Read the Docs Template Documentation

Release 1.0

Read the Docs

Feb 17, 2021

Contents

Contents:

CHAPTER 1

Authors

- Shah Rizal (New contributor)
- Fazuan
- Hisham

CHAPTER 2

Template

$project will solve your problem of where to start with documentation, by providing a basic explanation of how to do it easily.

Look how easy it is to use:

```
import project # Get your stuff done project.do_stuff()
```

2.1 Features

- Be awesome
- Make things faster

2.2 Installation

Install $project by running:

```
install project
```

2.3 Contribute

- Issue Tracker: github.com/$project/$project/issues
- Source Code: github.com/$project/$project

2.4 Support

If you are having issues, please let us know. We have a mailing list located at: project@google-groups.com

2.5 License

The project is licensed under the BSD license.

CHAPTER 3

Usage

To use this template, simply update it:

```
import read-the-docs-template
```

CHAPTER 4

web3.eth

The `web3-eth` package allows you to interact with an Ethereum blockchain and Ethereum smart contracts.

```
var Eth = require('web3-eth');

// "Eth.providers.givenProvider" will be set if in an Ethereum supported browser.
var eth = new Eth(Eth.givenProvider || 'ws://some.local-or-remote.node:8546');


// or using the web3 umbrella package

var Web3 = require('web3');
var web3 = new Web3(Web3.givenProvider || 'ws://some.local-or-remote.node:8546');

// -> web3.eth
```

4.1 Note on checksum addresses

All Ethereum addresses returned by functions of this package are returned as checksum addresses. This means some letters are uppercase and some are lowercase. Based on that it will calculate a checksum for the address and prove its correctness. Incorrect checksum addresses will throw an error when passed into functions. If you want to circumvent the checksum check you can make an address all lower- or uppercase.

4.1.1 Example

```
web3.eth.getAccounts(console.log);
> ["0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe" ,
↪"0x85F43D8a49eeB85d32Cf465507DD71d507100C1d"]
```

4.2 subscribe

For `web3.eth.subscribe` see the Subscribe reference documentation.

4.3 Contract

For `web3.eth.Contract` see the Contract reference documentation.

4.4 Iban

For `web3.eth.Iban` see the Iban reference documentation.

4.5 personal

For `web3.eth.personal` see the personal reference documentation.

4.6 accounts

For `web3.eth.accounts` see the accounts reference documentation.

4.7 ens

For `web3.eth.ens` see the ENS reference documentation.

4.8 abi

For `web3.eth.abi` see the ABI reference documentation.

4.9 net

For `web3.eth.net` see the net reference documentation.

4.10 defaultAccount

```
web3.eth.defaultAccount
```

This default address is used as the default `"from"` property, if no `"from"` property is specified in for the following methods:

- *web3.eth.sendTransaction()*
- web3.eth.call()
- new web3.eth.Contract() -> myContract.methods.myMethod().call()
- new web3.eth.Contract() -> myContract.methods.myMethod().send()

4.10.1 Property

`String` - 20 Bytes: Any ethereum address. You should have the private key for that address in your node or keystore. (Default is `undefined`)

4.10.2 Example

```
web3.eth.defaultAccount;
> undefined

// set the default account
web3.eth.defaultAccount = '0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe';
```

4.11 defaultBlock

```
web3.eth.defaultBlock
```

The default block is used for certain methods. You can override it by passing in the defaultBlock as last parameter. The default value is `"latest"`.

- web3.eth.getBalance()
- web3.eth.getCode()
- *web3.eth.getTransactionCount()*
- web3.eth.getStorageAt()
- web3.eth.call()
- new web3.eth.Contract() -> myContract.methods.myMethod().call()

4.11.1 Property

Default block parameters can be one of the following:

- `Number|BN|BigNumber`: A block number
- `"earliest"` - `String`: The genesis block
- `"latest"` - `String`: The latest block (current head of the blockchain)
- `"pending"` - `String`: The currently mined block (including pending transactions)

Default is `"latest"`

4.11.2 Example

```
web3.eth.defaultBlock;
> "latest"

// set the default block
web3.eth.defaultBlock = 231;
```

4.12 defaultHardfork

```
web3.eth.defaultHardfork
```

The default hardfork property is used for signing transactions locally.

