Customized scan module workflow

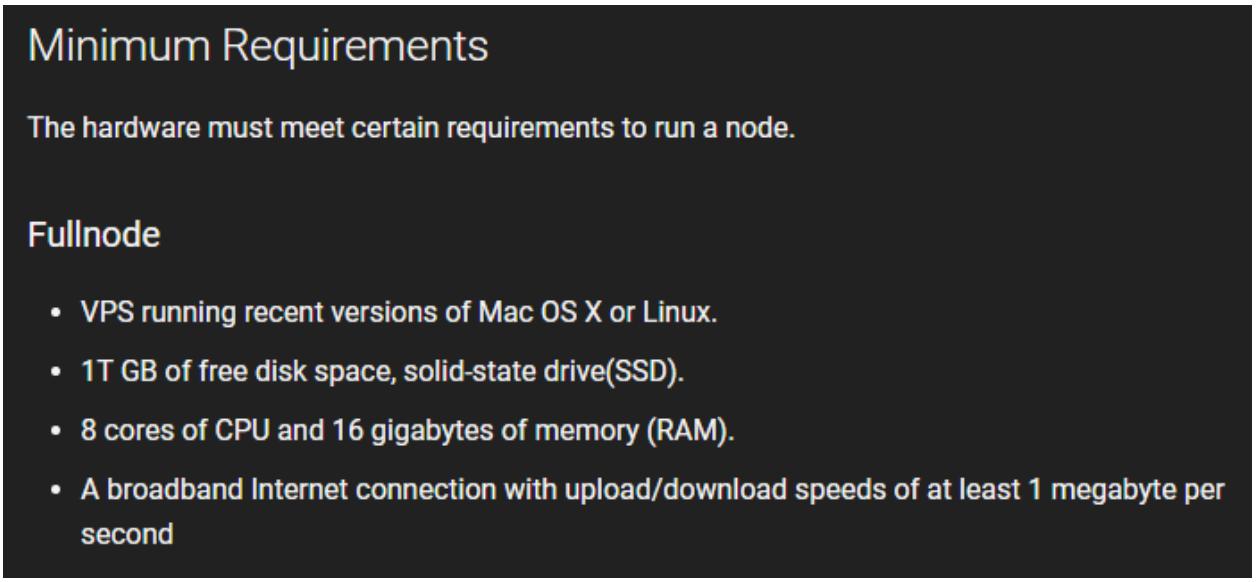
Pre-required

Binance Smart Chain is the blockchain that runs in parerell to Binance Chain. Unless BS, BSC has the great functionality, the compatibility with EVM (Ethereum Virtual Machine). So it's possible to access the BSC node via JSON-RPC like in the Ethereum.

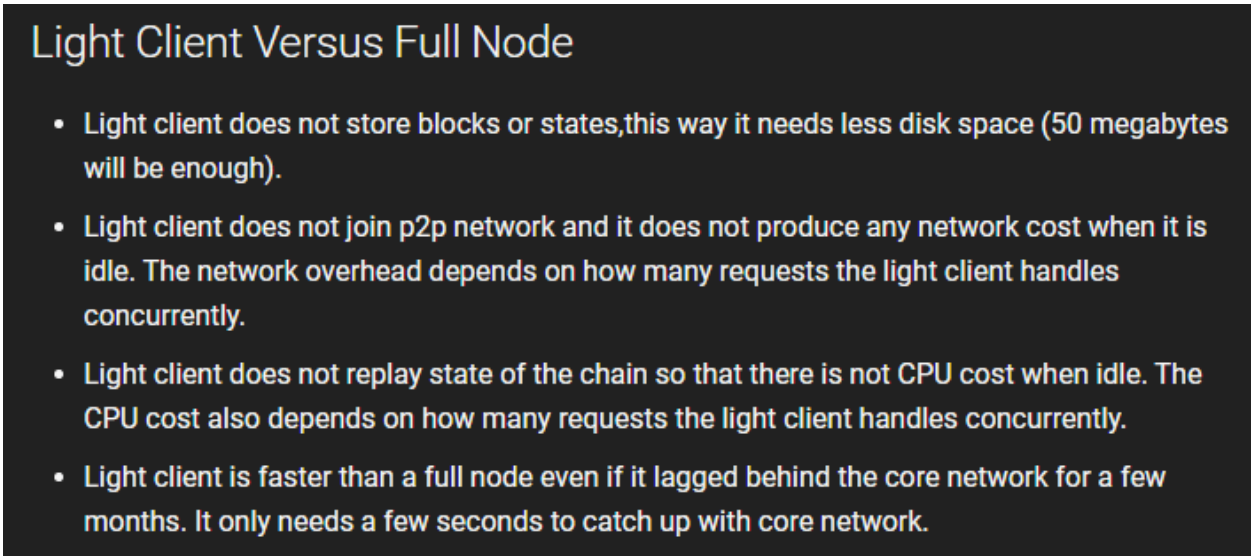
1. Possible to run a Full node to listen to and broadcast live updates on transactions, blocks, and consensus activities.

