



Blockchain

web3.js, dApp

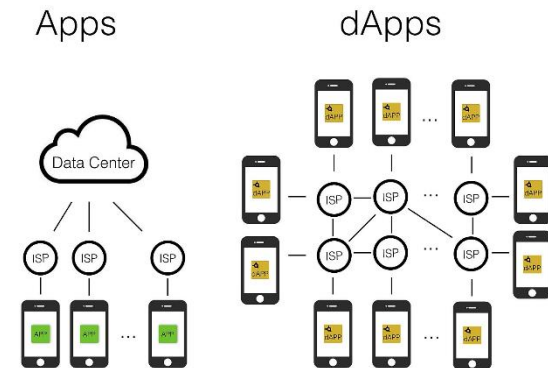
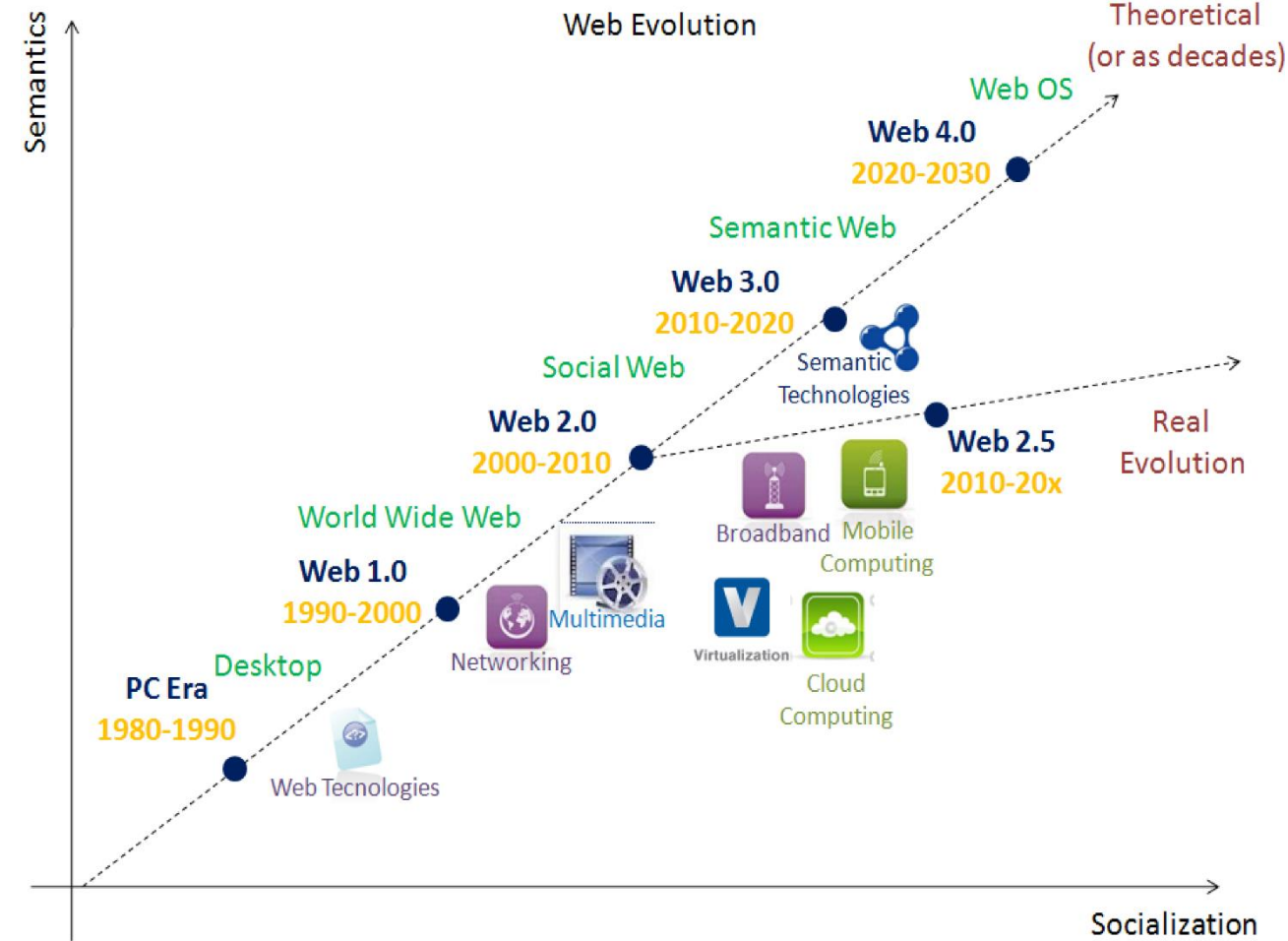


Dr. Sarwan Singh
NIELIT Chandigarh



agenda

- Blockchain and dApp
- Third Age of the Internet
- App vs dApp
- Web application vs Distributed Application(Dapp)
- Building dApp
- dApp workflow
- dApp Architecture



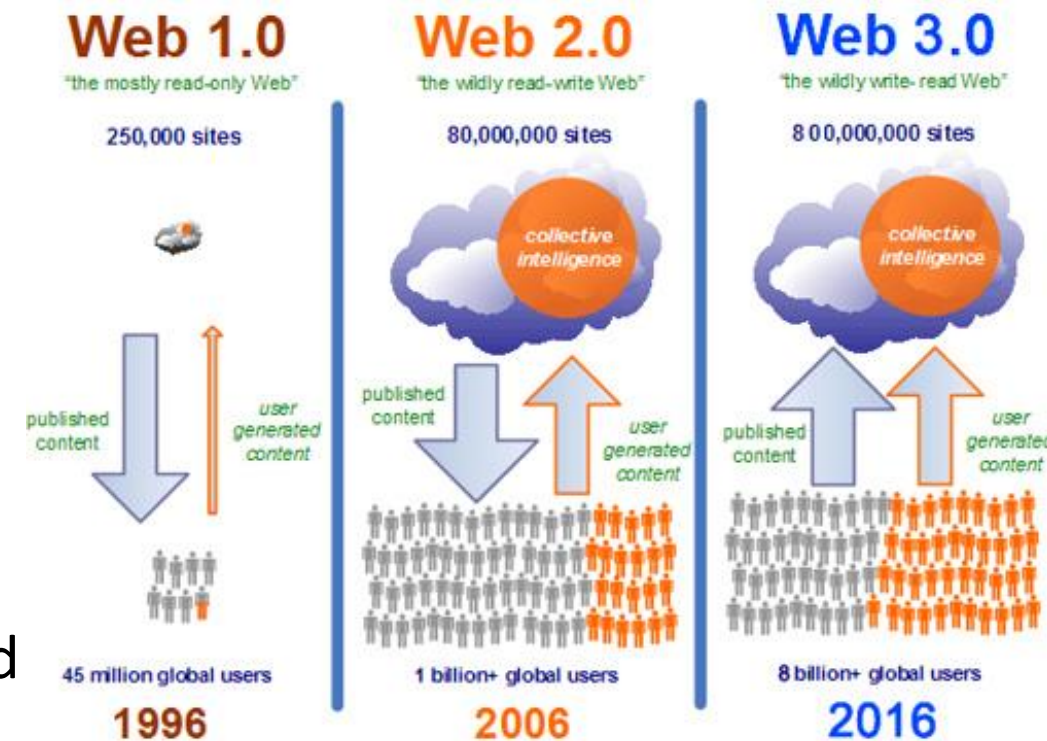


References

- Medium.com - Blockchain
- “*Blockchain Applications: A Hands-On Approach*”, Arshdeep Bahga, Vijay Madisetti
- eattheblocks.com
- dappuniversity.com
- towardsdatascience.com

From General-Purpose Blockchains to Decentralized Applications (DApps)