4.12.1 Property

The default hardfork property can be one of the following:

- `"chainstart"` - `String`
- `"homestead"` - `String`
- `"dao"` - `String`
- `"tangerineWhistle"` - `String`
- `"spuriousDragon"` - `String`
- `"byzantium"` - `String`
- `"constantinople"` - `String`
- `"petersburg"` - `String`
- `"istanbul"` - `String`

Default is `"petersburg"`

4.12.2 Example

```
web3.eth.defaultHardfork;
> "petersburg"

// set the default block
web3.eth.defaultHardfork = 'istanbul';
```

4.13 defaultChain

```
web3.eth.defaultChain
```

The default chain property is used for signing transactions locally.

4.13.1 Property

The default chain property can be one of the following:

- `"mainnet"` - `String`
- `"goerli"` - `String`
- `"kovan"` - `String`
- `"rinkeby"` - `String`
- `"ropsten"` - `String`

Default is `"mainnet"`

4.13.2 Example

```
web3.eth.defaultChain;
> "mainnet"

// set the default chain
web3.eth.defaultChain = 'goerli';
```

4.14 defaultCommon

```
web3.eth.defaultCommon
```

The default common property is used for signing transactions locally.

4.14.1 Property

The default common property does contain the following `Common` object:

- **`customChain` - `Object`: The custom chain properties**
  - `name` - `string`: (optional) The name of the chain
  - `networkId` - `number`: Network ID of the custom chain

  - `chainId` - `number`: Chain ID of the custom chain
- `baseChain` - `string`: (optional) `mainnet`, `goerli`, `kovan`, `rinkeby`, or `ropsten`
- `hardfork` - `string`: (optional) `chainstart`, `homestead`, `dao`, `tangerineWhistle`, `spuriousDragon`, `byzantium`, `constantinople`, `petersburg`, or `istanbul`

Default is `undefined`.

4.14.2 Example

```
web3.eth.defaultCommon;
> {customChain: {name: 'custom-network', chainId: 1, networkId: 1}, baseChain:
↪'mainnet', hardfork: 'petersburg'}

// set the default common
web3.eth.defaultCommon = {customChain: {name: 'custom-network', chainId: 1,␣
↪networkId: 1}, baseChain: 'mainnet', hardfork: 'petersburg'};
```

4.15 transactionBlockTimeout

```
web3.eth.transactionBlockTimeout
```

The `transactionBlockTimeout` is used over socket-based connections. This option defines the amount of new blocks it should wait until the first confirmation happens, otherwise the PromiEvent rejects with a timeout error.

4.15.1 Returns

`number`: The current value of transactionBlockTimeout (default: 50)

4.15.2 Example

```
web3.eth.transactionBlockTimeout;
> 50

// set the transaction block timeout
web3.eth.transactionBlockTimeout = 100;
```

4.16 transactionConfirmationBlocks

```
web3.eth.transactionConfirmationBlocks
```

This defines the number of blocks it requires until a transaction is considered confirmed.

4.16.1 Returns

`number`: The current value of transactionConfirmationBlocks (default: 24)

4.16.2 Example

```
web3.eth.transactionConfirmationBlocks;
> 24

// set the transaction confirmations blocks
web3.eth.transactionConfirmationBlocks = 50;
```

4.17 transactionPollingTimeout

```
web3.eth.transactionPollingTimeout
```

The `transactionPollingTimeout` is used over HTTP connections. This option defines the number of seconds Web3 will wait for a receipt which confirms that a transaction was mined by the network. Note: If this method times out, the transaction may still be pending.

4.17.1 Returns

`number`: The current value of transactionPollingTimeout (default: 750)

4.17.2 Example

```
web3.eth.transactionPollingTimeout;
> 750

// set the transaction polling timeout
web3.eth.transactionPollingTimeout = 1000;
```

4.18 handleRevert

```
web3.eth.handleRevert
```

The `handleRevert` options property defaults to `false` and returns the revert reason string if enabled for the following methods:

- web3.eth.call()
- *web3.eth.sendTransaction()*
- contract.methods.myMethod(...).send(...)
- contract.methods.myMethod(...).call(...)

Note: The revert reason string and signature exist as a property on the returned error.

