Web3.js

What is web3.js ?

Web3.js is a JavaScript library that provides an interface for interacting with the Ethereum blockchain and building decentralized applications (dApps).

Web3.js allows developers to interact with the Ethereum blockchain, including creating accounts, sending transactions, and querying blockchain data.

How is it different from web2 ?

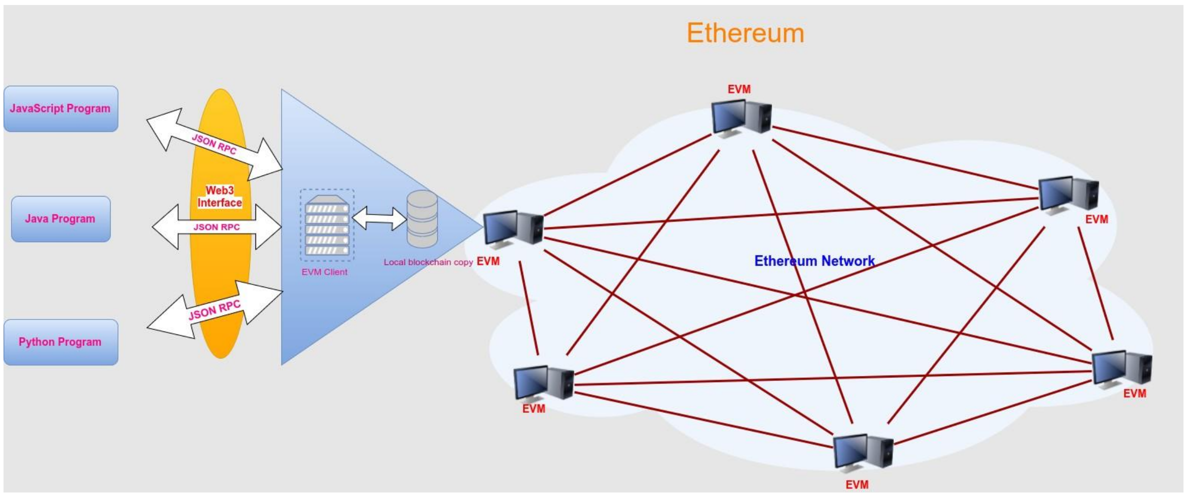
Web3, often referred to as the decentralized web, differs from Web2 (the traditional web) in several key aspects.

- Centralization vs. Decentralization: Web2 relies on centralized servers and services controlled by a few entities, while Web3 is based on decentralization, utilizing blockchain technology to distribute data and control across a network of participants.
- User Ownership and Control: In Web2, users typically have limited ownership and control over their data, which is stored on centralized servers. Web3 emphasizes user ownership and control, allowing individuals to own their data, manage their digital identities, and decide how their information is shared.

Basic requirements for Web3 development :

- **Web3.js:**
  - A popular library that will enable our front-end app to talk to the Blockchain.
- **Metamask:**
  - A popular Web3 Wallet.
- **http server:**
  - To serve our front-end application.

E.g.



Web3.js Vs Ether.js

➢ Web3.js

- Web3.js is a JavaScript library for interacting with the Ethereum blockchain.
- It provides a wide range of functionalities for Ethereum blockchain interaction, including account management, smart contract interaction, and event handling.

- Web3.js supports multiple Ethereum networks, such as the mainnet, testnets, and private networks.
- It is a more mature and widely adopted library, with extensive documentation and community support.
- Web3.js is not limited to Ethereum and can be used to interact with other blockchain networks that support the Web3 protocol.

➢ Ether.js

- Ether.js is a JavaScript library specifically designed for interacting with the Ethereum blockchain.
- It focuses on providing a lightweight and modular approach for Ethereum blockchain interaction.
- Ether.js has a narrower scope compared to Web3.js and primarily focuses on essential functionalities like account management, transaction handling, and contract deployment.
- It is developed by the Ethereum Foundation and aligns closely with Ethereum's standards and best practices.

- Ether.js aims to be a more developer-friendly library with a simpler API and improved performance.

How to run JavaScript code in web3.js

To run JavaScript code in Web3.js, you need to get Web3.js into your project.

This can be done using the following methods:

Dependencies:

There are a few dependencies that will help you start developing with Web3.js.

Node Package Manager (NPM):

The first dependency we need is Node Package Manager, or NPM, which comes with Node.js.

To check if you have Node already installed, go to your terminal and type:

```
node -v
```

Web3.js Library:

You can install the Web3.js library with NPM in your terminal like this:

```
npm install web3
```

Infura RPC URL:

To connect to an Ethereum node with JSON RPC on the mainnet, we need access to an Ethereum node.