- DApps represent a broader perspective than smart contracts.
- A DApp is composed of at least:
 - Smart contracts on a blockchain
 - A web frontend user interface
- In addition, many DApps include other decentralized components, such as:
 - A decentralized (P2P) storage protocol and platform
 - A decentralized (P2P) messaging protocol and platform

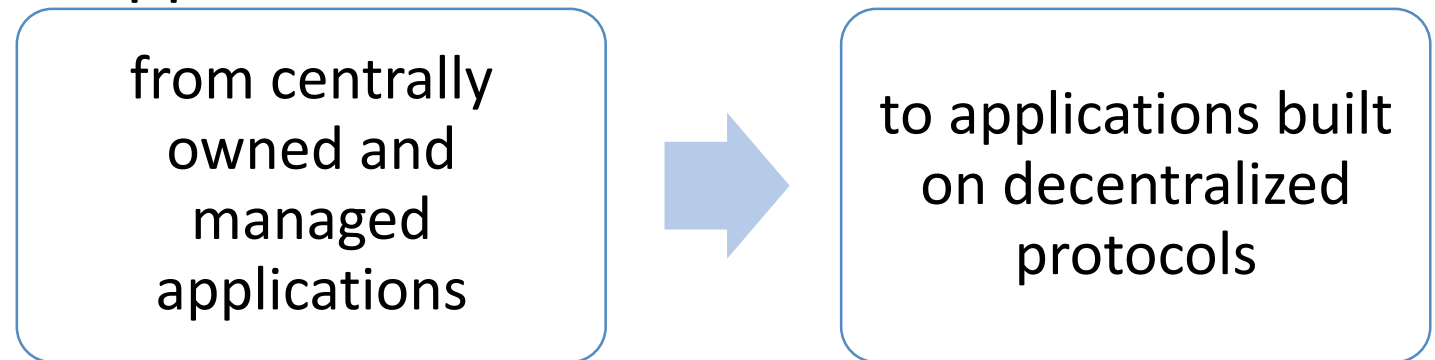






The Third Age of the Internet- *evolving WWW*

- In 2004, the term "Web 2.0" came to prominence, describing an evolution of the web towards user generated content, responsive interfaces and interactivity.
- **evolution is Web3** - first proposed by Gavin Wood, web3 represents a new vision and focus for web applications:



G. Wood is CTO and co-founder of Ethereum



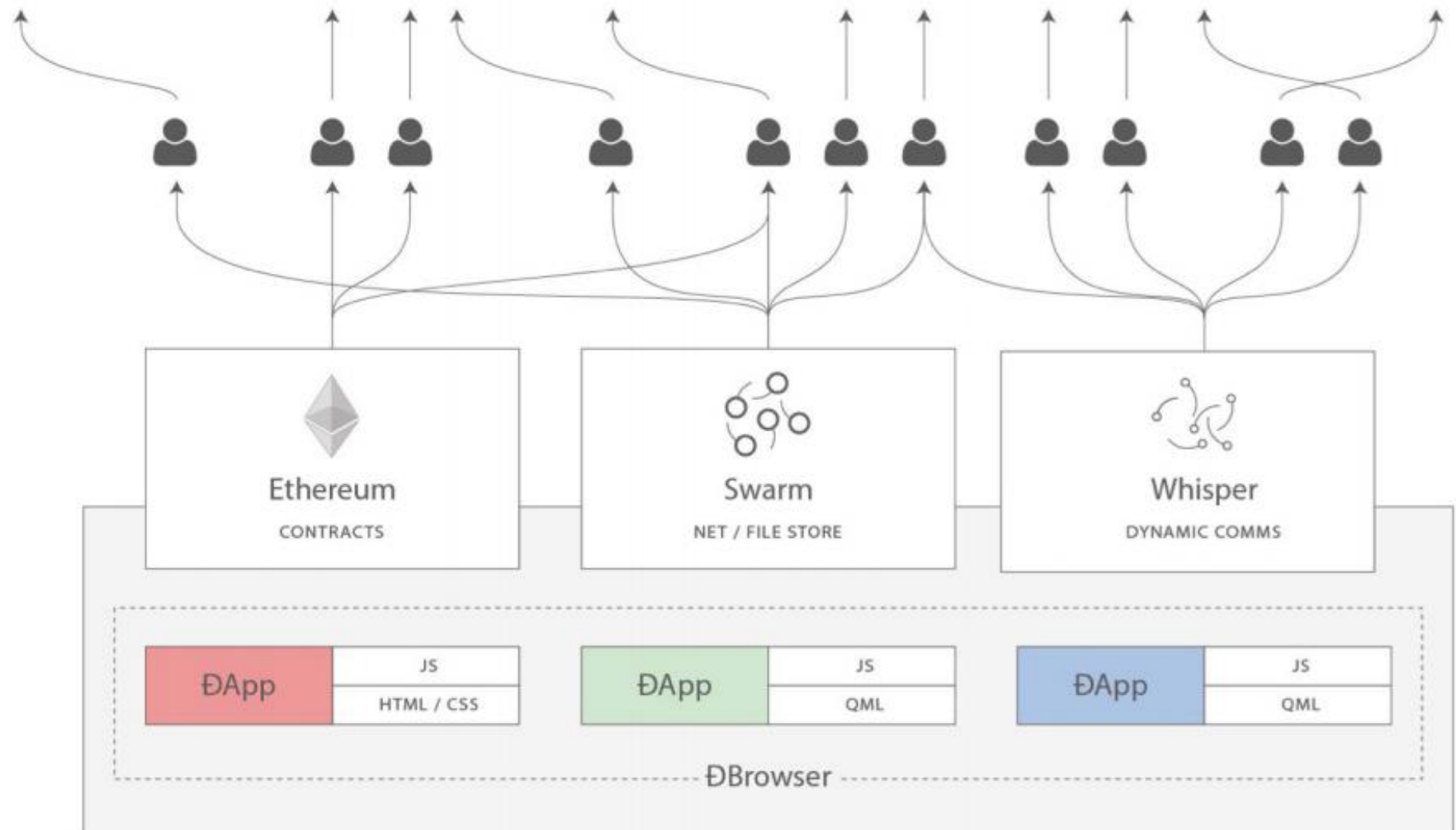
The Third Age of the Internet- *evolving WWW*

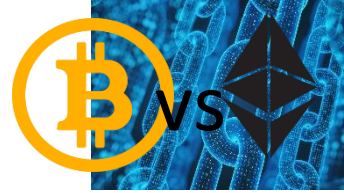
- Ethereum web3js JavaScript library which bridges JavaScript applications that run in your browser with the Ethereum blockchain.
- The web3.js library also includes an interface to a P2P storage network called **Swarm** and a P2P messaging service called **Whisper**.
- With these three components included in a JavaScript library running in your web browser, developers have a full application development suite that allows them to build web3 DApps



The Third Age of the Internet- *evolving WWW*

Web3: A suite of decentralized application components for the next evolution of the web



