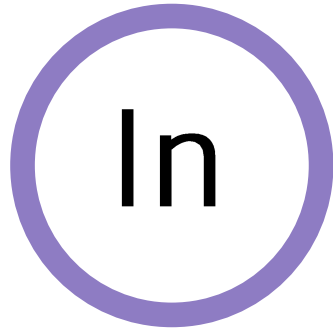


# 이더리움 Client Web3.js

Web3.js

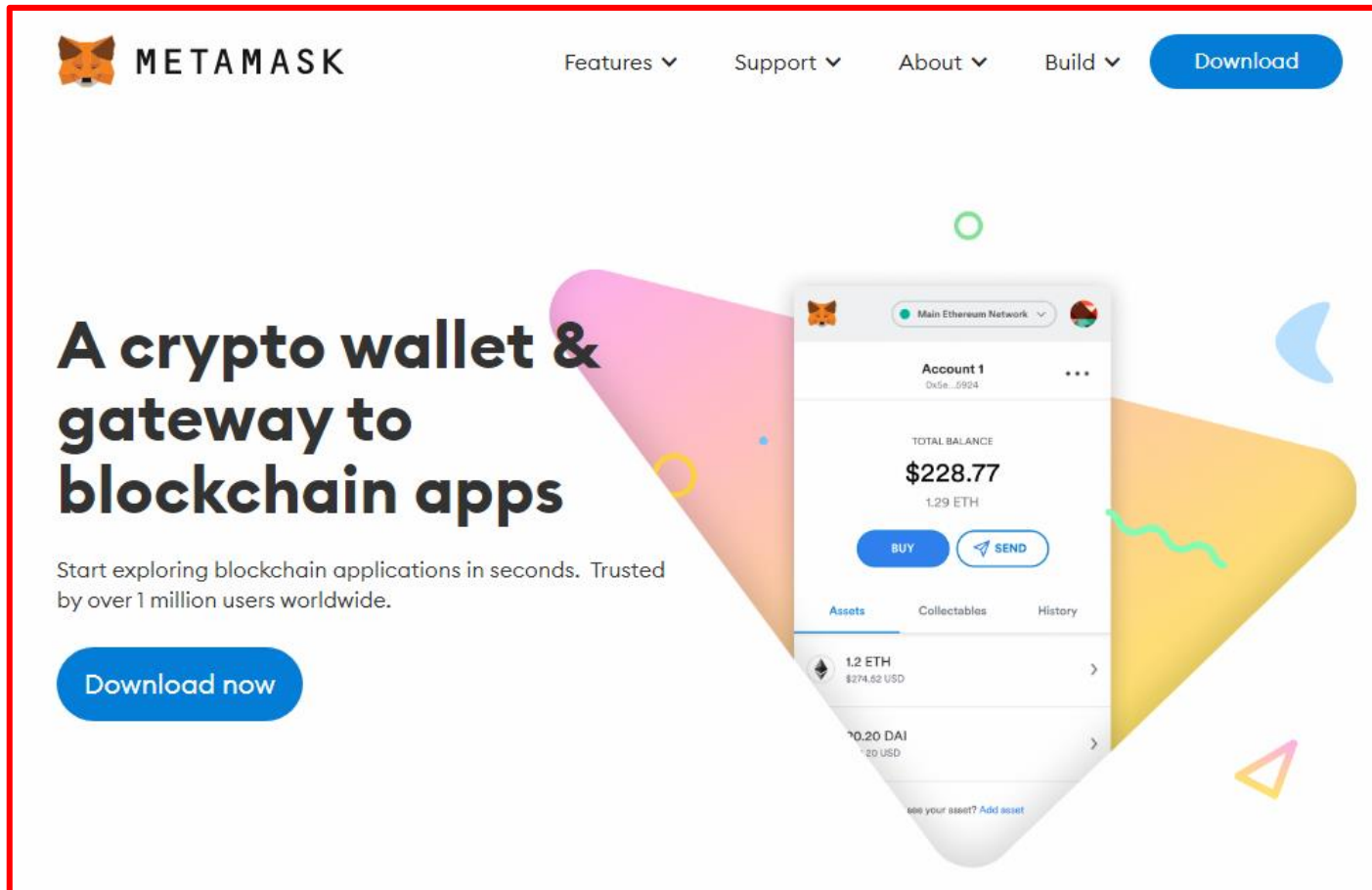
---



Web3.js와 Metamask

# 01 Metamask API 사용하기

## MetaMask

The image shows the Metamask website banner. At the top left is the Metamask logo (an orange fox head) followed by the text "METAMASK". To the right are navigation links: "Features", "Support", "About", and "Build", each with a dropdown arrow. Further right is a blue "Download" button. The main content area features the headline "A crypto wallet & gateway to blockchain apps" in large, bold, black font. Below this is a sub-headline: "Start exploring blockchain applications in seconds. Trusted by over 1 million users worldwide." At the bottom left of this section is a blue "Download now" button. On the right side of the banner is a screenshot of the Metamask mobile app interface. The app shows the "Main Ethereum Network" selected at the top. Below that is "Account 1" with a balance of "\$228.77" (1.29 ETH). There are "BUY" and "SEND" buttons. At the bottom, there's a list of assets: "1.2 ETH" (\$274.62 USD) and "10.20 DAI" (10.20 USD). The background of the banner has colorful abstract shapes in pink, yellow, and blue.

**METAMASK**

Features ▾ Support ▾ About ▾ Build ▾ [Download](#)

### A crypto wallet & gateway to blockchain apps

Start exploring blockchain applications in seconds. Trusted by over 1 million users worldwide.

[Download now](#)

Account 1  
0x5e...5924

TOTAL BALANCE  
**\$228.77**  
1.29 ETH

[BUY](#) [SEND](#)

Assets Collectables History

1.2 ETH  
\$274.62 USD

10.20 DAI  
10.20 USD

See your asset? [Add asset](#)

[Metamask Ethereum API]

<https://docs.metamask.io/guide/ethereum-provider.html#ethereum-provider-api>

# 01 Metamask API 사용하기

## MetaMask

이더리움 기반 웹사이트에서 계정 관리를 담당한다!