4.18.1 Returns

`boolean`: The current value of `handleRevert` (default: false)

4.18.2 Example

```
web3.eth.handlRevert;
> false

// turn revert handling on
web3.eth.handleRevert = true;
```

4.19 maxListenersWarningThreshold

```
web3.eth.maxListenersWarningThreshold
```

This defines the threshold above which a warning about the number of event listeners attached to a provider which supports sockets subscriptions will be written to the console. You may see this warning if you call `setProvider` on large numbers of Web3 contract objects.

4.19.1 Returns

`number`: The current value of maxListenersWarningThreshold (default: 100)

4.19.2 Example

```
web3.eth.maxListenersWarningThreshold;
> 100

// set the max listeners warning threshold
web3.eth.maxListenersWarningThreshold = 200;
```

4.20 getProtocolVersion

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

Returns the ethereum protocol version of the node.

4.20.1 Returns

`Promise` returns `String`: the protocol version.

4.20.2 Example

```
web3.eth.getProtocolVersion()
.then(console.log);
> "63"
```

4.21 isSyncing

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

Checks if the node is currently syncing and returns either a syncing object, or `false`.

4.21.1 Returns

`Promise` returns `Object|Boolean` - A sync object when the node is currently syncing or `false`:

- `startingBlock` - `Number`: The block number where the sync started.
- `currentBlock` - `Number`: The block number where the node is currently synced to.
- `highestBlock` - `Number`: The estimated block number to sync to.
- `knownStates` - `Number`: The number of estimated states to download.
- `pulledStates` - `Number`: The number of already downloaded states.

4.21.2 Example

```
web3.eth.isSyncing()
.then(console.log);

> {
    startingBlock: 100,
    currentBlock: 312,
    highestBlock: 512,
    knownStates: 234566,
    pulledStates: 123455
}
```

4.22 getCoinbase

```
getCoinbase([callback])
```

Returns the coinbase address to which mining rewards will go.

4.22.1 Returns

`Promise` returns `String` - bytes 20: The coinbase address set in the node for mining rewards.

4.22.2 Example

```
web3.eth.getCoinbase()
.then(console.log);
> "0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe"
```

4.23 isMining

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

Checks whether the node is mining or not.

4.23.1 Returns

`Promise` returns `Boolean`: `true` if the node is mining, otherwise `false`.

4.23.2 Example

```
web3.eth.isMining()
.then(console.log);
> true
```

4.24 getHashrate

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

Returns the number of hashes per second that the node is mining with.

4.24.1 Returns

`Promise` returns `Number`: Number of hashes per second.

4.24.2 Example

```
web3.eth.getHashrate()
.then(console.log);
> 493736
```

4.25 getGasPrice

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

Returns the current gas price oracle. The gas price is determined by the last few blocks median gas price.

4.25.1 Returns

`Promise` returns `String` - Number string of the current gas price in wei.

See the A note on dealing with big numbers in JavaScript.

4.25.2 Example

```
web3.eth.getGasPrice()
.then(console.log);
> "20000000000"
```

4.26 getAccounts

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

Returns a list of accounts the node controls.

4.26.1 Returns

`Promise` returns `Array` - An array of addresses controlled by node.

4.26.2 Example

```
web3.eth.getAccounts()
.then(console.log);
> ["0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe",
↪"0xDCc6960376d6C6dEa93647383FfB245CfCed97Cf"]
```

4.27 getBlockNumber

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

Returns the current block number.

4.27.1 Returns

`Promise` returns `Number` - The number of the most recent block.

4.27.2 Example

```
web3.eth.getBlockNumber()
.then(console.log);
> 2744
```

4.28 getBalance

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

Get the balance of an address at a given block.

4.28.1 Parameters

1. `String` - The address to get the balance of.
2. `Number|String|BN|BigNumber` - (optional) If you pass this parameter it will not use the default block set with *web3.eth.defaultBlock*. Pre-defined block numbers as `"earliest"`, `"latest"` and `"pending"` can also be used.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.28.2 Returns

`Promise` returns `String` - The current balance for the given address in wei.

See the A note on dealing with big numbers in JavaScript.

4.28.3 Example

```
web3.eth.getBalance("0x407d73d8a49eeb85d32cf465507dd71d507100c1")
.then(console.log);
> "1000000000000"
```

4.29 getStorageAt

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

Get the storage at a specific position of an address.