   Advantage: it's the best way to run everything in your control (Very safety and fully-control)

   Disadvantage: Very expensive (more time & effort & budget)

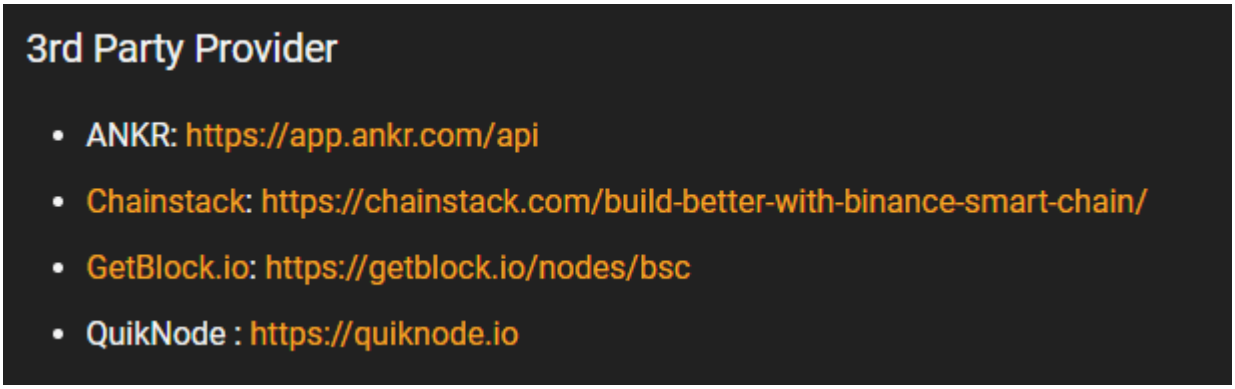
   

   If necessary, it's possible to use light client

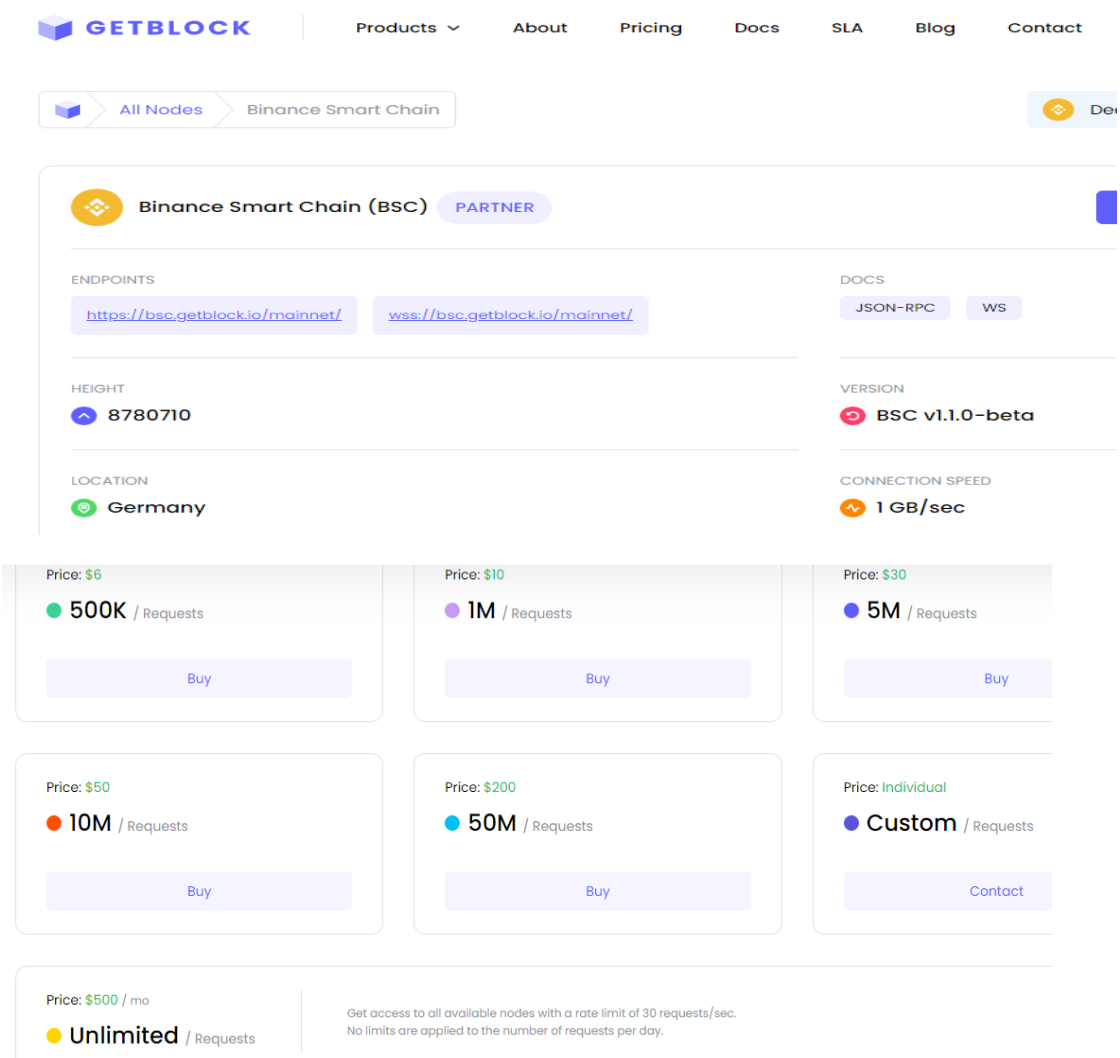
   