MetaMask injects a global API into websites visited by its users at `window.ethereum`. This API allows websites to request users' Ethereum accounts, read data from blockchains the user is connected to, and suggest that the user sign messages and transactions. The presence of the provider object indicates an Ethereum user. We recommend using `@metamask/detect-provider` to detect our provider, on any platform or browser.

[Metamask Ethereum API]

<https://docs.metamask.io/guide/ethereum-provider.html#ethereum-provider-api>

## Why MetaMask

MetaMask was created to meet the needs of secure and usable Ethereum-based web sites. In particular, it handles account management and connecting the user to the blockchain.

**Metamask는 Extension설치시 자동으로 script가 주입됨!**

→ `window.ethereum`

# 01 Metamask

## Metamask API 시작하기



[로그인 클릭시 metmask실행]

1. window.ethereum(메타마스크) 존재하는지 확인
- 2-1. window.ethereum이 없으면 Metamask설치페이지로 이동
- 2-2. window.ethereum이 존재하면 연결요청

```
1  async function connectMetamask(){
2    if (typeof window.ethereum === 'undefined') {
3      // metamask 설치 안되어 있을시
4      console.log('MetaMask is not installed!');
5      const answer = confirm('Metmask가 필요합니다. 설치하시겠습니까?');
6      if (answer) {
7        window.open(
8          'https://metamask.io/download',
9          '_blank' // 새 탭에서 열기
10       );
11     }
12   } else {
13     await window.ethereum.request({ method: 'eth_requestAccounts' });
14   }
15 }
```

# 01 Metamask

## Metamask API 시작하기

```
1  async function connectMetamask(){
2    if (typeof window.ethereum === 'undefined') {
3      // metamask 설치 안되어 있을시
4      console.log('MetaMask is not installed!');
5      const answer = confirm('Metamask가 필요합니다. 설치하시겠습니까?');
6      if (answer) {
7        window.open(
8          'https://metamask.io/download',
9          '_blank' // 새 탭에서 열기
10       );
11     }
12   } else {
13     await window.ethereum.request({ method: 'eth_requestAccounts' });
14   }
15 }
```

**Metamask API**

public/js/scripts.js

```
49
50   <button
51     type="button"
52     class="btn btn-warning"
53     style="color: white; font-weight: 500"
54     id="login-button"
55     onclick="connectMetamask()"
56   > 로그인
57 </button>
```

header.html(ejs || nunjucks ...)