There are a few ways you could do this:

One way you could do this is by running your own

Ethereum node with Geth or Parity.

But this requires you to download a lot of data from the Blockchain and keep it in sync.

This is a huge headache if you've never tried to do this before.

Mostly for convenience, you can use Infura to access an Ethereum node without having to run one yourself.

Infura is a service that provides a remote Ethereum node for free.

All you need to do is sign up, and obtain an API key and the RPC URL for the network you want to connect to.

Once you've signed up, your Infura RPC URL should look like this:

https://mainnet.infura.io/v3/e7f78a69f3d9444fa599d8bb 856c26ef

Checking Account Balances with Web3.js.

Web3.js can be used in the front-end and back-end to check account balances, read data from the Blockchain, make transactions, and even deploy smart contracts.

Now that all of your dependencies are installed, let's walk through how you can check your account balance with web3.js.

First, you should fire up the Node console in your terminal like this:

```
node
```

Now you've got the Node console open! Inside the Node console, you can require Web3.js like this:

const Web3 = require('web3');

Now, you have access to a variable where you can create a new webg connection.

Before we generate a web3 connection, we must first assign the Infura URL to a variable like this:

const rpcURL=https://mainnet.infura.io/v3/c7f78a69f3d9444fa599d8bb856c26ef

Now you can instantiate a Webg connection like this:

const web3 = new Web3(rpcURL);

Now, you have a live web3 connection that will allow to talk to the Ethereum mainnet.

Let's use this connection to check the account balance for this

account: 0x105cb19ba40384a8f2985816DA7883b076969cA7.

We can see how much Ether this account holds by checking its balance with **web3.eth.getBalance().**

The first step, let's assign the address to a variable:

```
const address =
"0x105cb19ba40384a8f2985816DA7883b076969cA7":
```

Let's now check the account balance like this:

```
web3.eth.getBalance(address, (err, wei) => { balance =
web3.utils.fromWei(wei, 'ether')
});
console.log(balance);
```

And that's it! That's how you check your account balance with web3.js.

Here is a summary of the code we wrote in node console is

as below:

```
const Web3 = require('web3')
const rpcURL = ''; // Your RPC URL goes here
const web3 = new Web3(rpcURL)
const address = ''; // Your account address goes here
web3.eth.getBalance(address, (err, wei) => {
balance = web3.utils.fromWei(wei, 'ether')
});
Console.log(balance);
```

First, we use to check the balance by calling **web3.eth.getBalance(),** which accepts a callback function with two arguments, the balance itself. an error and

We'll ignore the error argument for now, and reference the balance with the **wei** argument.

Ethereum expresses balances in **wei**, which is the smallest subdivision of ether, kind of like a tiny penny.

We can convert this balance to ether with **web3.utils.fromWei(wei, 'ether');.**

Read data from the Smart Contract with Web3.js

Before reading the data from the Smart Contract with Web3.js.

We need to connect to an Ethereum node with JSON RPC on the mainnet, we need access to an Ethereum node in the following way.

First, you should fire up the Node console in your terminal like this:

```
node
```

Now you have got the Node console open! Inside the Node console, you can require web3.js like this:

```
const Web3 = require('web3');
```

Now, you have access to a variable where you can create a new webg connection.

Before we generate a webs connection, we must first assign the Infura URL to a variable like this:

```
const rpcURL = "https://mainnet.infura.io/v3/e-f-8a6of3d0444fa500d8bb856026ef":
```

Make sure that you replace your own **INFURA API KEY** obtained from Infura service. actual

Now you can instantiate a Web3 connection like this:

const web3 = new Web3(rpcURL);

Now, you have a live web3 connection that will allow you to talk to the Ethereum mainnet.

In order to read data from smart contracts with web3.js, we need two things:

- A JavaScript representation of the smart contract we want to interact with.
- A way to call the functions on the smart contract when reading the data.

We can get a JavaScript representation of an Ethereum smart contract with the **"web3.eth.Contract()"** function.

This function expects two arguments:

one for the smart contract ABI and another one for the smart contract address.

A smart contract ABI stands for "Abstract Binary Interface", and it's a JSON array that describes how a specific smart contract works.