4.29.1 Parameters

1. `String` - The address to get the storage from.
2. `Number|String|BN|BigNumber` - The index position of the storage.
3. `Number|String|BN|BigNumber` - (optional) If you pass this parameter it will not use the default block set with *web3.eth.defaultBlock*. Pre-defined block numbers as `"earliest"`, `"latest"` and `"pending"` can also be used.
4. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.29.2 Returns

`Promise` returns `String` - The value in storage at the given position.

4.29.3 Example

```
web3.eth.getStorageAt("0x407d73d8a49eeb85d32cf465507dd71d507100c1", 0)
.then(console.log);
> "0x033456732123ffff2342342dd12342434324234234fd234fd23fd4f23d4234"
```

4.30 getCode

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

Get the code at a specific address.

4.30.1 Parameters

1. `String` - The address to get the code from.
2. `Number|String|BN|BigNumber` - (optional) If you pass this parameter it will not use the default block set with *web3.eth.defaultBlock*. Pre-defined block numbers as `"earliest"`, `"latest"` and `"pending"` can also be used.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.30.2 Returns

`Promise` returns `String` - The data at given address `address`.

4.30.3 Example

```
web3.eth.getCode("0xd5677cf67b5aa051bb40496e68ad359eb97cfbf8")
.then(console.log);
>
↪"0x600160008035811a818181146012578301005b601b6001356025565b8060005260206000f25b600060078202905091900
↪"
```

4.31 getBlock

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

Returns a block matching the block number or block hash.

4.31.1 Parameters

1. `String|Number|BN|BigNumber` - The block number or block hash. Or the string `"earliest"`, `"latest"` or `"pending"` as in the *default block parameter*.
2. `Boolean` - (optional, default `false`) If specified `true`, the returned block will contain all transactions as objects. If `false` it will only contains the transaction hashes.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.31.2 Returns

`Promise` returns `Object` - The block object:

- `number` - `Number`: The block number. `null` if a pending block.
- `hash` 32 Bytes - `String`: Hash of the block. `null` if a pending block.
- `parentHash` 32 Bytes - `String`: Hash of the parent block.
- `nonce` 8 Bytes - `String`: Hash of the generated proof-of-work. `null` if a pending block.
- `sha3Uncles` 32 Bytes - `String`: SHA3 of the uncles data in the block.
- `logsBloom` 256 Bytes - `String`: The bloom filter for the logs of the block. `null` if a pending block.
- `transactionsRoot` 32 Bytes - `String`: The root of the transaction trie of the block.
- `stateRoot` 32 Bytes - `String`: The root of the final state trie of the block.
- `miner` - `String`: The address of the beneficiary to whom the mining rewards were given.
- `difficulty` - `String`: Integer of the difficulty for this block.
- `totalDifficulty` - `String`: Integer of the total difficulty of the chain until this block.
- `extraData` - `String`: The “extra data” field of this block.
- `size` - `Number`: Integer the size of this block in bytes.
- `gasLimit` - `Number`: The maximum gas allowed in this block.
- `gasUsed` - `Number`: The total used gas by all transactions in this block.

- `timestamp` - `Number`: The unix timestamp for when the block was collated.
- `transactions` - `Array`: Array of transaction objects, or 32 Bytes transaction hashes depending on the `returnTransactionObjects` parameter.
- `uncles` - `Array`: Array of uncle hashes.

4.31.3 Example

```
web3.eth.getBlock(3150)
.then(console.log);

> {
    "number": 3,
    "hash": "0xef95f2f1ed3ca60b048b4bf67cde2195961e0bba6f70bcbea9a2c4e133e34b46",
    "parentHash": "0x2302e1c0b972d00932deb5dab9eb2982f570597d9d42504c05d9c2147eaf9c88
↪",
    "nonce": "0xfb6e1a62d119228b",
    "sha3Uncles": "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347
↪",
    "logsBloom":
↪"0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
↪",
    "transactionsRoot":
↪"0x3a1b03875115b79539e5bd33fb00d8f7b7cd61929d5a3c574f507b8acf415bee",
    "stateRoot": "0xf1133199d44695dfa8fd1bcfe424d82854b5cebef75bddd7e40ea94cda515bcb",
    "miner": "0x8888f1f195afa192cfee860698584c030f4c9db1",
    "difficulty": '21345678965432',
    "totalDifficulty": '324567845321',
    "size": 616,
    "extraData": "0x",
    "gasLimit": 3141592,
    "gasUsed": 21662,
    "timestamp": 1429287689,
    "transactions": [
        "0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b"
    ],
    "uncles": []
}
```

4.32 getBlockTransactionCount

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

Returns the number of transaction in a given block.