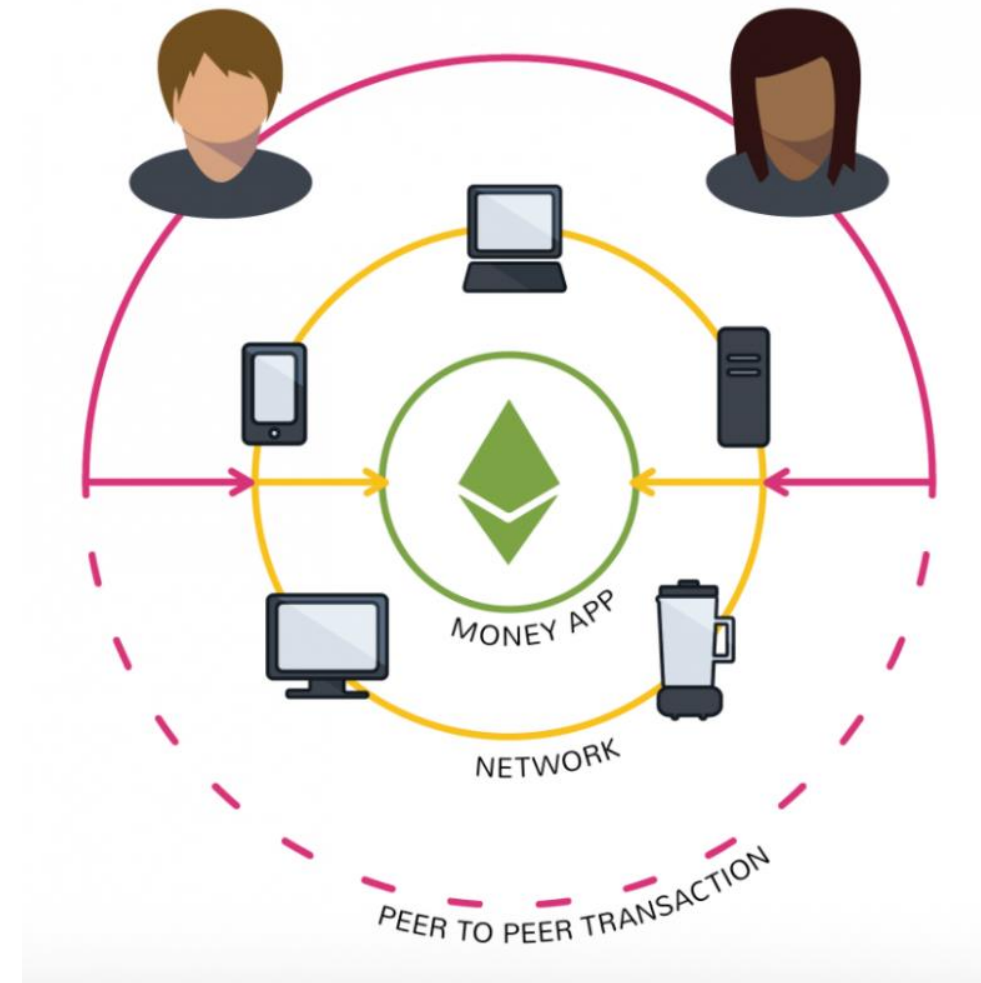


Decentralized Applications

- Ethereum enables developers to build and deploy decentralized applications. A **decentralized application** or **Dapp** serve some particular purpose to its users.
- Decentralized applications are made up of code that runs on a blockchain network, they are **not controlled** by any individual or central entity.
- In the Ethereum, instead of mining for bitcoin, miners work to earn Ether, a type of crypto token that fuels the network. Beyond a tradeable cryptocurrency, Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.
- There is a second type of token that is used to pay miners fees for including transactions in their block, it is called gas, and every smart contract execution requires a certain amount of gas to be sent along with it to entice miners to put it in the blockchain.

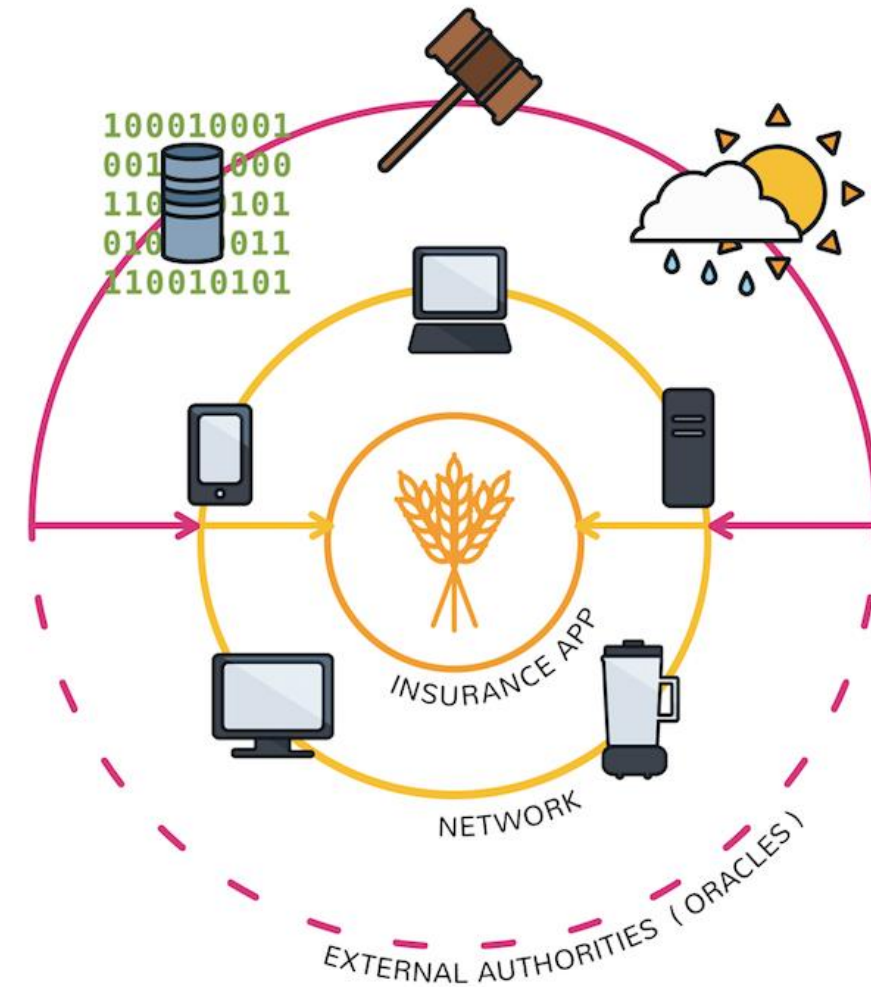
Decentralized Applications

- dapps into three types:
 - apps that manage money {exchange ether as a way to settle a contract with another user}
 - apps where money is involved (but also requires another piece) {app mixes money with information from outside the blockchain. e.g. crop insurance}
 - apps in the “other” category, which includes voting and governance systems.



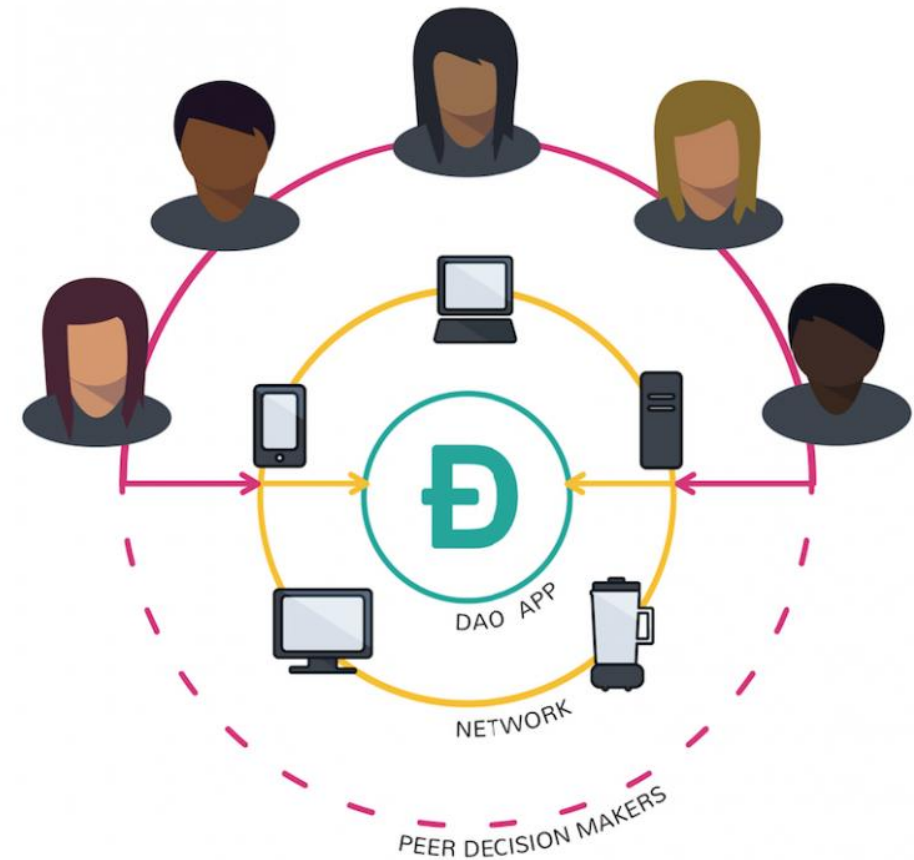
Crop Insurance

- Crop Insurance application is dependent on an outside weather feed. (Say a farmer buys a derivative that automatically pays out if there's a drought that impacts his work.)
- To execute, these smart contracts rely on so-called “**oracles**” that relay up-to-date information about the outside world. (Though, it's worth noting that some developers *are skeptical* that this use case can be done in a decentralized way.)