Here is an example of an ABI:

```
const abi =

[{"inputs":[],"stateMutability":"nonpayable","type":"
constructor"},{"anonymous":false,"inputs":[{"indexed
":true,"internalType":"address","name":"owner","typ
e":"address"},{"indexed":true,"internalType":"addres
s","name":"approved","type":"address"},{"indexed":t
rue,"internalType":"uint256","name":"tokenId","type
":"uint256"}],"name":"Approval","type":"event"},{"a
nonymous":false,"inputs":[{"indexed":true,"internalT
ype":"address","name":"owner","type":"address"},{"
indexed":true,"internalType":"address","name":"ope
rator","type":"address"},{"indexed":false,"internalTy
pe":"bool","name":"approved","type":"bool"}],"nam
e":"ApprovalForAll","type":"event"},{"anonymous":fa
lse,"inputs":[{"indexed":false,"internalType":"uint256
","name":"_fromTokenId","type":"uint256"},{"indexe
d":false,"internalType":"uint256","name":"_toTokenI
d","type":"uint256"}],"name":"BatchMetadataUpdat
e","type":"event"},{"anonymous":false,"inputs":[{"in
dexed":false,"internalType":"uint256","name":"_toke
nId","type":"uint256"}],"name":"MetadataUpdate","t
ype":"event"},{"anonymous":false,"inputs":[{"indexe
d":true,"internalType":"address","name":"previousO
wner","type":"address"},{"indexed":true,"internalTyp
```

e":"address","name":"newOwner","type":"address"}
],"name":"OwnershipTransferred","type":"event"},{"
anonymous":false,"inputs":[{"indexed":true,"internal
Type":"address","name":"from","type":"address"},{"i
ndexed":true,"internalType":"address","name":"to","
type":"address"},{"indexed":true,"internalType":"uint
256","name":"tokenId","type":"uint256"}],"name":"T
ransfer","type":"event"},{"inputs":[{"internalType":"a
ddress","name":"to","type":"address"},{"internalTyp
e":"uint256","name":"tokenId","type":"uint256"}],"n
ame":"approve","outputs":[],"stateMutability":"nonp
ayable","type":"function"},{"inputs":[{"internalType":
"address","name":"owner","type":"address"}],"name
":"balanceOf","outputs":[{"internalType":"uint256","
name":"","type":"uint256"}],"stateMutability":"view"
,"type":"function"},{"inputs":[{"internalType":"uint25
6","name":"tokenId","type":"uint256"}],"name":"get
Approved","outputs":[{"internalType":"address","na
me":"","type":"address"}],"stateMutability":"view","
type":"function"},{"inputs":[{"internalType":"address
","name":"owner","type":"address"},{"internalType":
"address","name":"operator","type":"address"}],"na
me":"isApprovedForAll","outputs":[{"internalType":"
bool","name":"","type":"bool"}],"stateMutability":"vi
ew","type":"function"},{"inputs":[{"internalType":"ad
dress","name":"recipient","type":"address"},{"intern

alType":"string","name":"tokenURI","type":"string"}],
"name":"mintNFT","outputs":[{"internalType":"uint2
56","name":"","type":"uint256"}],"stateMutability":"
nonpayable","type":"function"},{"inputs":[],"name":"
name","outputs":[{"internalType":"string","name":""
,"type":"string"}],"stateMutability":"view","type":"fu
nction"},{"inputs":[],"name":"owner","outputs":[{"in
ternalType":"address","name":"","type":"address"}],
"stateMutability":"view","type":"function"},{"inputs"
:[{"internalType":"uint256","name":"tokenId","type":
"uint256"}],"name":"ownerOf","outputs":[{"internalT
ype":"address","name":"","type":"address"}],"state
Mutability":"view","type":"function"},{"inputs":[],"n
ame":"renounceOwnership","outputs":[],"stateMuta
bility":"nonpayable","type":"function"},{"inputs":[{"i
nternalType":"address","name":"from","type":"addre
ss"},{"internalType":"address","name":"to","type":"a
ddress"},{"internalType":"uint256","name":"tokenId"
,"type":"uint256"}],"name":"safeTransferFrom","outp
uts":[],"stateMutability":"nonpayable","type":"functi
on"},{"inputs":[{"internalType":"address","name":"fr
om","type":"address"},{"internalType":"address","na
me":"to","type":"address"},{"internalType":"uint256"
,"name":"tokenId","type":"uint256"},{"internalType":
"bytes","name":"data","type":"bytes"}],"name":"safe
TransferFrom","outputs":[],"stateMutability":"nonpa

