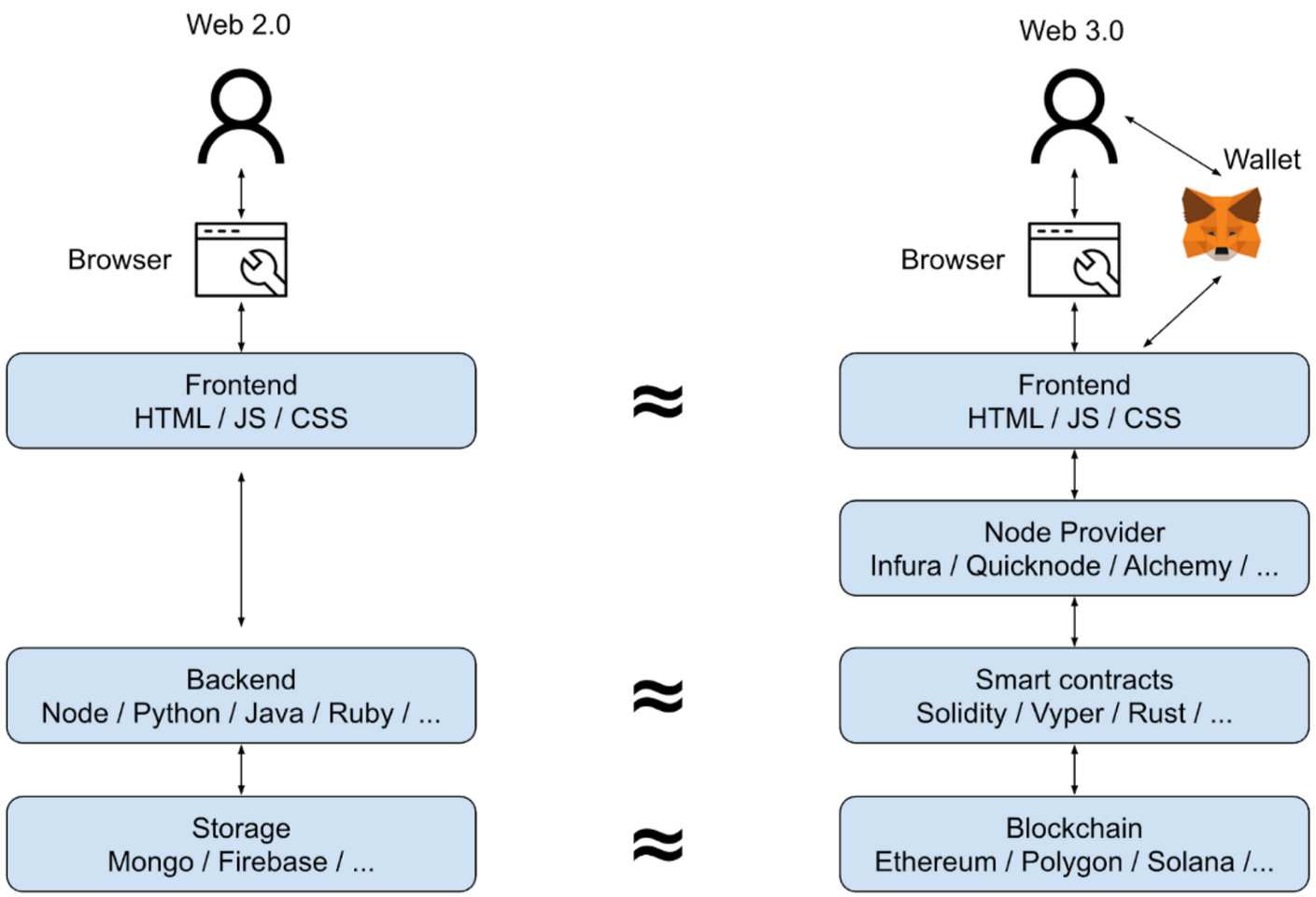
Coding on the World Computer

Hi, I'm Kaushal

- SDE2 at Pedals Up
- CTO and Cofounder at getriff.xyz
- Builder

Twitter, Linkedin, Github, Instagram



What is a dApp?

The Proposed Solution

Decentralized applications (dApps) are digital applications or programs that exist and run on a blockchain or peer-to-peer (P2P) network of computers instead of a single computer. DApps (also called "dapps") are thus outside the purview and control of a single authority.

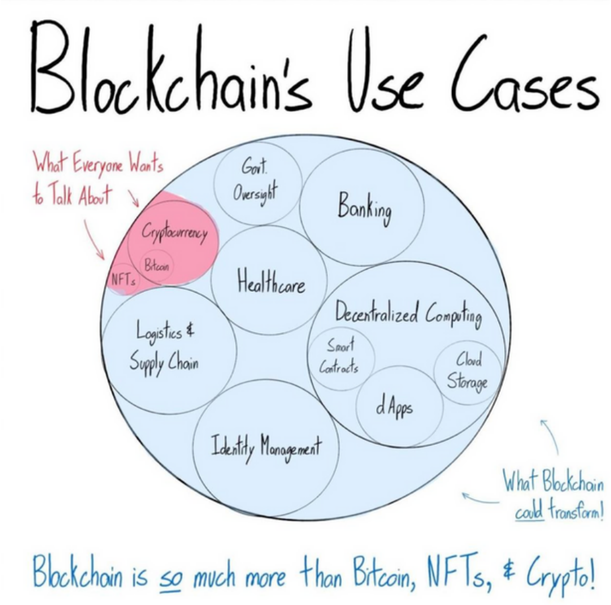
“Decentralised APIs? APIs on the blockchain? Eh.”