DAO

- Decentralized autonomous organizations are one particularly ambitious breed of dapp
- The goal is form a leaderless company, program rules at the beginning about how members can vote and how to release company funds





Building applications on Ethereum

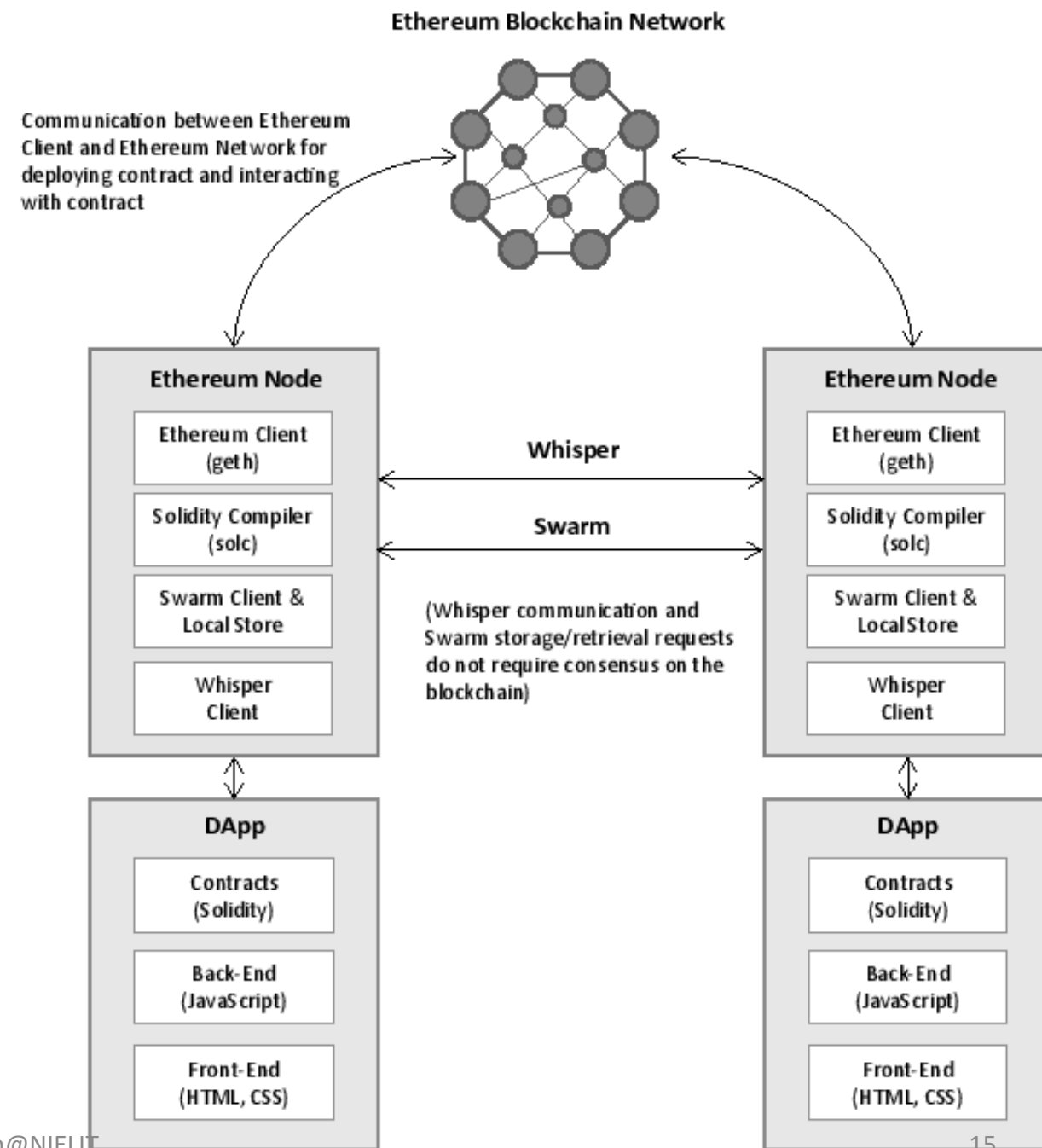
- Cryptocurrency wallets that let you make cheap, instant payments with ETH or other assets
- Financial applications that let you borrow, lend, or invest your digital assets
- Decentralized markets, that let you trade digital assets, or even trade “predictions” about events in the real world
- Games where you own in-game assets, and can even make real money
- And much, much more.



Benefits of a decentralized Ethereum Platform

- **Immutability** – A third party cannot make any changes to data.
- **Corruption & tamper proof** – Apps are based on a network formed around the principle of consensus, making censorship impossible.
- **Secure** – With no central point of failure and secured using cryptography, applications are well protected against hacking attacks and fraudulent activities.
- **Zero downtime** – Apps never go down and can never be switched off.