4.32.1 Parameters

1. `String|Number|BN|BigNumber` - The block number or hash. Or the string `"earliest"`, `"latest"` or `"pending"` as in the *default block parameter*.
2. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.32.2 Returns

`Promise` returns `Number` - The number of transactions in the given block.

4.32.3 Example

```
web3.eth.getBlockTransactionCount("0x407d73d8a49eeb85d32cf465507dd71d507100c1")
.then(console.log);
> 1
```

4.33 getBlockUncleCount

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

Returns the number of uncles in a block from a block matching the given block hash.

4.33.1 Parameters

1. `String|Number|BN|BigNumber` - The block number or hash. Or the string `"earliest"`, `"latest"` or `"pending"` as in the *default block parameter*.
2. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.33.2 Returns

`Promise` returns `Number` - The number of transactions in the given block.

4.33.3 Example

```
web3.eth.getBlockUncleCount("0x407d73d8a49eeb85d32cf465507dd71d507100c1")
.then(console.log);
> 1
```

4.34 getUncle

```
web3.eth.getUncle(blockHashOrBlockNumber, uncleIndex [, returnTransactionObjects] [,␣
↪callback])
```

Returns a blocks uncle by a given uncle index position.

4.34.1 Parameters

1. `String|Number|BN|BigNumber` - The block number or hash. Or the string `"earliest"`, `"latest"` or `"pending"` as in the *default block parameter*.
2. `Number` - The index position of the uncle.
3. `Boolean` - (optional, default `false`) If specified `true`, the returned block will contain all transactions as objects. By default it is `false` so, there is no need to explictly specify false. And, if `false` it will only contains the transaction hashes.
4. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.34.2 Returns

`Promise` returns `Object` - the returned uncle. For a return value see *web3.eth.getBlock()*.

Note: An uncle doesn't contain individual transactions.

4.34.3 Example

```
web3.eth.getUncle(500, 0)
.then(console.log);
> // see web3.eth.getBlock
```

4.35 getTransaction

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

Returns a transaction matching the given transaction hash.

4.35.1 Parameters

1. `String` - The transaction hash.
2. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.35.2 Returns

`Promise` returns `Object` - A transaction object its hash `transactionHash`:

- `hash` 32 Bytes - `String`: Hash of the transaction.
- `nonce` - `Number`: The number of transactions made by the sender prior to this one.
- `blockHash` 32 Bytes - `String`: Hash of the block where this transaction was in. `null` if pending.
- `blockNumber` - `Number`: Block number where this transaction was in. `null` if pending.
- `transactionIndex` - `Number`: Integer of the transactions index position in the block. `null` if pending.

- `from` - `String`: Address of the sender.
- `to` - `String`: Address of the receiver. `null` if it's a contract creation transaction.
- `value` - `String`: Value transferred in wei.
- `gasPrice` - `String`: Gas price provided by the sender in wei.
- `gas` - `Number`: Gas provided by the sender.
- `input` - `String`: The data sent along with the transaction.

4.35.3 Example

```
web3.eth.getTransaction(
↪'0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b§234')
.then(console.log);

> {
    "hash": "0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b",
    "nonce": 2,
    "blockHash": "0xef95f2f1ed3ca60b048b4bf67cde2195961e0bba6f70bcbea9a2c4e133e34b46",
    "blockNumber": 3,
    "transactionIndex": 0,
    "from": "0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b",
    "to": "0x6295ee1b4f6dd65047762f924ecd367c17eabf8f",
    "value": '123450000000000000',
    "gas": 314159,
    "gasPrice": '2000000000000',
    "input": "0x57cb2fc4"
}
```

4.36 getPendingTransactions

```
web3.eth.getPendingTransactions([, callback])
```

Returns a list of pending transactions.

4.36.1 Parameters

1. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.36.2 Returns

`Promise<object[]>` - Array of pending transactions:

- `hash` 32 Bytes - `String`: Hash of the transaction.
- `nonce` - `Number`: The number of transactions made by the sender prior to this one.
- `blockHash` 32 Bytes - `String`: Hash of the block where this transaction was in. `null` if pending.
- `blockNumber` - `Number`: Block number where this transaction was in. `null` if pending.