-: Kaushal Patil

A smart contract is a self-executing contract with the terms of the agreement between buyer and seller being directly written into lines of code - Szabo

https://en.wikipedia.org/wiki/Smart_contract

Smart Contract

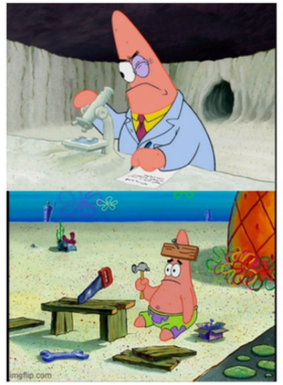


A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain.

Pick Your Poi.. <u>Language</u>

1. Solidity
2. Vyper
3. Yul and Yul+
4. FE

Solidity



- 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).
- Supports:
  - Inheritance (you can extend other contracts).
  - Libraries (you can create reusable code that you can call from different contracts – like static functions in a static class in other object oriented programming languages).
  - Complex user-defined types

Solidity Docs

For Self Study :P

Let's build a pokedex on ethereum/polygon!
https://remix.ethereum.org/

https://github.com/Kaushal1011/AI-Web3-Talk/blob/main/Pokedex.sol

Some Concepts

1. Functions
2. Data Locations
3. Modifiers
4. Inheritance
5. Visibility and Interface

Anatomy of a Contract

Application: <u>Aave</u>

Aave is an Open Source and Non-Custodial protocol to earn interest on deposits and borrow assets with a variable or stable interest rate. The protocol is designed for easy integration into your products and services.

Libraries and Extending Contracts

We can extend contract functionality by using inheritance and overloading

Contract functionality and also be extended by importing libraries from other contracts or inheriting contracts from packages.

Concepts

- Overloading
- Library Contracts
- Imports

Transactions and Calls Among Contracts

You can send ether to a contract, send ether from a contract to a contract, from a contract to a user

Sending Ether, Receive Ether

Lets Dive Deep!

Disect the above contract, Find bugs and Fix them!

Contract Interaction

1. Via Contract/Interface (Passing inside)
2. Via Contract/Interface type
3. Using Call

Contract Interaction Examples

1. The below-mentioned contracts contain bugs, that should be solved by you.
2. They are in no way the "correct way" of doing things but just serve as an example of contract interaction.
3. They are not safe for production.

- ScamToken
- ScamTokenInterface
- ContractInteractionExample

Challenge: Create a contract in which the user can buy/sell a token, where the price is proportional to the amount of time passed from the start of a year.

Bigger Challenge: Make the price grow on a quadratic curve

i.e. : Token is cheapest in Jan and Costliest in Dec

ABI Encoding and Decoding

- Solidity Docs
- More Information

Hash In Solidity

- Hash
- Medium Blog on Keccak256
- Applied

Keccak256 Hash Functions

```
function collisionExample(stri
public pure returns (bytes32)
    return keccak256(abi.en
}
```

(AAA,
(AA,

Function Selectors

When a function is called, the first 4 bytes of calldata specifies which function to call. This 4 bytes is called a function selector.

```
bytes4(keccak256("foo(uint256,address,string,uint256[2])"))
```

- Solidity By Example
- Information

Delegate Calls

There exists a special variant of a message call, named delegatecall which is identical to a message call apart from the fact that the code at the target address is executed in the context of the calling contract and msg.sender and msg.value do not change their values.

What can be a possible use of delegate calls?

Some EVM OP Codes

EVM- OPCodes

- The EVM uses a set of instructions (called *opcodes*) to execute specific tasks.
- At the time of writing, there are **140 unique opcodes**.
- For simplicity's sake, we can split all opcodes into the different categories.

Categories of opcodes

- **Stack-manipulating opcodes** (*POP, PUSH, DUP, SWAP*)
- **Arithmetic/comparison/bitwise opcodes** (*ADD, SUB, GT, LT, AND, OR*)
- **Environmental opcodes** (*CALLER, CALLVALUE, NUMBER*)
- **Memory-manipulating opcodes** (*MLOAD, MSTORE, MSTORE8, MSIZE*)
- **Storage-manipulating opcodes** (*SLOAD, SSTORE*)
- **Program counter related opcodes** (*JUMP, JUMPI, PC, JUMPDEST*)
- **Halting opcodes** (*STOP, RETURN, REVERT, INVALID, SELFDESTRUCT*)

Body