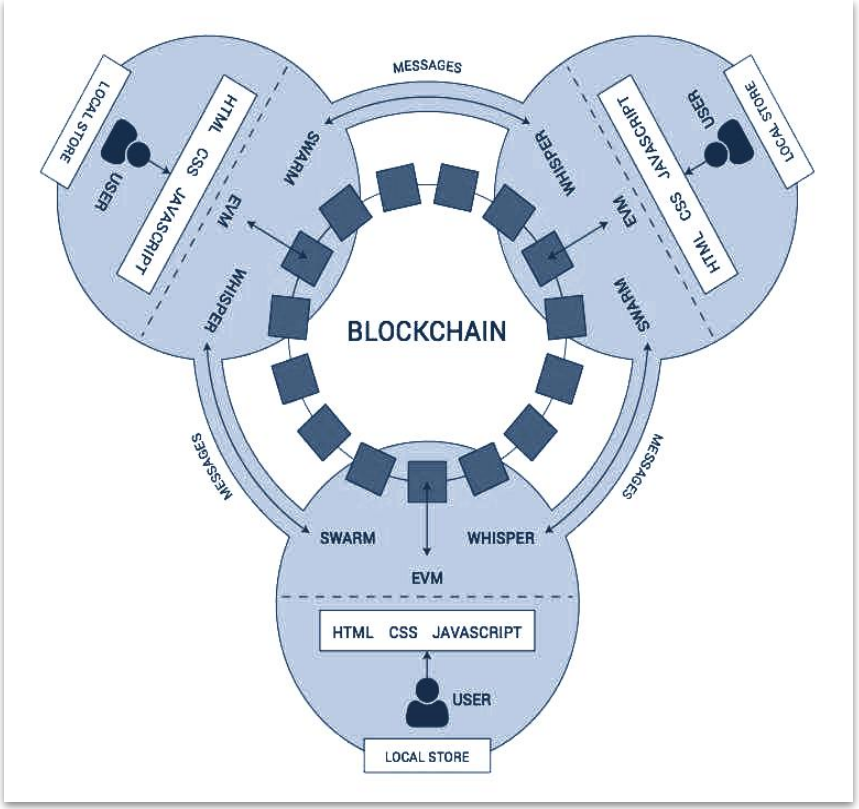
Ethereum DApp Architecture

DApp - Definition & Features

- DApp is an abbreviated form for decentralized application.
- A DApp has its backend code running on a decentralized peer-to-peer network
- Contrast this with an app where the backend code is running on centralized servers.
- DApps have been mostly popularized by distributed ledger technologies, namely the Ethereum Blockchain
- **App = frontend + server**
- **DApp = frontend + contracts**



DApp - Definition & Features

- dApps connect users and developers directly without the need for a middleman to host and manage the code and user data

- Permission is not needed to build a dApp and there is no company or centralized group of people that can change the rules of the platform

- Today there are over 1000 dApps built on Ethereum, the leading dApp platform.

  http://www.stateofthedapps.com

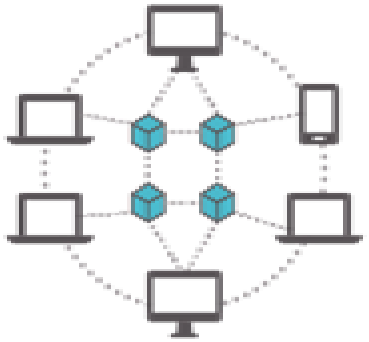
- **91 of the top 100 dApps are built on Ethereum, has 30 times more developers than the next blockchain community.**

- A dApp can have its frontend code written in any programming language that makes API calls to its backend

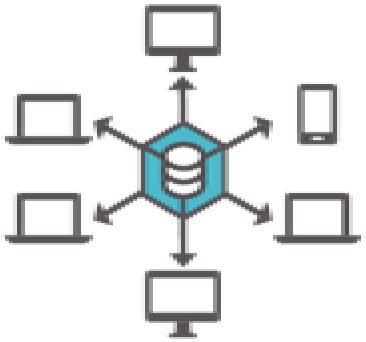
DApp - Definition & Features

- The biggest difference between dApp development and traditional app development: **Level of scrutiny required before taking app to production..!**
- For example: Hardware Vs Software Development
- In dApp development, a smart contract can't be changed once it's launched on the mainnet
- A bug in the smart contract loses users' funds and tarnish the reputation of the dApp developers.
- App development, in contrast, tends to emphasize fast iteration cycles as best practice - you want to build a minimum viable product, get people testing the product, and release updated versions as quickly as possible.
- The motto **"move fast and break things"** will not work in the case of dApps !!!

Difference between App and dApp



DECENTRALIZED APP



TRADITIONAL APP

- No notion of centrality
- Runs on a peer to peer network
- Source code of app is available to all.
- App has crypto-tokens/digital assets for fueling itself.
- Generates tokens and has an inbuilt consensus mechanism.