- `transactionIndex` - `Number`: Integer of the transactions index position in the block. `null` if pending.
- `from` - `String`: Address of the sender.
- `to` - `String`: Address of the receiver. `null` when it's a contract creation transaction.
- `value` - `String`: Value transferred in wei.
- `gasPrice` - `String`: The wei per unit of gas provided by the sender in wei.
- `gas` - `Number`: Gas provided by the sender.
- `input` - `String`: The data sent along with the transaction.

4.36.3 Example

```
web3.eth.getPendingTransactions().then(console.log);
> [
    {
        hash: '0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b',
        nonce: 2,
        blockHash:
↪'0xef95f2f1ed3ca60b048b4bf67cde2195961e0bba6f70bcbea9a2c4e133e34b46',
        blockNumber: 3,
        transactionIndex: 0,
        from: '0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b',
        to: '0x6295ee1b4f6dd65047762f924ecd367c17eabf8f',
        value: '123450000000000000',
        gas: 314159,
        gasPrice: '2000000000000',
        input: '0x57cb2fc4'
        v: '0x3d',
        r: '0xaabc9ddafffb2ae0bac4107697547d22d9383667d9e97f5409dd6881ce08f13f',
        s: '0x69e43116be8f842dcd4a0b2f760043737a59534430b762317db21d9ac8c5034'
    },....,{
        hash: '0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b',
        nonce: 3,
        blockHash:
↪'0xef95f2f1ed3ca60b048b4bf67cde2195961e0bba6f70bcbea9a2c4e133e34b46',
        blockNumber: 4,
        transactionIndex: 0,
        from: '0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b',
        to: '0x6295ee1b4f6dd65047762f924ecd367c17eabf8f',
        value: '123450000000000000',
        gas: 314159,
        gasPrice: '2000000000000',
        input: '0x57cb2fc4'
        v: '0x3d',
        r: '0xaabc9ddafffb2ae0bac4107697547d22d9383667d9e97f5409dd6881ce08f13f',
        s: '0x69e43116be8f842dcd4a0b2f760043737a59534430b762317db21d9ac8c5034'
    }
]
```

4.37 getTransactionFromBlock

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

Returns a transaction based on a block hash or number and the transaction's index position.

4.37.1 Parameters

1. `String|Number|BN|BigNumber` - A block number or hash. Or the string `"earliest"`, `"latest"` or `"pending"` as in the *default block parameter*.
2. `Number` - The transaction's index position.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.37.2 Returns

`Promise` returns `Object` - A transaction object, see *web3.eth.getTransaction*:

4.37.3 Example

```
var transaction = web3.eth.getTransactionFromBlock('0x4534534534', 2)
.then(console.log);
> // see web3.eth.getTransaction
```

4.38 getTransactionReceipt

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

Returns the receipt of a transaction by transaction hash.

Note: The receipt is not available for pending transactions and returns `null`.

4.38.1 Parameters

1. `String` - The transaction hash.
2. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.38.2 Returns

`Promise` returns `Object` - A transaction receipt object, or `null` if no receipt was found:

- `status` - `Boolean`: `TRUE` if the transaction was successful, `FALSE` if the EVM reverted the transaction.
- `blockHash` 32 Bytes - `String`: Hash of the block where this transaction was in.

- `blockNumber` - `Number`: Block number where this transaction was in.
- `transactionHash` 32 Bytes - `String`: Hash of the transaction.
- `transactionIndex`- `Number`: Integer of the transactions index position in the block.
- `from` - `String`: Address of the sender.
- `to` - `String`: Address of the receiver. `null` when it's a contract creation transaction.
- `contractAddress` - `String`: The contract address created, if the transaction was a contract creation, otherwise `null`.
- `cumulativeGasUsed` - `Number`: The total amount of gas used when this transaction was executed in the block.
- `gasUsed`- `Number`: The amount of gas used by this specific transaction alone.
- `logs` - `Array`: Array of log objects, which this transaction generated.