**\*\* 당연히 각 문서(html)에서 script load가 필요합니다.**

# 01 Metamask

## Metamask API 함수

### ethereum.isConnected()

```
ethereum.isConnected(): boolean;
```

ts

→ 메타마스크가 ethereum 체인과 연결되어 있는지

# 01 Metamask

## Metamask API 함수

### ethereum.request(args)

```
interface RequestArguments {  
  method: string;  
  params?: unknown[] | object;  
}  
  
ethereum.request(args: RequestArguments): Promise<unknown>;
```

→ Ethereum RPC Server와 통신하기.

→ method 호출

[RPC API Documents]

<https://docs.metamask.io/guide/rpc-api.html#table-of-contents>

[https://eth.wiki/json-rpc/API#eth\\_accounts](https://eth.wiki/json-rpc/API#eth_accounts)

- 최근 Metamask 업데이트로 인해, Metamask안에서 Contract를 호출할 수 있게 되었음.

```
params: [  
  {  
    from: '0xb60e8dd61c5d32be8058bb8eb970870f07233155',  
    to: '0xd46e8dd67c5d32be8058bb8eb970870f07244567',  
    gas: '0x76c0', // 30400  
    gasPrice: '0x9184e72a000', // 10000000000000  
    value: '0x9184e72a', // 2441406250  
    data:  
      '0xd46e8dd67c5d32be8d46e8dd67c5d32be8058bb8eb970870f072445675058bb8eb970870f072445675',  
  },  
];  
  
ethereum  
  .request({  
    method: 'eth_sendTransaction',  
    params,  
  })  
  .then((result) => {  
    // The result varies by by RPC method.  
    // For example, this method will return a transaction hash hexadecimal string on success.  
  })  
  .catch((error) => {  
    // If the request fails, the Promise will reject with an error.  
  });
```



# 01 Metamask

## Metamask API Event

### connect

```
interface ConnectInfo {  
  chainId: string;  
}  
  
ethereum.on('connect', handler: (connectInfo: ConnectInfo) => void);
```

ts

### disconnect

```
ethereum.on('disconnect', handler: (error: ProviderRpcError) => void);
```

ts

# 01 Metamask

## Metamask API Event

chainChanged – chain 연결 변경시

```
ethereum.on('chainChanged', handler: (chainId: string) => void);
```

ts

accountsChanged – account 변경시

```
ethereum.on('accountsChanged', handler: (accounts: Array<string>) => void);
```

ts

## 02 Web3.js

### **web3.js - Ethereum JavaScript API**

web3.js is a collection of libraries that allow you to interact with a local or remote ethereum node using HTTP, IPC or WebSocket.

The following documentation will guide you through [installing and running web3.js](#) as well as providing an API reference documentation with examples.

Contents:

<https://web3js.readthedocs.io/en/v1.4.0/index.html>

## 02 Web3.js

### [web3 구성요소]

- web3-eth      스마트컨트랙트를 위한 모듈
- web3-shh      whisper 프로토콜(P2P 커뮤니케이션)
- web3-bzz      swarm 프로토콜(데이터 분산저장)
- web3-utils      유틸리티 모듈

- [http://wiki.hash.kr/index.php/%EC%9C%84%EC%8A%A4%ED%8D%BC\\_%ED%94%84%EB%A1%9C%ED%86%A0%EC%BD%9C](http://wiki.hash.kr/index.php/%EC%9C%84%EC%8A%A4%ED%8D%BC_%ED%94%84%EB%A1%9C%ED%86%A0%EC%BD%9C)
- <http://wiki.hash.kr/index.php/%EC%8A%A4%EC%9B%9C%ED%94%84%EB%A1%9C%ED%86%A0%EC%BD%9C>

- 최근 Metamask 업데이트로 인해, Metamask Contract를 호출할 수 있음.
- web3 vs metamask?

우리는 Web3를 사용할 예정!