Blockchain Components



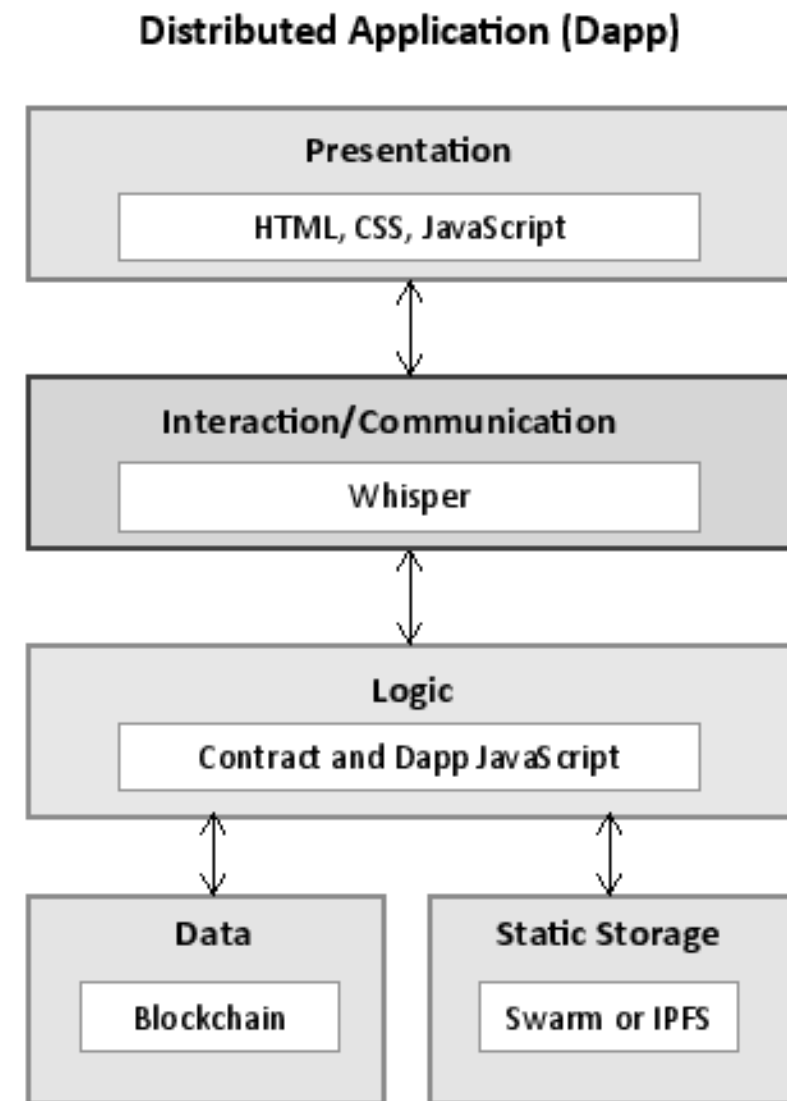
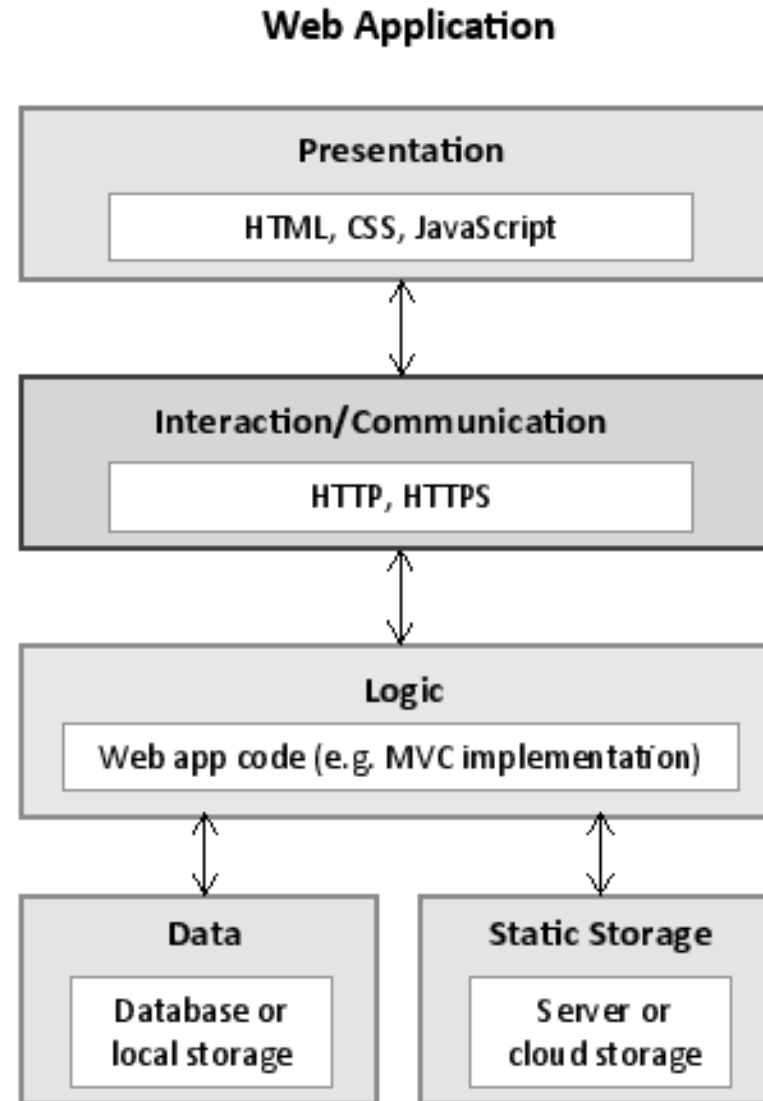


Cloud/Web Application vs Dapp

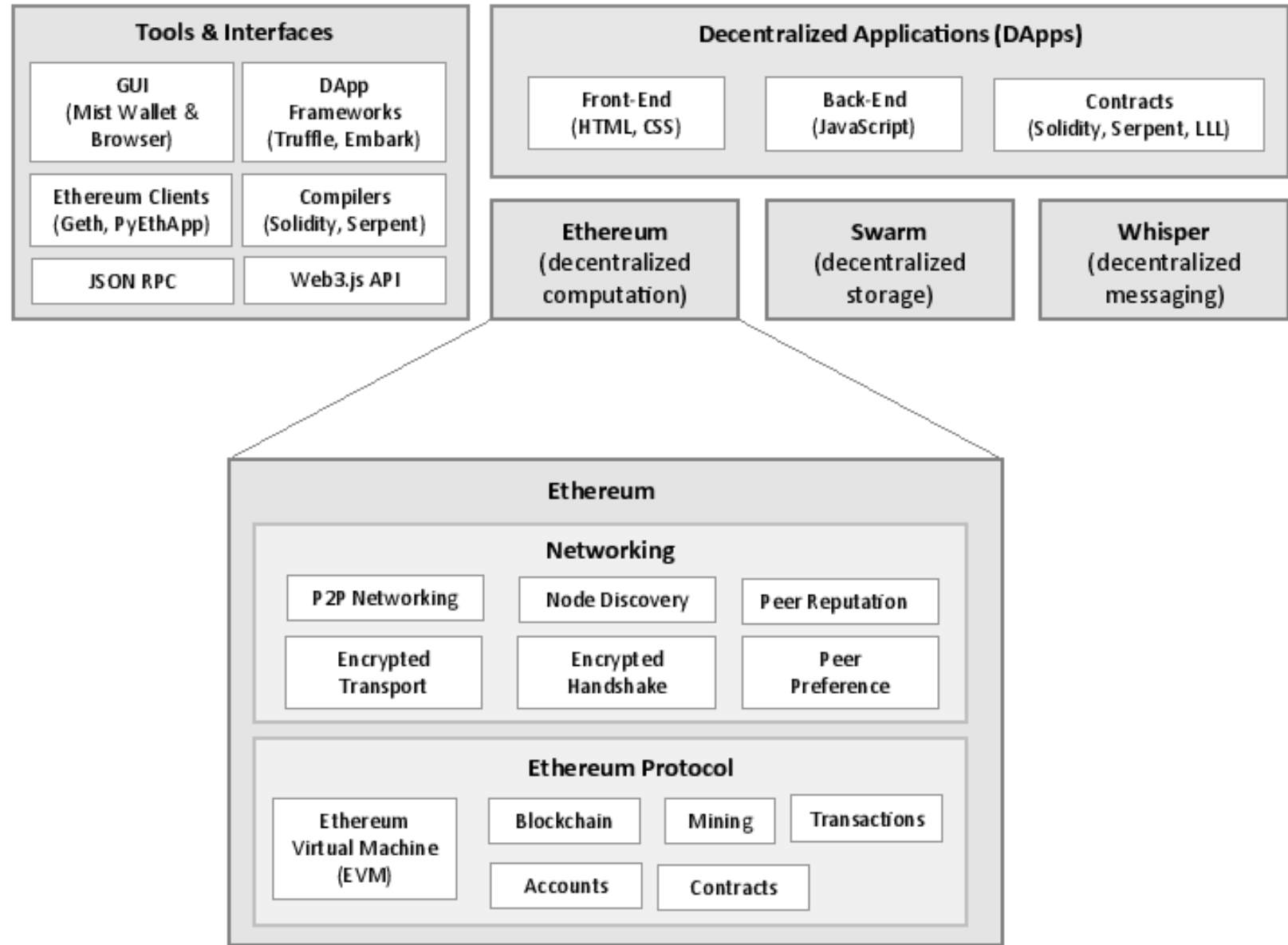
Category	Cloud/Web Application	Dapp
Logic	Web app code (e.g. model, view, controller implementation)	Contract and Dapp JavaScript
Data	Database or local storage	Blockchain
Presentation	HTML, CSS, JavaScript	HTML, CSS, JavaScript
Static Storage	Server or cloud storage	Swarm or IPFS
Interaction/ Communication	HTTP, HTTPS	Whisper