2. Fortunately, there are some third party provider to access the BSC node. We can explore the transaction history and blocks on the chain, via bsccan, API and node RPC interfaces.

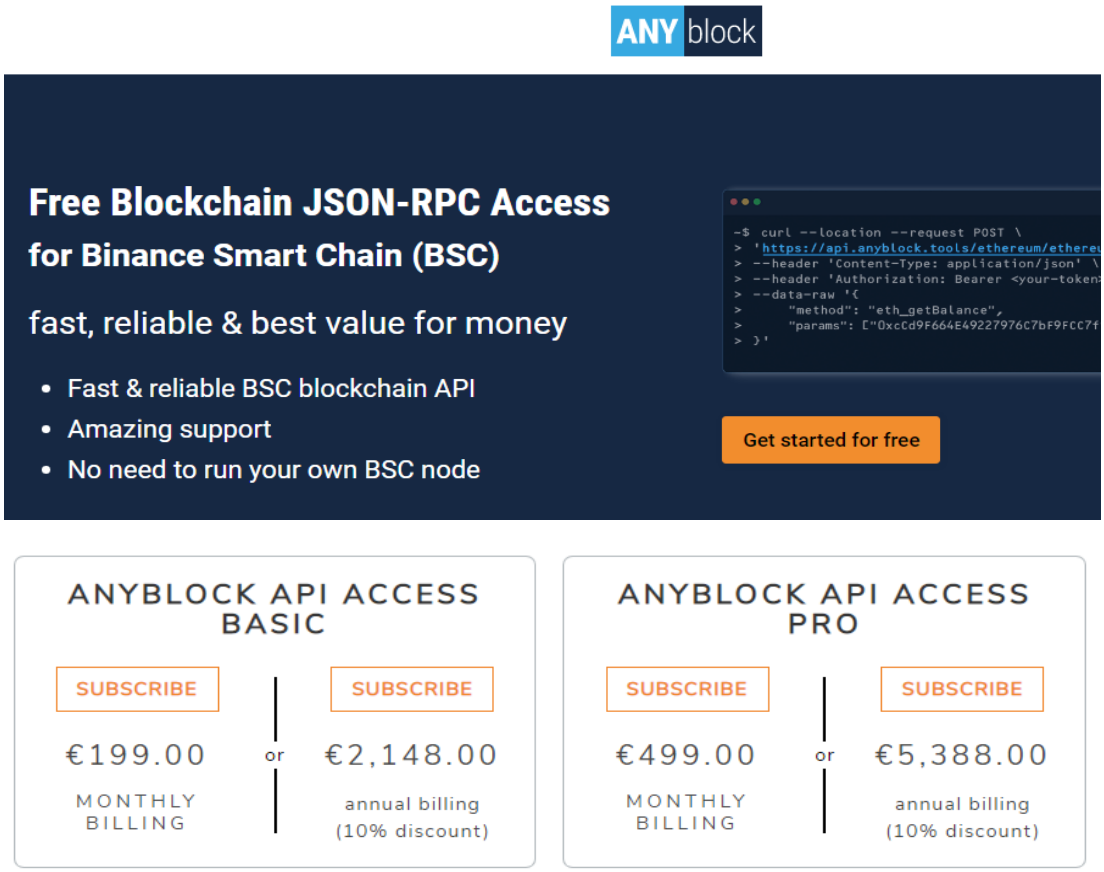
   We can choose suitable service for our project