## 03 Metamask로 ethereum 사용하기

시작하기

**ethereum.request(args)**

```
interface RequestArguments {  
  method: string;  
  params?: unknown[] | object;  
}  
  
ethereum.request(args: RequestArguments): Promise<unknown>;
```

**ethereum.request 메소드를 이용한다**

이더리움 account 요청하기. //로그인기능

```
ethereum.request({ method: 'eth_requestAccounts' });
```

```
const accounts = await window.ethereum.request({ method: 'eth_requestAccounts' });
```

## 03 Metamask로 ethereum 사용하기

[주요한 method]

web3\_clientVersion (현재 사용하는 ethereum Client)

```
await ethereum.request({method: "web3_clientVersion"})  
"MetaMask/v9.8.4"
```

---

net\_listening (현재 client의 network연결이 가능한지)

```
await ethereum.request({method: "net_listening"})  
true
```

## 03 Web3로 ethereum 사용하기

[모듈 확인하기]

> Web3.modules

[Web3 사용하기]

```
<script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
```

```
const web3 = new Web3(Web3.givenProvider || "ws://localhost:7545");
```

---

# 03 Web3로 ethereum 사용하기

[모듈 확인하기]

> Web3.modules

```
▼ {Eth: f, Net: f, Personal: f, Shh: f, Bzz: f} ⓘ  
  ▶ Bzz: f t(e)  
  ▶ Eth: f ()  
  ▶ Net: f ()  
  ▶ Personal: f ()  
  ▶ Shh: f ()  
  ▶ [[Prototype]]: Object
```

---

[Web3 사용하기]

```
const web3 = new Web3(Web3.givenProvider || "ws://localhost:7545");
```

---



## 03 Web3로 ethereum 사용하기

[web3 버전 확인하기]

> Web3.version

[Web3 이용가능한 instance]

web3.eth

▶ *k {\_requestManager: t, givenProvider: Proxy, providers: {...}, \_provider: Proxy, ...}*

---

web3.shh

▶ *s {\_requestManager: t, givenProvider: Proxy, providers: {...}, \_provider: Proxy, ...}*

---

web3.bzz

▶ *t {givenProvider: null, pick: {...}, currentProvider: null, isAvailable: f, upload: f, ...}*

---

web3.utils

▶ *{\_fireError: f, \_jsonInterfaceMethodToString: f, \_flattenTypes: f, randomHex: f, BN: f, ...}*

# 03 Web3로 ethereum 사용하기

Method 종류

**[requestAccounts]** – 로그인 시도

```
web3.eth.requestAccounts([callback])
```

**[getNodeInfo]** – 클라이언트 확인

```
web3.eth.getNodeInfo([callback])
```

# 03 Web3.eth 모듈

## web3.eth 모듈

### [account 확인하기]

> web3.eth.getAccounts([callback])

### [defaultAccount] – 기본 계정 설정 // TX를 위한 from을 자동으로 삽입시켜줌

> web3.eth.defaultAccount

> web3.eth.defaultAccount = (await web3.eth.getAccounts())[0]

### [getGasPrice] – 현재 가스 가격을 return (마지막 블록들의 가스 중간값으로 판단.) // 단위: wei

> web3.eth.getGasPrice([callback])

### [getBlockNumber] – 블록 넘버 가져옴

> web3.eth.getBlockNumber([callback])

## 03 Web3.eth 모듈

### web3.eth 모듈 (조회하기)

#### [getBalance] – 잔고 가져옴

```
web3.eth.getBalance(address [, defaultBlock] [, callback])
```

```
await web3.eth.getBalance(web3.eth.defaultAccount)  
"99891245660000000000"
```

```
await web3.eth.getBalance(web3.eth.defaultAccount, 0)  
"1000000000000000000000000"
```

#### [getCode] – CA의 잔고 가져옴

```
web3.eth.getCode(address [, defaultBlock] [, callback])
```

```
await web3.eth.getCode("0xa790CD036AE26b79D6d81DC21653980108800e50")
```

## 03 Web3.eth 모듈

### web3.eth 모듈 (조회하기)

**[getBlock]** – 블록 가져옴. // mainnet에서 테스트 해보자.

```
web3.eth.getBlock(blockHashOrBlockNumber [, returnTransactionObjects] [, callback])
```

```
> await web3.eth.getBlock(6*24*50001, true)
```

**[getBlockTransactionCount]** – 블록의 TX Count. // mainnet에서 테스트 해보자.

```
web3.eth.getBlockTransactionCount(blockHashOrBlockNumber [, callback])
```

**[getUncle]** – 블록의 잉클블록. // mainnet에서 테스트 해보자.

```
web3.eth.getBlockTransactionCount(blockHashOrBlockNumber [, callback])
```

```
await web3.eth.getUncle(6*24*50001, 0)
```