Web application vs Distributed Application(Dapp)

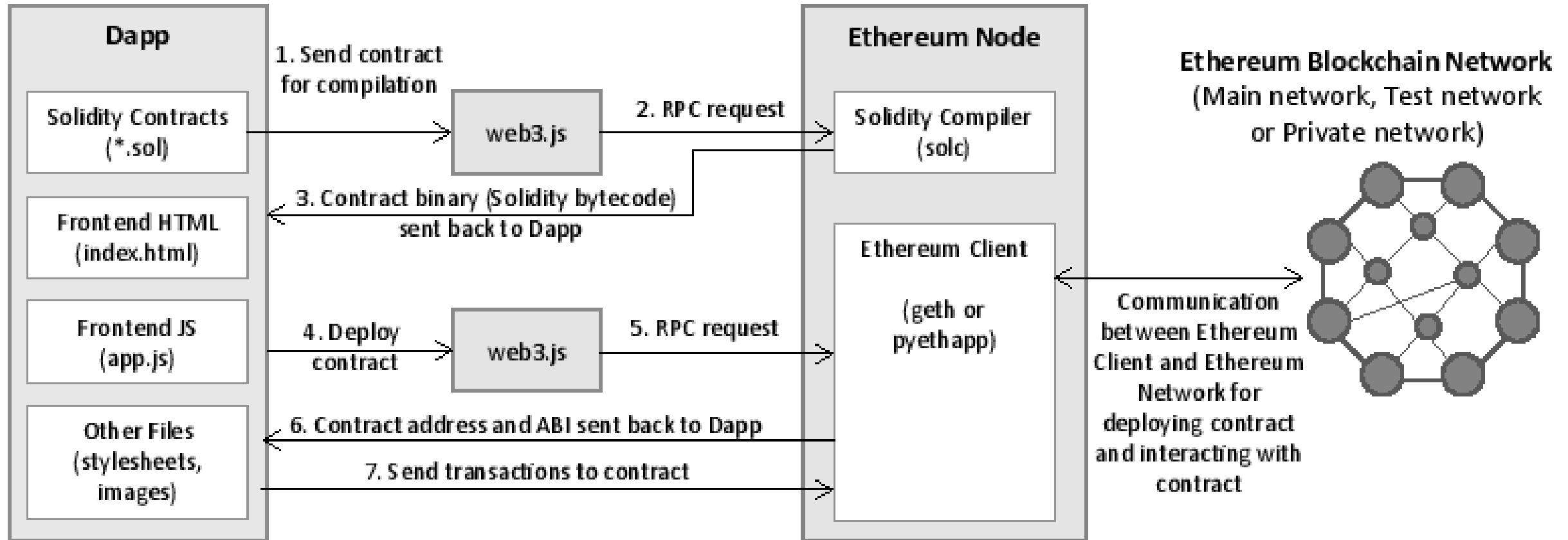


Architecture



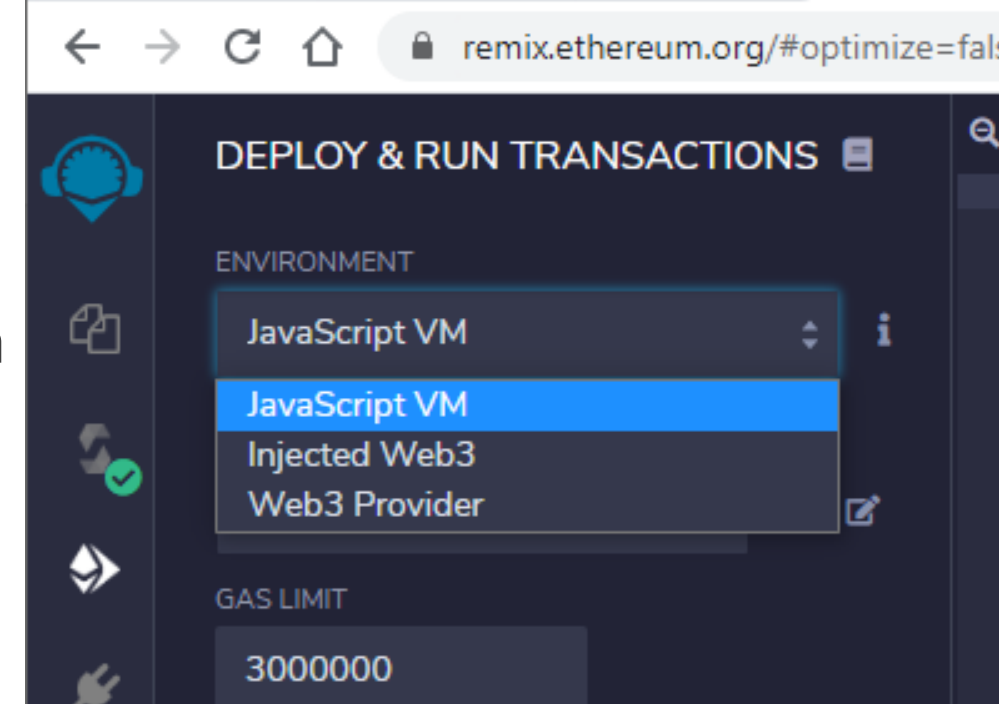


Dapp creation workflow



Environment

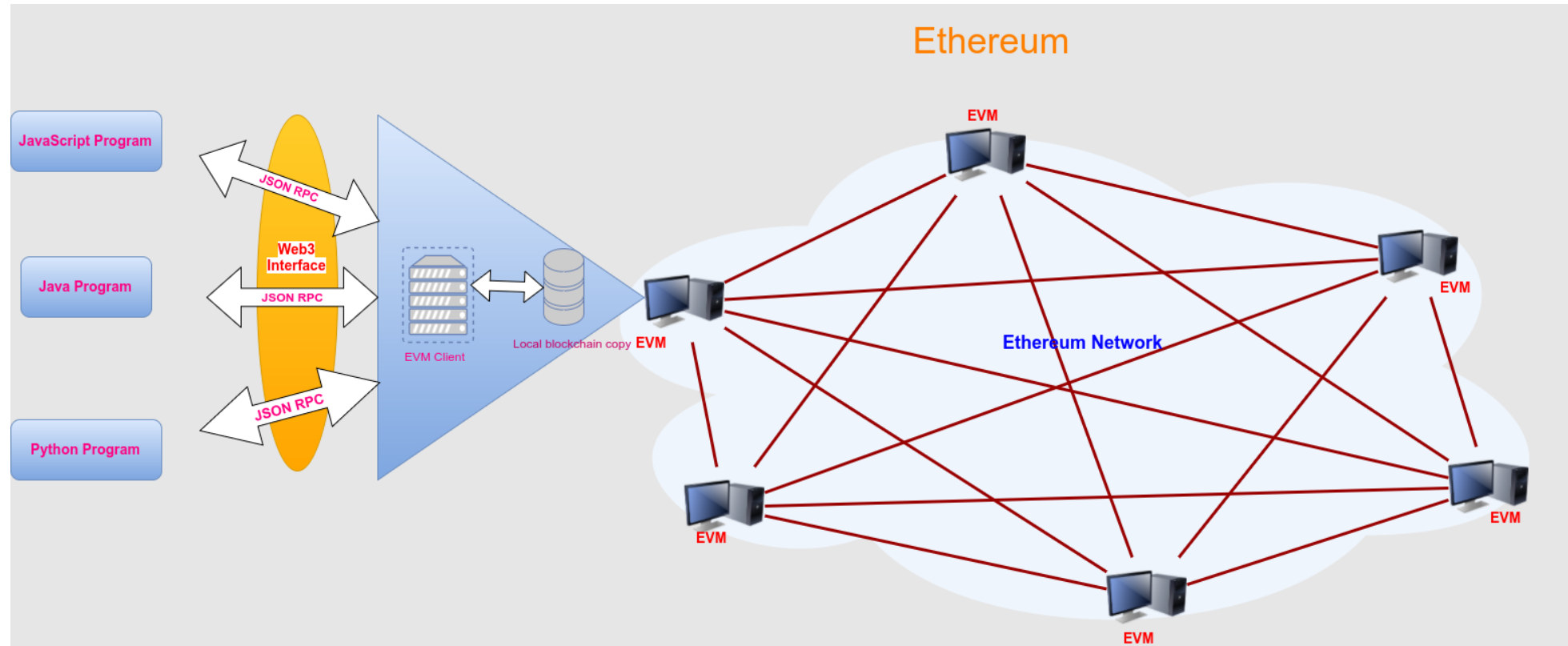
- **JavaScript VM:** All the transactions will be executed in a sandbox blockchain in the browser. This means nothing will be persisted when you reload the page. The JsVM is its own blockchain and on each reload it will start a new blockchain, the old one will not be saved.
- **Injected Provider:** Remix will connect to an injected web3 provider. **Metamask** is an example of a provider that inject web3.
- **Web3 Provider:** Remix will connect to a remote node. You will need to provide the URL to the selected provider: geth, parity or any Ethereum client.