4.38.3 Example

```
var receipt = web3.eth.getTransactionReceipt(
↪'0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b')
.then(console.log);

> {
  "status": true,
  "transactionHash":
↪"0x9fc76417374aa880d4449a1f7f31ec597f00b1f6f3dd2d66f4c9c6c445836d8b",
  "transactionIndex": 0,
  "blockHash": "0xef95f2f1ed3ca60b048b4bf67cde2195961e0bba6f70bcbea9a2c4e133e34b46",
  "blockNumber": 3,
  "contractAddress": "0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe",
  "cumulativeGasUsed": 314159,
  "gasUsed": 30234,
  "logs": [{
         // logs as returned by getPastLogs, etc.
     }, ...]
}
```

4.39 getTransactionCount

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

Get the number of transactions sent from this address.

4.39.1 Parameters

1. `String` - The address to get the numbers of transactions from.
2. `Number|String|BN|BigNumber` - (optional) If you pass this parameter it will not use the default block set with *web3.eth.defaultBlock*. Pre-defined block numbers as `"earliest"`, `"latest"` and `"pending"` can also be used.

3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.39.2 Returns

`Promise` returns `Number` - The number of transactions sent from the given address.

4.39.3 Example

```
web3.eth.getTransactionCount("0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe")
.then(console.log);
> 1
```

4.40 sendTransaction

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

Sends a transaction to the network.

4.40.1 Parameters

1. **`Object` - The transaction object to send:**
    - `from` - `String|Number`: The address for the sending account. Uses the web3.eth.defaultAccount property, if not specified. Or an address or index of a local wallet in web3.eth.accounts.wallet.
    - `to` - `String`: (optional) The destination address of the message, left undefined for a contract-creation transaction.
    - `value` - `Number|String|BN|BigNumber`: (optional) The value transferred for the transaction in wei, also the endowment if it's a contract-creation transaction.
    - `gas` - `Number`: (optional, default: To-Be-Determined) The amount of gas to use for the transaction (unused gas is refunded).
    - `gasPrice` - `Number|String|BN|BigNumber`: (optional) The price of gas for this transaction in wei, defaults to *web3.eth.gasPrice*.
    - `data` - `String`: (optional) Either a ABI byte string containing the data of the function call on a contract, or in the case of a contract-creation transaction the initialisation code.
    - `nonce` - `Number`: (optional) Integer of the nonce. This allows to overwrite your own pending transactions that use the same nonce.
    - `chain` - `String`: (optional) Defaults to `mainnet`.
    - `hardfork` - `String`: (optional) Defaults to `petersburg`.
    - **`common` - `Object`: (optional) The common object**
        - **`customChain` - `Object`: The custom chain properties**
            - `name` - `string`: (optional) The name of the chain
            - `networkId` - `number`: Network ID of the custom chain

    * chainId - number: Chain ID of the custom chain
  - baseChain - string: (optional) mainnet, goerli, kovan, rinkeby, or ropsten
  - hardfork - string: (optional) chainstart, homestead, dao, tangerineWhistle, spuriousDragon, byzantium, constantinople, petersburg, or istanbul
2. callback - Function: (optional) Optional callback, returns an error object as first parameter and the result as second.

Note: The from property can also be an address or index from the web3.eth.accounts.wallet. It will then sign locally using the private key of that account, and send the transaction via *web3.eth.sendSignedTransaction()*. If the properties chain and hardfork or common are not set, Web3 will try to set appropriate values by querying the network for its chainId and networkId.

4.40.2 Returns

The **callback** will return the 32 bytes transaction hash.

PromiEvent: A promise combined event emitter. Resolves when the transaction *receipt* is available. The following events are also available:

- sending returns payload: Object: Fired immediately before transmitting the transaction request.
- sent returns payload: Object: Fired immediately after the request body has been written to the client, but before the transaction hash is received.
- "transactionHash" returns transactionHash: String: Fired when the transaction hash is available.
- "receipt" returns receipt: Object: Fired when the transaction receipt is available.
- "confirmation" returns confirmationNumber: Number, receipt: Object, latestBlockHash: String: Fired for every confirmation up to the 12th confirmation. Receives the confirmation number as the first and the *receipt* as the second argument. Fired from confirmation 0 on, which is the block where it's mined.

"error" returns error: Error: Fired if an error occurs during sending. If the transaction was rejected by the network with a receipt, the receipt will be available as a property on the error object.