yable","type":"function"},{"inputs":[{"internalType":"
address","name":"operator","type":"address"},{"inte
rnalType":"bool","name":"approved","type":"bool"}]
,"name":"setApprovalForAll","outputs":[],"stateMuta
bility":"nonpayable","type":"function"},{"inputs":[{"i
nternalType":"bytes4","name":"interfaceId","type":"
bytes4"}],"name":"supportsInterface","outputs":[{"in
ternalType":"bool","name":"","type":"bool"}],"state
Mutability":"view","type":"function"},{"inputs":[],"n
ame":"symbol","outputs":[{"internalType":"string","n
ame":"","type":"string"}],"stateMutability":"view","t
ype":"function"},{"inputs":[{"internalType":"uint256"
,"name":"tokenId","type":"uint256"}],"name":"token
URI","outputs":[{"internalType":"string","name":"","t
ype":"string"}],"stateMutability":"view","type":"func
tion"},{"inputs":[{"internalType":"address","name":"f
rom","type":"address"},{"internalType":"address","n
ame":"to","type":"address"},{"internalType":"uint25
6","name":"tokenId","type":"uint256"}],"name":"tra
nsferFrom","outputs":[],"stateMutability":"nonpayabl
e","type":"function"},{"inputs":[{"internalType":"add
ress","name":"newOwner","type":"address"}],"name
":"transferOwnership","outputs":[],"stateMutability":
"nonpayable","type":"function"}]

I know that seems like a really long, unbroken array.

Don't worry if it looks overwhelming.

This example is the ABI for the OmiseGo token, which implements the ERC-20 token standard. You can find more details about this token, including its ABI and address on Etherscan.

We'll use this smart contract ABI for the rest of the examples.

While we're here, I'll go ahead and store the address to the OMG token from the Ethereum mainnet:

```
const address =
"Oxd26114ed6FE289AccF82350c8d8487fedB8A0C07":
```

Now that we have both of these values assigned, we can create a complete JavaScript representation of the OMG token smart contract like this:

```
const contract = new web3.eth.Contract(abi, address);
```

Reading data from the smart contract by calling its functions.

All of the smart contract functions are listed under the **contract.methods** namespace within the assigned Webg contract.

For example, we can,

call **contract.methods.myFunction()** if the contract implements **myFunction().**

Great! So we can theoretically call any function that the smart contract implements.

But how do we know which functions it implements?

For one, we could log **contract.methods** to the console, and see what's returned.

However, since this smart contract implements the ERC- 20 standard, we know that it implements several functions like **totalSupply(), name(), symbol(), and balanceOf().**

We can read each of those values individually, like this:

First, the total supply of all OMG tokens in existence:

```
contract.methods.totalSupply().call((err, result) => {
console.log(result)});
```

Second, the name of the OMG token:

```
contract.methods.name().call((err, result) => {
console.log(result) });
```

Last, we can check the balance for a given account.

"oxd26114cd6EE289AccF82350c8d8487fedB8A0C07".

We can check the balance for this account like this:

```
contract.methods.balanceOf('oxd26114cd6EE289
AccF82350c8d8487fedB8AoCo7').call((err, result) => {
console.log(result)});
```

And that's it! That's how you read data from smart contracts with web3.js.

Here is a summary of all the code:

```
const Web3 = require('web3');
const rpcURL =
"https://mainnet.infura.io/v3/c7f78a69f3d9444fa599
d8bb856c26ef";
const web3 = new Web3(rpeURL);
```

const abi =

[{"inputs":[],"stateMutability":"nonpayable","type":"constructor"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"owner","type":"address"},{"indexed":true,"internalType":"address","name":"approved","type":"address"},{"indexed":true,"internalType":"uint256","name":"tokenId","type":"uint256"}],"name":"Approval","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":"address","name":"owner","type":"address"},{"indexed":true,"internalType":"address","name":"operator","type":"address"},{"indexed":false,"internalType":"bool","name":"approved","type":"bool"}],"name":"ApprovalForAll","type":"event"},{"anonymous":false,"inputs":[{"indexed":false,"internalType":"uint256","name":"_fromTokenId","type":"uint256"},{"indexed":false,"internalType":"uint256","name":"_toTokenI