Why to use web.js

- Developing websites or clients that interact with the blockchain - writing code that reads and writes data from the blockchain with smart contracts.





Web3.js

- Web3.js is a popular library that allows programmers to interact with the Ethereum blockchain.
- It represents a JavaScript language binding for Ethereum's JSON RPC interface, which makes it directly usable in web technology, as JavaScript is natively supported in almost all web browsers.
- Web3.js is also commonly used on the server side in Node.js applications
- Web3.js can be used to connect to the Ethereum network via any Ethereum node that allows access via HTTP. This may be a local node, a node hosted by the DApp provider, or public gateways



Web3.js

- jQuery to make Ajax calls to a web server
- Instead of using a jQuery to read and write data from a web server, you can use Web3.js to read and write to The Ethereum Blockchain.
- Web3.js talks to The Ethereum Blockchain with JSON RPC("Remote Procedure Call" protocol).
- Ethereum is a peer-to-peer network of nodes that stores a copy of all the data and code on the blockchain.
- Web3.js allows us to make requests to an individual Ethereum node with JSON RPC in order to read and write data to the network.



Dependencies with Web3.js

- **Node Package Manager (NPM)**
 - \$ node -v
- **Web3.js Library**
 - \$ npm install web3 (install the Web3.js library with NPM)
- **RPC URL**
 - In order to connect to an Ethereum node with JSON RPC on the Main Net, we need access to an Ethereum node. One way is - run your own Ethereum node with [Geth](#) or Parity.(it has lot of data requirement)



