor() public{
        value = "My value";
    }

    string public value;

    function get() public view returns (string memory){
        return value;
    }

    function set(string memory _value) public{
       value = _value;
   }
}
```

Basics of Solidity

Lecture 1

EVM, Smart Contracts and Basics of Solidity

Solidity Programming Language

- Object-oriented, high-level language for implementing smart contracts.
- Curly-bracket language that has been most profoundly influenced by C++.
- Statically typed (the type of a variable is known at compile time).
- Features
  - Inheritance
  - Libraries
  - Complex user-defined types

Layout of a Solidity Source File

- SPDX License Identifier

```
// SPDX-License-Identifier: MIT
```

- Pragmas
  - Version Pragmas

    ```
    pragma solidity ^0.5.2;
    ```

  - ABI Coder Pragmas

    ```
    pragma abicoder v1;
    pragma abicoder v2;
    ```

Layout of a Solidity Source File

- Importing other source files and paths

```
import "filename";
```

- Comments

```
    // This is a single-line comment.

    /*
    This is a
    multi-line comment.
    */
```

Structure of a Contract

- State Variables
- Functions
- Function Modifiers
- Events
- Errors
- Structs
- Enums

Solidity Basics

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.16 <0.9.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
  }
```

License

Solidity Version

Value Types and Everything after that...

Our Lord and Savior:
The Documentation

Thank You for Listening!

Any Questions?

Lecture 1

EVM, Smart Contracts and Basics of Solidity