## 03 Web3.eth 모듈

### web3.eth 모듈 (조회하기)

#### [getTransaction] – getTransaction

```
web3.eth.getTransaction(transactionHash [, callback])
```

```
await web3.eth.getTransaction("0x1878dde1fca0db4e0e23344ead2f2822010c081d776187e46f354653e2f420c9")
```

#### [getTransactionFromBlock] – Block에서 idnex로 조회하기

```
getTransactionFromBlock(hashStringOrNumber, indexNumber [, callback])
```

#### [getTransactionReceipt] – 영수증 확인

```
web3.eth.getTransactionReceipt(hash [, callback])
```

## 03 Web3.eth 모듈

### web3.eth 모듈 (TX보내기)

#### [sendTransaction] – Tx 보내기

```
web3.eth.sendTransaction(transactionObject [, callback])
```

#### contract deploy (to가 비어 있음)

```
web3.eth.sendTransaction({  
  from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',  
  data: code // deploying a contract  
})
```

#### TX 보내기(송금)


```
web3.eth.sendTransaction({  
  from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',  
  to: '0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe',  
  value: '10000000000000000'  
})
```

## 03 Web3.eth 모듈

### web3.eth 모듈 (TX보내기)

```
// using the event emitter
web3.eth.sendTransaction({
  from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',
  to: '0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe',
  value: '10000000000000000'
})
.on('transactionHash', function(hash){
  ...
})
.on('receipt', function(receipt){
  ...
})
.on('confirmation', function(confirmationNumber, receipt){ ... })
.on('error', console.error); // If a out of gas error, the second parameter is the receipt.
```





## 03 Web3.eth 모듈

### web3.eth 모듈 (TX보내기)

#### 실습

버튼을 클릭하면 0번 계좌에 송금을 보내는 코드를 작성하세요.

**[call] – call TX 보내기.**

```
web3.eth.call(callObject [, defaultBlock] [, callback])
```

- external view 함수 호출할 때 사용.
- Call TX은 가스를 소비하지 않음.

[illegible]

## 03 Web3.eth 모듈

### web3.eth 모듈 (TX보내기)

**[estimateGas]** – gas소비량 예상.

```
web3.eth.estimateGas(callObject [, callback])
```

**[estimateGas]** – gas소비량 예상.

```
web3.eth.estimateGas(callObject [, callback])
```

## 03 Web3.eth 모듈

**web3.eth.Contract 모듈**

**[contract 연결]**

```
new web3.eth.Contract(jsonInterface[, address][, options])
```

```
const contract= new web3.eth.Contract(d.abi)
```

---

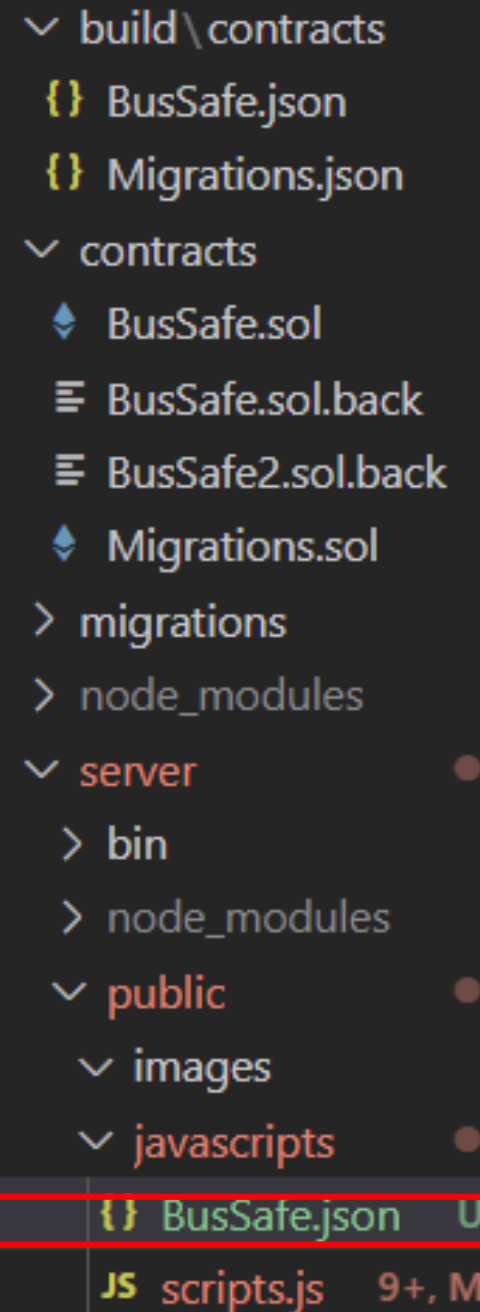
## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract 연결]

