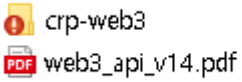
[require crp-web3]




```
var Web3 = require('../../web3');
var web3 = new Web3();

web3.setProvider(new web3.providers.HttpProvider('http://192.168.0.133:8545'));
```

위의 crp-web3 폴더자체를 require 해주세요.

- 위해서는 require(../../web3);이지만 실제로 저 폴더를 require한다면 require(../../crp-web3); 로 해줘야 합니다.
- 즉, 폴더명이 변경되면 해당 폴더명으로 require!

[web3.eth.getPermissionTx(<EOA>)]

```
return web3.eth.getPermissionTx(eoa)
```

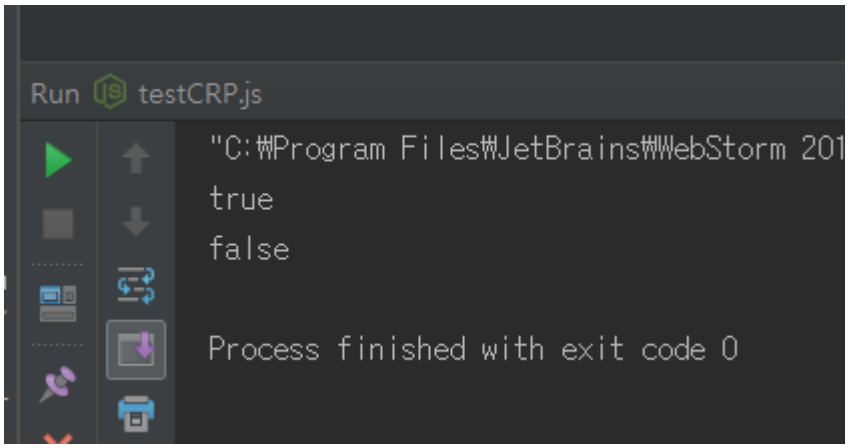
특정 EOA가 Token을 발행할 수 있는지 확인하는 API.

[PARAMETER]

String <eoa> 입력 필수 : erc20 token 발행 권한을 확인할 eoa

[OUTPUT]

<반환> : bool



[web3.eth.getMainContractAddress(<EOA>)]

```
web3.eth.getMainContractAddress(eoa)
```

특정 EOA의 Main Contract Address 정보를 가져옵니다.

[PARAMETER]

String <eoa> 입력 필수 : MCA를 확인할 EOA

[OUTPUT]

<반환> : string -> "mca값"

```
0xd7357f6a2789f451861c93aa5f254f88af0f20b3
```

<null일 경우> : string -> "0x"

```
0x
```

[web3.eth.sendPermissionTx(<from>, <EOA>)]

```
web3.eth.sendPermissionTx(from, account)
```

```
web3.eth.getPermissionTx(from, account, gas)
```

특정 Account에게 Contract를 Deploy를 할 수 있는 권한을 부여합니다.

- 해당 API는 운영자 전용이며, from이 운영자가 아니면 적용되지 않습니다.

[PARAMETER]

String <From> 입력 필수 : 운영자 EOA

String <EOA> 입력 필수 : 권한을 부여할 유저 EOA

Number <gas> : TX 수수료 지정.

- **From, account, gas에서 gas를 입력하지 않으면 기본 gas 값이 적용됩니다.**

[OUTPUT]

<반환> : TX Hash를 반환 (string)

```
0xc89ea79e728b3bb04633e6b8a944ff4df3cb8614d8e956d6bf0ecbad438ef5e0
```

[web3.personal.clearMainContractAddress(<EOA>, <PWD>)]

```
web3.personal.clearMainContractAddress(eoa, pwd)
```

```
web3.personal.clearMainContractAddress(eoa, pwd, gas)
```

```
web3.personal.clearMainContractAddress(eoa, pwd, gas, duration)
```

프로젝트가 끝나고 새로운 프로젝트를 진행하기 위해서 최초로 올린 Contract를 제거하는 API

[PARAMETER]

String <EOA> 입력 필수 : MCA를 제거할 Account.

String <PWD> 입력 필수 : account의 비밀번호

Number <gas> : TX 수수료 지정.

Number <duration> : account unlock 지속시간.

[OUTPUT]

<반환> : Tx Hash를 반환 (string)

```
0xe6301abab128f92107d392b234df2c18bddeefb4ec552ff5b50981a5092d0cf1
```

[web3.personal.loginAccount(<EOA>, <PWD>)]

```
web3.personal.loginAccount(web3.eth.accounts[0], pwd);
```

GUI Wallet에서 Login을 하기 위한 API

[PARAMETER]

string <EOA> 입력 필수 : Login을 시도할 EOA

string <PWD> 입력 필수 : 해당 EOA의 PWD

[OUTPUT]

<반환> : bool

```
> personal.loginAccount(eth.accounts[2], "1")
true
> personal.loginAccount(eth.accounts[2], "2")
false
```

[web3.eth.isCA(<CA>)]

```
web3.eth.isCA(ca0, function (err, isCa) {
    console.log("isCA : " + isCa);
    console.log("err : " + err)
});
```

CA 유효성 검사 API

[PARAMETER]

string <CA> 입력 필수 : 유효성을 검사할 CA

[OUTPUT]

isCa = 유효성 체크 결과 (true이면 CA)

error = 에러 (Error 발생시 isCa는 false)

```
isCa : true
err : null
```

[web3.eth.getAdminAddress()]

```
var Web3 = require('../../web3');
var web3 = new Web3();

web3.setProvider(new web3.providers.HttpProvider('http://192.168.0.133:8545'));

console.log(web3.eth.getAdminAddress());
```

현재 Super User인 EOA를 반환합니다.