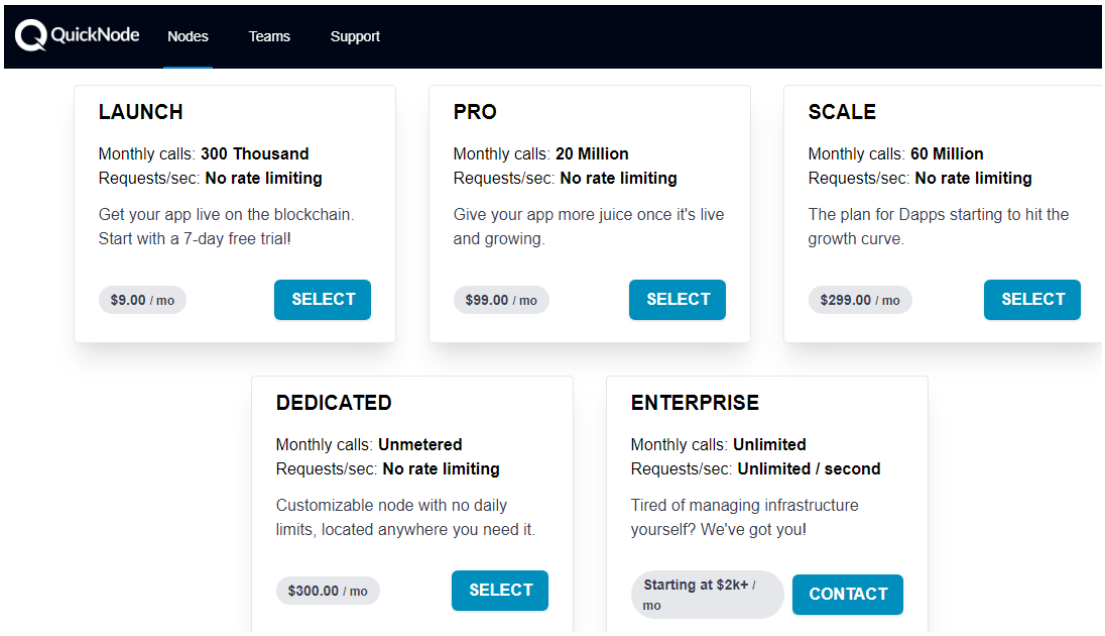
   - Getblock



- Anyblock



- Quicknode



Workflow

1. Get new token info

Sample code snip for listen pending transactions the lastest block(web3.js)

```
// In case you are using Node.js
const Web3 = require('web3');
// Get Personal API
const API_KEY = '95cc2497-d50c-44c8-b574-4bd348811be8';
// Setting getblock node as HTTP provider
// const provider = new Web3.providers.HttpProvider("https://bsc.getblock.io/mainnet/?api_key=" + API_KEY);
// or as WebSocket provider
const provider = new Web3.providers.WebsocketProvider("wss://bsc.getblock.io/testnet/?api_key=" + API_KEY);
// Creating web3 instance with given provider
const web3 = new Web3(provider);
// Initializing web3.eth method
var block = web3.eth.getBlockNumber().then(console.log);
web3.eth.subscribe('pendingTransactions', function (error, result) {
}).on("data", async function (transactionHash) {
    let transaction = await web3.eth.getTransaction(transactionHash);
    if (transaction != null) {
        console.log(transaction);
    }
});
```