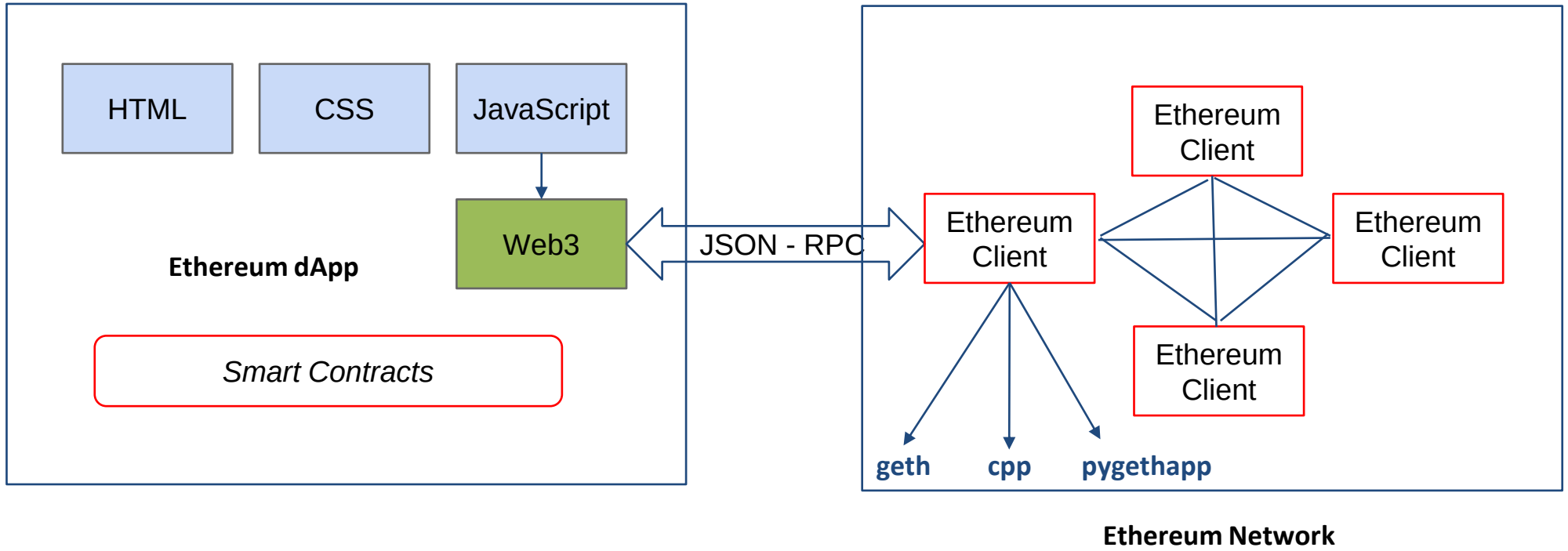
- Notion of centrality
- Runs on a centralized server
- Source code of app is not available to all.
- Chances of failure is more
- Prone to attacks
- Less reliable

DApp - Classifications

- Based on the Blockchain used by the dApps, we can classify them into 3:

Type I	Type II	Type III
These types of DApps have their own blockchain Example: Bitcoin	These types of DApps use the blockchain of Type I DApps Type II decentralized applications are protocols and have tokens that are necessary for their function. The Omni Protocol is an example of Type II decentralized application.	These types of DApps use the protocol of a Type II DApp. For example, the SAFE network uses the Omni Protocol for issuing SafeCoins that are then used to build distributed file storage.

Ethereum dApp Architecture



Ethereum dApp Architecture

- **Frontend**: HTML + CSS + JavaScript

- **web3.js: Ethereum JavaScript Library**

  - Is a collection of libraries
  - allow you to interact with a local or remote ethereum node
  - HTTP or IPC connection.
  - reads and writes data from the blockchain with smart contracts.
  - Web3.js talks to The Ethereum Blockchain with JSON RPC

Ethereum dApp Architecture

- **JSON-RPC:**

  - Is a remote procedure call protocol encoded in JSON
  - It is a simple protocol, only a few data types and commands
  - Is a stateless, lightweight remote procedure call (RPC) protocol.

- **Ethereum Client:**
  - It refers to any node able to parse and verify the blockchain
  - It also provides interfaces to create transactions
  - Mine blocks which is the key for any blockchain interaction
  - **Example: geth (go-ethereum), pyethapp (Python), cpp-ethereum**

Ethereum Tech Stack

Front-End / UI / DApp

HTML | LAMP | iOS | Android | IoT | API

Abstracted Out

Truffle

Web3

Ethereum Core

Smart Contracts

C++ ETH | Go Geth | Java | Haskell

Ethereum Client

Networking Layer

Windows Server | **Linux**

Dev VM

Web2 Vs Web3

	WEB2 APP	WEB3 DAPP
COMPUTATION	SERVER	PEER-TO-PEER NETWORK
HOSTING	WEB SERVER	DISTRIBUTED CDNS (SWARM/IPFS)
SERVICES LAYER	HTTP API	SMART CONTRACTS
STORAGE	DATABASE	DISTRIBUTED STORAGE, BLOCKCHAIN

- Web 1.0 represented the 'readable' internet, presenting webpages and other forms of information
- Web 2.0 represents the 'writeable' internet, allowing a greater level of interactivity between users and websites
- Web 3.0 represents the 'executable' phase of the internet, computers can interpret information like humans and intelligently generate and distribute useful content tailored to the needs of users.
- Web 3.0 is the 3rd generation of the Internet where the devices are connected in a decentralized network rather than depending on server-based databases
- The new internet is user-centric, more secured, private and better connected.

The History of the Web and benefits of Web 3.0

- **Web 1.0** – Basic HTML and e-mail (1990s)
- **Web 2.0** – Informative and Interactive (2000) – Information Centric
- **Web 3.0** – Decentralized, Private and Secure (2020) – User Centric

BENEFITS

- Anti-monopoly and Pro-privacy
- Secure Network
- Data Ownership
- Interoperability
- No Interruption in Service
- Permission-less Blockchains
- Semantic Web

Web 3.0 Stack

dApp Browsers

Decentralized Applications

Storage | Messaging | EVM | Consensus |
Data Feed | Off-chain Computing | Internet of Things

Hardware Clients

Internet Protocol Networks

THANK YOU