1. BusSafe.json파일을 옮긴다.  
*localhost:3000/javascripts/BusSafe.json 접속 확인*
2. js 내부에서 해당 파일을 불러온다.  
*fetch 함수.*

```
29 const resp = await fetch("/javascripts/scripts.js");
30 const data = resp.json();
31 const busSafeContract = new web3.eth.Contract(data.abi);
```



## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract options]

```
myContract.options
```

```
> {  
  address: '0x1234567890123456789012345678901234567891',  
  jsonInterface: [...],  
  from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',  
  gasPrice: '1000000000000000',  
  gas: 1000000  
}
```

내용 확인.

## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract methods]

**[clone]** // 복사하고 특정 옵션만 변경할 때(from 등)

```
var contract1 = new eth.Contract(abi, address, {gasPrice: '12345678', from: fromAddress});

var contract2 = contract1.clone();
contract2.options.address = address2;
```

## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract 사용하기]

```
myContract.methods.myMethod([param1[, param2[, ...]]])
```

- 우리의 Contract method 접근방법.
- Transaction을 알아서 만들어줌.
- 결과치에 이용할 수 있는 function(call, send, estimateGas, encodeABI(실제 전달할 데이터))



## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract 사용하기]

#### [call – 조회]

```
myContract.methods.myMethod([param1[, param2[, ...]]]).call(options [, defaultBlock] [, callback])
```

options: {from, gasPrice, gas}...

```
myContract.methods.myMethod(123).call({from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe'},
```

## 03 Web3.eth 모듈

**web3.eth.Contract 모듈**

**[contract 사용하기]**

**[send – tx보내기]**

```
myContract.methods.myMethod([param1[, param2[, ...]]]).send(options[, callback])
```

options:

- from
- gasPrice
- gas
- value

## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract 사용하기]

#### [send – tx보내기]

```
myContract.methods.myMethod(123).send({from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe'})  
  .on('transactionHash', function(hash){  
    ...  
  })  
  .on('receipt', function(receipt){  
    ...  
  })  
  .on('confirmation', function(confirmationNumber, receipt){  
    ...  
  })  
  .on('error', function(error, receipt) {  
    ...  
  });
```

## 03 Web3.eth 모듈

**web3.eth.Contract 모듈**

**[contract 사용하기]**

**[estimateGas- gas 계산하기]**

```
myContract.methods.myMethod([param1[, param2[, ...]]]).estimateGas(options[, callback])
```

options:

- from
- gas
- value

예상 가스량이 나옴.

## 03 Web3.eth 모듈

**web3.eth.Contract 모듈**

**[contract 사용하기]**

**[encodeABI – 실제 전달할 data!]**

```
myContract.methods.myMethod([param1[, param2[, ...]])].encodeABI()
```

## 03 Web3.eth 모듈

### web3.eth.Contract 모듈

#### [contract 사용하기]

#### [Event 처리하기]

##### Event

- contract.once("<Event 이름> || "allEvents", function(error,event): "이벤트리스너") ➔ 한번 실행
- contract.events.<Event이름 > || "allEvents"

```
myContract.events.MyEvent([options][, callback])
```

## 03 Web3.eth 모듈

**[Event 처리하기]** - contract.events.<Event이름> || "allEvents"

```
myContract.events.MyEvent({
  filter: {myIndexedParam: [20,23], myOtherIndexedParam: '0x123456789...'}, // Using an array means
  fromBlock: 0
}, function(error, event){ console.log(event); })
.on("connected", function(subscriptionId){
  console.log(subscriptionId);
})
.on('data', function(event){
  console.log(event); // same results as the optional callback above
})
```

### event Emitter

- data : 이벤트 발생시
- connected: 연결(이벤트 주기적으로 받음)
- error: 이벤트 연결시 에러

### returnValue

- event
- signature
- address
- returnValues
- txindex, txhash
- blockidx, blockHash