Result:

```
E:\work\bsc_subscriber>node getblock.js
10221163
{
  blockHash: null,
  blockNumber: null,
  from: '0xc7a57270daAD312F2aB9086A4f3496f870790656',
  gas: 6700000,
  gasPrice: '10000000000',
  hash: '0x6780a4f2f38b0093482d5ddf61c85a1b39348eff62d6951f4404436117deb11
2',
  input: '0x51160630',
  nonce: 2801,
  to: '0x2B9A60061949dbc1D6FB6ccD38101Fb0EcAcCAC7',
  transactionIndex: null,
  value: '0',
  type: '0x0',
  v: '0xe5',
  r: '0x734cd4bcc329062e37fa9903ccfa0c6726243e51521135bdfa849c8c7e7ca550',
  s: '0x132ce37983d24170f02799ff38a9e0588bdf1a476387562cb7e6ce3fb62e0b83'
}
```

Get transaction receipt from tx hash

```
const web3 = new Web3(provider);
// Initializing web3.eth method
var block = web3.eth.getBlockNumber().then(console.log);
var receipt = web3.eth.getTransactionReceipt('0x6780a4f2f38b0093482d5ddf61c85a1b39348eff62d6951f4404436117deb112')
.then(console.log);
```

If contractAddress is not null, this is new contract and can access from address

```
E:\work\bsc_subscriber>node getblock.js
10223422
{
  blockHash: '0x406cf68279520e977aa1ff1a56286ff941b5e4ba4b267a1c195764a55eb705f
5',
  blockNumber: 10221167,
  contractAddress: null,
  cumulativeGasUsed: 8202635,
  from: '0xc7a57270daad312f2ab9086a4f3496f870790656',
  gasUsed: 31310,
  logs: [],
  logsBloom: '0x00000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000',
  status: false,
  to: '0x2b9a60061949dbc1d6fb6ccd38101fb0ecaccac7',
  transactionHash: '0x6780a4f2f38b0093482d5ddf61c85a1b39348eff62d6951f440443611
7deb112',
  transactionIndex: 7,
  type: '0x0'
}
```

Once got new address, we can get token details from bscscan or other API easily

Get Token Info by ContractAddress PRO

Only return token info for a token contract that is updated on BscScan

```
https://api.bscscan.com/api?module=token&action=tokeninfo&contractaddress=0x0e09fabb73bd3ade0a17ecc321fd13a19e81ce82&apikey=YourApiKey
ken
```

* The above API endpoint is throttled to 2 calls/second regardless of API Pro tier.

Sample return of token info API:


```
{
  "status": "1",
  "message": "OK",
  "result": [
    {
      "contractAddress": "0x...",
      "tokenName": "Token Name",
      "symbol": "Token Symbol",
      "divisor": "18",
      "tokenType": "BEP20",
      "totalSupply": "100000000000000",
      "blueCheckmark": "true",
      "description": "Token Description",
      "website": "https://token.website",
      "email": "email@token.website",
      "blog": "https://blog.token.website/",
      "reddit": "https://www.reddit.com/r/tokenwebsite/",
      "slack": "https://chat.token.website/",
```

Get Token Holder List by ContractAddress PRO

Return the current token holder and number of tokens held

```
https://api.bscscan.com/api?module=token&action=tokenholderlist&contractaddress=0x0e09fabb73bd3ade0a17ecc321fd13a19e81ce82&page=1&offse
=10000&apikey=YourApiKeyToken
```

Sample return of token info API:


```
{
  "status": "1",
  "message": "OK",
  "result": [
    {
      "TokenHolderAddress": "0x0000000000000000000000000000000000000000",
      "TokenHolderQuantity": "0"
    },
    {
      "TokenHolderAddress": "0x0000000000000000000000000000000000000001",
      "TokenHolderQuantity": "844975208812149014804348"
    },
    {
      "TokenHolderAddress": "0x0000000000000000000000000000000000000002",
      "TokenHolderQuantity": "120243030179598696424583296"
    }
  ]
}
```


CoinMarketCap

Technical Notes

- A historic quote for every "interval" period between your "time_start" and "time_end" will be returned.
- If a "time_start" is not supplied, the "interval" will be applied in reverse from "time_end".
- If "time_end" is not supplied, it defaults to the current time.
- At each "interval" period, the historic quote that is closest in time to the requested time will be returned.
- If no historic quotes are available in a given "interval" period up until the next interval period, it will be skipped.