d","type":"uint256"}],"name":"BatchMetadataUpdat
e","type":"event"},{"anonymous":false,"inputs":[{"in
dexed":false,"internalType":"uint256","name":"_toke
nId","type":"uint256"}],"name":"MetadataUpdate","t
ype":"event"},{"anonymous":false,"inputs":[{"indexe
d":true,"internalType":"address","name":"previousO
wner","type":"address"},{"indexed":true,"internalTyp
e":"address","name":"newOwner","type":"address"}
],"name":"OwnershipTransferred","type":"event"},{"
anonymous":false,"inputs":[{"indexed":true,"internal
Type":"address","name":"from","type":"address"},{"i
ndexed":true,"internalType":"address","name":"to","
type":"address"},{"indexed":true,"internalType":"uint
256","name":"tokenId","type":"uint256"}],"name":"T
ransfer","type":"event"},{"inputs":[{"internalType":"a
ddress","name":"to","type":"address"},{"internalTyp
e":"uint256","name":"tokenId","type":"uint256"}],"n
ame":"approve","outputs":[],"stateMutability":"nonp
ayable","type":"function"},{"inputs":[{"internalType":
"address","name":"owner","type":"address"}],"name
":"balanceOf","outputs":[{"internalType":"uint256","
name":"","type":"uint256"}],"stateMutability":"view"
,"type":"function"},{"inputs":[{"internalType":"uint25
6","name":"tokenId","type":"uint256"}],"name":"get
Approved","outputs":[{"internalType":"address","na
me":"","type":"address"}],"stateMutability":"view","

type":"function"},{"inputs":[{"internalType":"address","name":"owner","type":"address"},{"internalType":"address","name":"operator","type":"address"}],"name":"isApprovedForAll","outputs":[{"internalType":"bool","name":"","type":"bool"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"address","name":"recipient","type":"address"},{"internalType":"string","name":"tokenURI","type":"string"}],"name":"mintNFT","outputs":[{"internalType":"uint256","name":"","type":"uint256"}],"stateMutability":"nonpayable","type":"function"},{"inputs":[],"name":"name","outputs":[{"internalType":"string","name":"","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"owner","outputs":[{"internalType":"address","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[{"internalType":"uint256","name":"tokenId","type":"uint256"}],"name":"ownerOf","outputs":[{"internalType":"address","name":"","type":"address"}],"stateMutability":"view","type":"function"},{"inputs":[],"name":"renounceOwnership","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"from","type":"address"},{"internalType":"address","name":"to","type":"address"},{"internalType":"uint256","name":"tokenId","type":"uint256"}],"name":"safeTransferFrom","outp

uts":[],"stateMutability":"nonpayable","type":"functi
on"},{"inputs":[{"internalType":"address","name":"fr
om","type":"address"},{"internalType":"address","na
me":"to","type":"address"},{"internalType":"uint256"
,"name":"tokenId","type":"uint256"},{"internalType":
"bytes","name":"data","type":"bytes"}],"name":"safe
TransferFrom","outputs":[],"stateMutability":"nonpa
yable","type":"function"},{"inputs":[{"internalType":"
address","name":"operator","type":"address"},{"inte
rnalType":"bool","name":"approved","type":"bool"}]
,"name":"setApprovalForAll","outputs":[],"stateMuta
bility":"nonpayable","type":"function"},{"inputs":[{"i
nternalType":"bytes4","name":"interfaceId","type":"
bytes4"}],"name":"supportsInterface","outputs":[{"in
ternalType":"bool","name":"","type":"bool"}],"state
Mutability":"view","type":"function"},{"inputs":[],"n
ame":"symbol","outputs":[{"internalType":"string","n
ame":"","type":"string"}],"stateMutability":"view","t
ype":"function"},{"inputs":[{"internalType":"uint256"
,"name":"tokenId","type":"uint256"}],"name":"token
URI","outputs":[{"internalType":"string","name":"","t
ype":"string"}],"stateMutability":"view","type":"func
tion"},{"inputs":[{"internalType":"address","name":"f
rom","type":"address"},{"internalType":"address","n
ame":"to","type":"address"},{"internalType":"uint25
6","name":"tokenId","type":"uint256"}],"name":"tra

nsferFrom","outputs":[],"stateMutability":"nonpayable","type":"function"},{"inputs":[{"internalType":"address","name":"newOwner","type":"address"}],"name":"transferOwnership","outputs":[],"stateMutability":"nonpayable","type":"function"}]

```
const address =
'oxd26114cd6EE289AccF82350c8d8487fedB8A0C07';


const contract = new web3.eth.Contract(abi, address);
contract.methods.totalSupply().call((err, result) => {
console.log(result) });
contract.methods.name().call((err, result) => {
console.log(result) });
contract.methods.symbol().call((err, result) => {
console.log(result)});


contract.methods.balanceOf('oxd26114cd6EE289AccF
82 350c8d8487fedB8A0C07').call((err, result) => {
console.log(result)});
```