Frontend

JavaScript
(React.js)
+ HTML
+ CSS

Web3.js

Ethereum Wallets

MetaMask

MyEtherWallet

Mist

Others

Blockchain

Ganache

Ropsten

Kovan

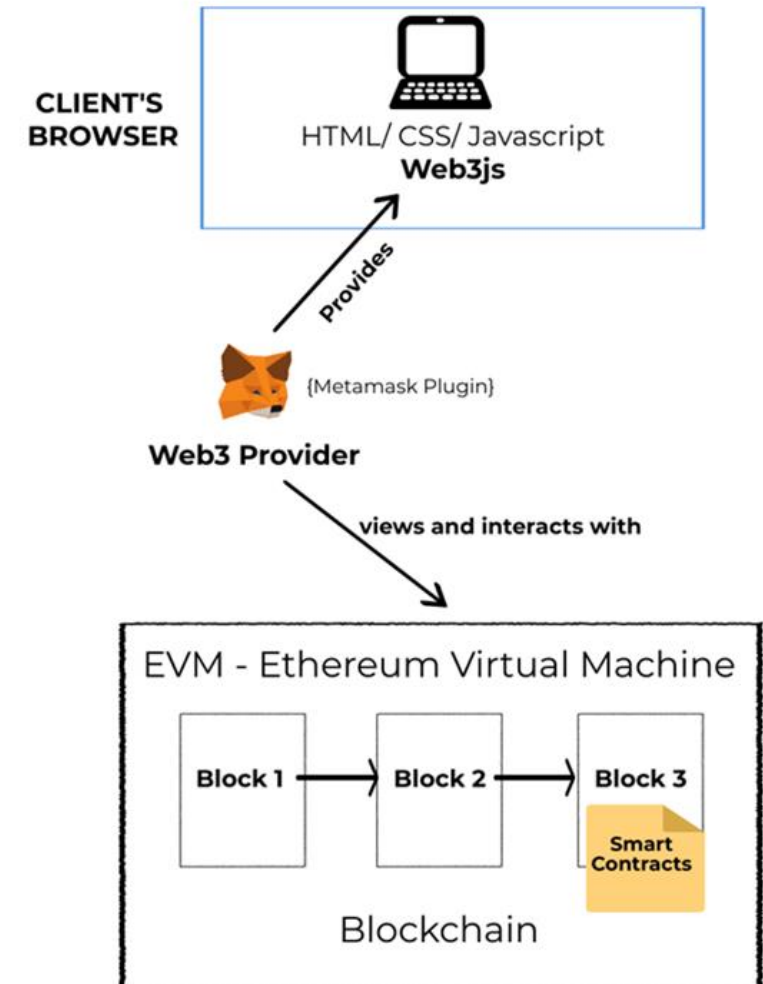
Rinkeby

Ethereum Node

Building dApp

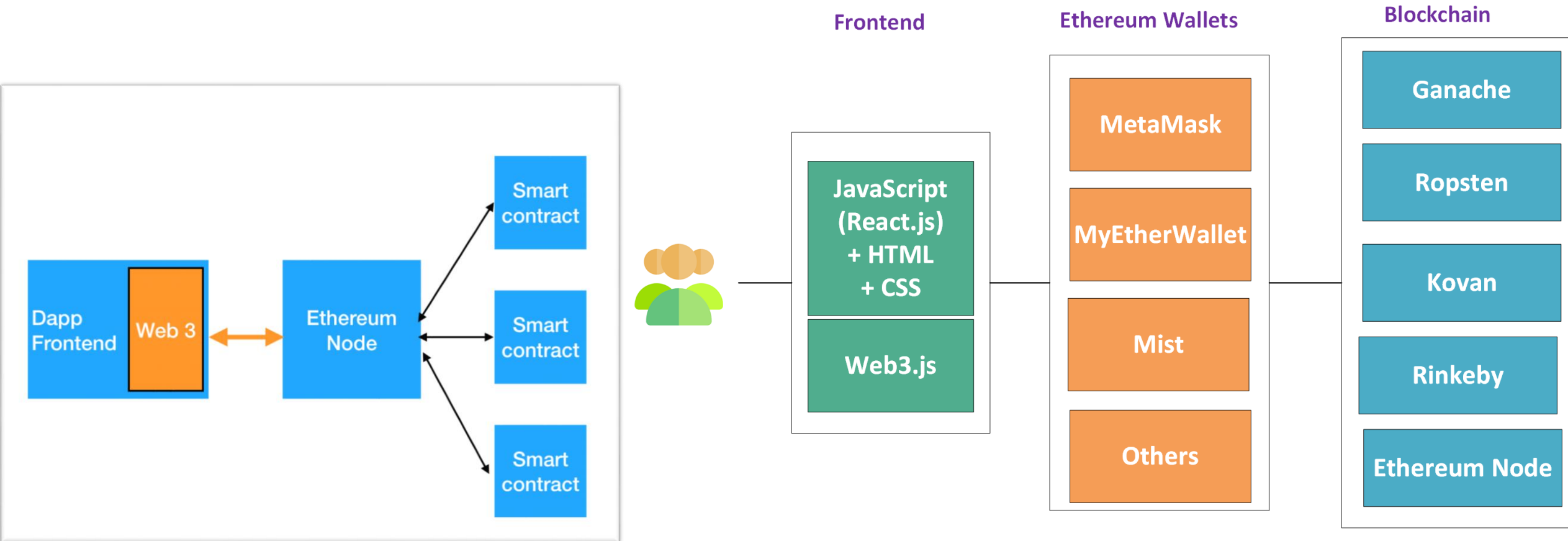
Metamask

- One common way of integrating a web browser application with Ethereum is to use the **Metamask** browser extension in combination with Web3.js.
- Metamask is an in-browser Ethereum wallet that injects a Web3 provider object into the browser. A Web3 provider is a data-structure providing a link to publicly accessible Ethereum nodes.
- Using Metamask allows users to manage private keys and sign transactions within their web browser.
- Using Metamask in combination with Web3.js, in a web interface, provides a convenient way to interact with the Ethereum network.





Basic workflow of typical dApp





The HTML code

- `const Web3 = require('web3');` *//include the library*
//web application to connect via the Metamask browser extension
- `const web3 = new Web3(Web3.givenProvider, null, {});`
//Metamask injects a Web3 provider into the browser.
- `let contract = new web3.eth.Contract(abi, address);` *//initialize contract*
//use object for sending transactions to the contract, for example
- `contract.methods.transfer(toAddress, value).send({from: myAddress});`



You're sharing your screen

Email: FutureSkill-1 x Applied_ML_and_I x Slack | north-regio x Remix - Ethereum x localhost:8000/pa x part1.html x New Tab x

File | F:/blockchain/WEB3/part1.html

Responsive 479 x 509 100% Online

Elements Console Sources Network Performance Memory

top Filter

jQuery.Deferred exception: web3 is not defined ReferenceError: web3 is not defined jquery-3.3.1.slim.min.js:2

at HTMLDocument.<anonymous> (file:///F:/blockchain/WEB3/part1.html:77:18)

at 1 (https://code.jquery.com/jquery-3.3.1.slim.min.js:2:29567)

at c (https://code.jquery.com/jquery-3.3.1.slim.min.js:2:29869) undefined

Uncaught ReferenceError: web3 is not defined jquery-3.3.1.slim.min.js:2

at HTMLDocument.<anonymous> (part1.html:77)

at 1 (jquery-3.3.1.slim.min.js:2)

at c (jquery-3.3.1.slim.min.js:2)

Console What's New x

Highlights from the Chrome 84 update

The new Issues tab

The Issues tab aggregates warnings from the browser in a structured, aggregated, and actionable way, links to affected resources within DevTools, and provides guidance on how to fix the issues.

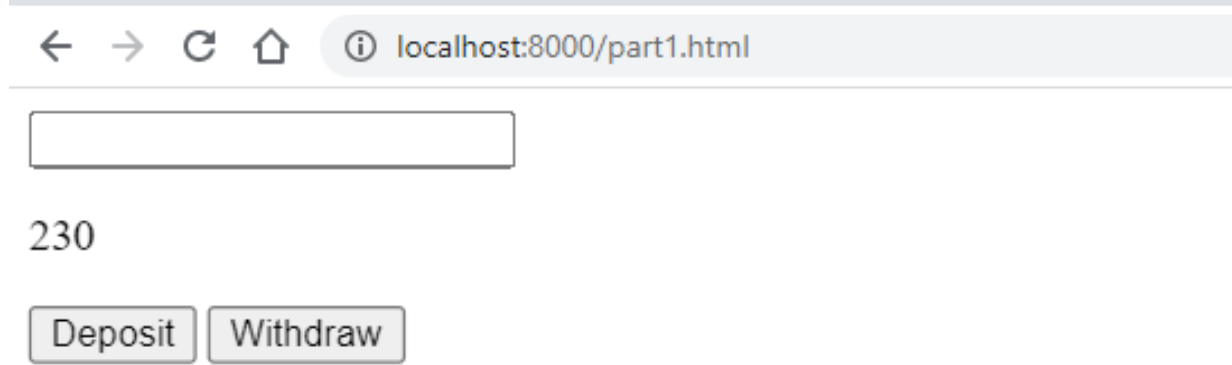
New accessibility information in the Inspect Mode tooltip

The tooltip now indicates whether an element has an accessible name and role and is keyboard-focusable.

Performance panel updates

Type here to search

14:47 27-07-2020



<div>

<input type="text" id="amount">

<p id='balance' ></p>

<button id = 'deposit'>Deposit</button>

<button id = 'withdraw'> Withdraw</button>

</div>

```
<script src="https://cdn.jsdelivr.net/gh/ethereum/web3.js@1.0.0-beta.36/dist/web3.min.js"></script>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" crossorigin="anonymous"></script>
<script>
    var contract ;
    $(document).ready(function()
    {
        var address = "0x-----112";
        var abi = [ .....];
        web3=new Web3(web3.currentProvider);
        contract = new web3.eth.Contract (abi, address );
        contract.methods.getBalance().call().then(function(bal)
        {
            $('#balance').html(bal);
        })
    })
</script>
```

localhost:8000/part2.html

Bank dApp

Current Balance :

230

Amount

25

Deposit

Withdraw

MetaMask Notification

POA Sokol Testnet

Sarwan

→

0x4eca...41...

DEPOSIT

0

DETAILS

DATA

GAS FEE

0

EDIT

No Conversion Rate Available

AMOUNT + GAS FEE

TOTAL

0

No Conversion Rate Available

Reject

Confirm



```
$('#withdraw').click(function()  
{  
    var amt =0;  
    amt = parseInt($('#amount').val());  
    web3.eth.getAccounts().then(function(accounts){  
        var acc = accounts[0];  
        return contract.methods.withdraw(amt).send({from:acc});  
    }).then(function(tx)  
    {  
        console.log(tx);  
    }).catch(function(tx)  
    {  
        console.log(tx);  
    })  
})
```



```
$('#deposit').click(function()  
{  
    var amt =0;  
    amt = parseInt($('#amount').val());  
    web3.eth.getAccounts().then(function(accounts){  
        var acc = accounts[0];  
        return contract.methods.deposit(amt).send({from:acc});  
    }).then(function(tx)  
    {  
        console.log(tx);  
    }).catch(function(tx)  
    {  
        console.log(tx);  
    })  
})  
})
```


localhost:8000/part2.html

>

Bank dApp


Current Balance :

230

Amount

Deposit

Withdraw

 **Confirmed transaction**
Transaction 24 confirmed! View on Etherscan
Google Chrome

localhost:8000/part2.html

Bank dApp

Current Balance :

255

Amount

Deposit

Withdraw



localhost:8000/part2.html

Bank dApp

Current Balance :
255

Amount

Deposit

Withdraw

MetaMask Notification

POA Sokol Testnet

Sarwan

→

0x4eca...41...

WITHDRAW

0

DETAILS

DATA

EDIT

GAS FEE

0

No Conversion Rate Available

AMOUNT + GAS FEE

TOTAL

0

No Conversion Rate Available

Reject

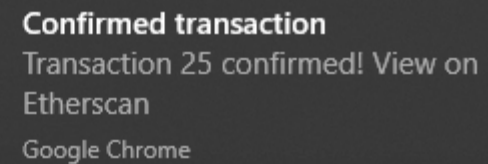
Confirm

255

45

Deposit

Withdraw





localhost:8000/part2.html

Bank dApp

Current Balance :

210

Amount

Deposit

Withdraw