[OUTPUT]

<반환> : tx hash를 반환합니다. (string)

```
0x05a714a3eeb9fd4c91bfd8472487448ba838cf1c
```

[web3.eth.isAccount(<EOA>)]

```
console.log(web3.eth.isAccount(account))
```

해당 EOA가 Geth에 존재하는 EOA인지 체크.

[PARAMETER]

string <EOA> 입력 필수 : 존재 유무를 체크할 EOA

[OUTPUT]

True : Geth(Node)에 존재하는 EOA

False : Geth(Node)에 존재하지 않는 EOA

[web3.eth.ResendTx(<Tx Hash>)]

```
web3.eth.resendTx(txHash)
```

기존의 Resend를 사용하기 편하도록 업데이트!

재전송하고 싶은 Tx Hash값만 넣으면 자동으로 gasPrice를 10% 펌핑 후 재전송합니다.

[PARAMETER]

string <TxHash> 입력 필수 : 재전송 할 Tx Hash

[OUTPUT]

<반환> : (string)Tx Hash

```
0xff05bb7250b02059cc87eed2be89aeba29e66735861f67f2e05805460322b8cc
```

[web3.eth.getCaBalanceOf(<ca>, <eoa>)]

```
web3.eth.getCaBalanceOf(ca, eoa)
```

```
web3.eth.getCaBalanceOf(ca, eoa, port)
```

```
web3.eth.getCaBalanceOf(ca, eoa, port, protocol)
```

Contract에서 해당 EOA가 소지한 토큰의 개수를 반환.

[PARAMETER]

string <ca> 입력 필수 : balanceOf할 CA

string <eoa> 입력 필수 : 소지한 토큰을 확인할 EOA

int <port> : default 8545

Bool <protocol> : default는 false(http)

- True의 경우 https가 적용됩니다.

[OUTPUT]

<반환> : (string) 소지한 Token 개수

```
> test getBal.js
500000000000000000
```

[web3.eth.getTokenInfo(<eoa>)]

```
web3.eth.getTokenInfo(account)
```

Eoa가 소유하고 있는 Token의 CA들을 반환합니다.

- 해당 EOA가 어떠한 토큰을 1개라도 가지고 있으면 해당 ca 반환.

[PARAMETER]

string <eoa> 입력 필수 : CA리스트를 확인할 EOA

[OUTPUT]

<반환> : ([]string) Ca

```
[ '0xD7E451f6Be8F3E06AAA4cEb7a155378BE413BfcD',
  '0x7Dc017005104269127D852EEaf3b4F491D070510',
  '0xE20DE2079dB35B5d317F4dCDB68A5963ef9751a5',
  '0xE98aD60Cb4fd2d507BDE47b66Df06593eFce3a18',
  '0xd7357F6a2789F451861c93aa5F254F88af0f20B3',
  '0x25b2FB642afe0bA385AcfaeEB2115605b7128311' ]
```

[web3.eth.checkTxPool(<eoa>)]

```
return web3.eth.checkTxPool(eoa)
```

TxPool에 해당 eoa의 txs가 있는지 체크.

[PARAMETER]

string <eoa> 입력 필수 : tx를 필터링할 key값.

[OUTPUT]

<반환> : (bool) pending and queue 체크 결과

```
> eth.checkTxPool(eth.coinbase)
true
> eth.checkTxPool(eth.accounts[2])
false
```

[contract.crpNew()]

[원본]

50블록 이내에 Receipt가 반환하는지 모니터링 On

```
let instance = await contract.token.new(_name, _symbol, _type, main_addr, {
    from: _owner,
    data: contract.token_data,
    gas: gas,
    mca: main_addr
}); // 4. Tx 발행 (contract.token.new)
```

[추가 방식]

50블록 이내에 Receipt가 반환하는지 모니터링 Off

```
let instance = await contract.token.crpNew(_name, _symbol, _type, main_addr, {
    from: _owner,
    data: contract.token_data,
    gas: gas,
    mca: main_addr
}); // 4. Tx 발행 (contract.token.new)
```

[web3.eth.getCaTokenDB(<tx-nonce>)]

```
return web3.eth.getCaTokenDB(0)
```

TokenDB Contract Address를 가져옵니다.

[PARAMETER]

int <tx-nonce> 입력 필수 : TokenDB가 Deploy된 Tx nonce

[OUTPUT]

<반환> : (string) Contract Address

```
> eth.getCaTokenDB(0)
"0xb6f476267d26dbe06b553dcab620005cb746bf84"
```

[web3.eth.getCaStepDB(<tx-nonce>)]

```
return web3.eth.getCaTokenDB(0)
```

StepDB Contract Address를 가져옵니다.

[PARAMETER]

int <tx-nonce> 입력 필수 : StepDB가 Deploy된 Tx nonce

[OUTPUT]

<반환> : (string) Contract Address

```
> eth.getCaStepDB(0)
"0xb6f476267d26dbe06b553dcab620005cb746bf84"
```