Interval Options

There are 2 types of time interval formats that may be used for "interval".

The first are calendar year and time constants in UTC time:
"hourly" - Get the first quote available at the beginning of each calendar hour.
"daily" - Get the first quote available at the beginning of each calendar day.
"weekly" - Get the first quote available at the beginning of each calendar week.
"monthly" - Get the first quote available at the beginning of each calendar month.
"yearly" - Get the first quote available at the beginning of each calendar year.

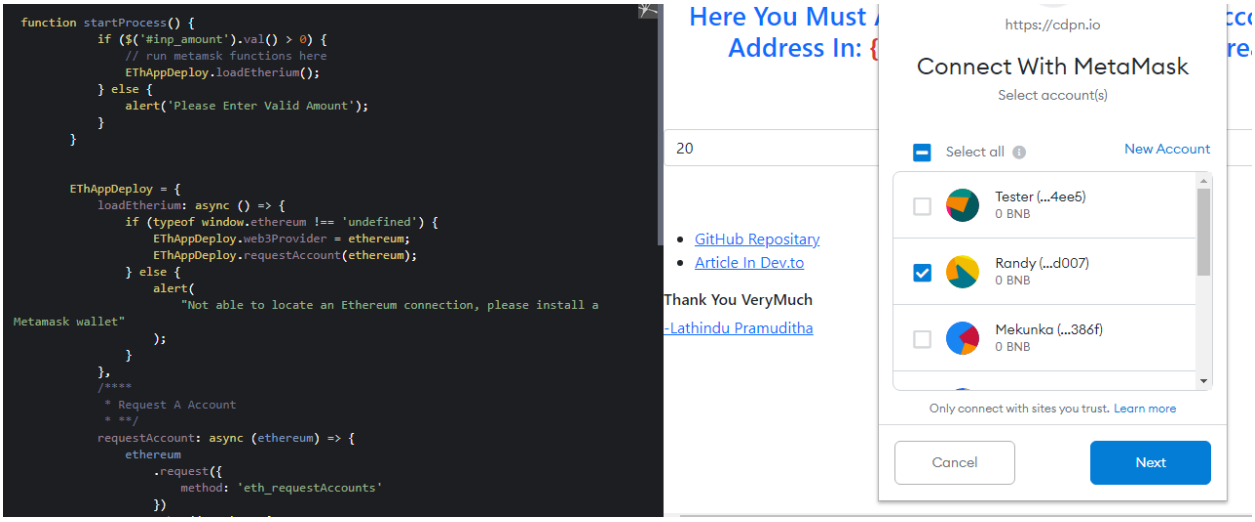
The second are relative time intervals.
"m": Get the first quote available every "m" minutes (60 second intervals). Supported minutes are: "5m", "10m", "15m", "30m", "45m".

200 Successful · 400 Bad Request
401 Unauthorized · 403 Forbidden
429 Too Many Requests · 500 Internal Server Error


```
{
    "data": {
        "quotes": [
            {
                "timestamp": "2018-07-31T00:02:(
                "btc_dominance": 47.9949,
                "active_cryptocurrencies": 2500,
                "active_exchanges": 600,
                "active_market_pairs": 1000,
                "quote": {
                    "USD": {
                        "total_market_cap": 292
                        "total_volume_24h": 176
                        "total_volume_24h_repor
                        "altcoin_market_cap": 1
                        "altcoin_volume_24h": 3
                        "altcoin_volume_24h_rep
                        "timestamp": "2018-07-3
```


And I think it’s necessary to get some items from seveal APIs

This is the sample code to integrate Metamask to website



```
function startProcess() {
          if ($('#inp_amount').val() > 0) {
              // run metamsk functions here
              EThAppDeploy.loadEtherium();
          } else {
              alert('Please Enter Valid Amount');
          }
      }


      EThAppDeploy = {
          loadEtherium: async () => {
              if (typeof window.ethereum !== 'undefined') {
                  EThAppDeploy.web3Provider = ethereum;
                  EThAppDeploy.requestAccount(ethereum);
              } else {
                  alert(
                      "Not able to locate an Ethereum connection, please install a
Metamask wallet"
                  );
              }
          },
          /****
           * Request A Account
           * **/
          requestAccount: async (ethereum) => {
              ethereum
                  .request({
                      method: 'eth_requestAccounts'
                  })
```


I will research more suitable API and prepare all to start.

Please share me updated theme and charting view libraries.