4.40.3 Example

```
// compiled solidity source code using https://remix.ethereum.org
var code =
↪"603d80600c6000396000f3007c01000000000000000000000000000000000000000000000000000000006000350463c68
↪";


// using the callback
web3.eth.sendTransaction({
    from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',
    data: code // deploying a contracrt
}, function(error, hash){
    ...
});
```

(continues on next page)

(continued from previous page)

```
// using the promise
web3.eth.sendTransaction({
    from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',
    to: '0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe',
    value: '1000000000000000'
})
.then(function(receipt){
    ...
});


// using the event emitter
web3.eth.sendTransaction({
    from: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe',
    to: '0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe',
    value: '1000000000000000'
})
.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.
```

4.41 sendSignedTransaction

```
web3.eth.sendSignedTransaction(signedTransactionData [, callback])
```

Sends an already signed transaction, generated for example using web3.eth.accounts.signTransaction.

4.41.1 Parameters

1. `String` - Signed transaction data in HEX format
2. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.41.2 Returns

`PromiEvent`: A promise combined event emitter. Resolves when the transaction *receipt* is available.

Please see the return values for *web3.eth.sendTransaction* for details.

4.41.3 Example

```
var Tx = require('ethereumjs-tx').Transaction;
var privateKey = Buffer.from(
↪'e331b6d69882b4cb4ea581d88e0b604039a3de5967688d3dcffdd2270c0fd109', 'hex');

var rawTx = {
  nonce: '0x00',
  gasPrice: '0x09184e72a000',
  gasLimit: '0x2710',
  to: '0x0000000000000000000000000000000000000000',
  value: '0x00',
  data: '0x7f7465737432000000000000000000000000000000000000000000000000000000600057'
}

var tx = new Tx(rawTx, {'chain':'ropsten'});
tx.sign(privateKey);

var serializedTx = tx.serialize();

// console.log(serializedTx.toString('hex'));
//␣
↪0xf889808609184e72a00082271094000000000000000000000000000000000000000080a47f74657374320000000000000(

web3.eth.sendSignedTransaction('0x' + serializedTx.toString('hex'))
.on('receipt', console.log);

> // see eth.getTransactionReceipt() for details
```

Note: When using *ethereumjs-tx@2.0.0* if you don't specify the parameter *chain* it will use *mainnet* by default.

4.42 sign

```
web3.eth.sign(dataToSign, address [, callback])
```

Signs data using a specific account. This account needs to be unlocked.

4.42.1 Parameters

1. `String` - Data to sign. If it is a string it will be converted using web3.utils.utf8ToHex.
2. `String|Number` - Address to sign data with. Can be an address or the index of a local wallet in web3.eth.accounts.wallet.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.42.2 Returns

`Promise` returns `String` - The signature.

4.42.3 Example

```
web3.eth.sign("Hello world", "0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe")
.then(console.log);
>
↪"0x30755ed65396facf86c53e6217c52b4daebe72aa4941d89635409de4c9c7f9466d4e9aaec7977f05e923889b33c0d0dc
↪"

// the below is the same
web3.eth.sign(web3.utils.utf8ToHex("Hello world"),
↪"0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe")
.then(console.log);
>
↪"0x30755ed65396facf86c53e6217c52b4daebe72aa4941d89635409de4c9c7f9466d4e9aaec7977f05e923889b33c0d0dc
↪"
```

4.43 signTransaction

```
web3.eth.signTransaction(transactionObject, address [, callback])
```

Signs a transaction. This account needs to be unlocked.

4.43.1 Parameters

1. `Object` - The transaction data to sign. See *web3.eth.sendTransaction()* for more.
2. `String` - Address to sign transaction with.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.43.2 Returns

`Promise` returns `Object` - The RLP encoded transaction. The `raw` property can be used to send the transaction using *web3.eth.sendSignedTransaction*.

4.43.3 Example

```
web3.eth.signTransaction({
    from: "0xEB014f8c8B418Db6b45774c326A0E64C78914dC0",
    gasPrice: "20000000000",
    gas: "21000",
    to: '0x3535353535353535353535353535353535353535',
    value: "1000000000000000000",
    data: ""
}).then(console.log);
> {
    raw:
↪'0xf86c808504a817c800825208943535353535353535353535353535353535353535880de0b6b3a76400008025a04f4c17
↪',
    tx: {
```

(continues on next page)

(continued from previous page)

```
        nonce: '0x0',
        gasPrice: '0x4a817c800',
        gas: '0x5208',
        to: '0x3535353535353535353535353535353535353535',
        value: '0xde0b6b3a7640000',
        input: '0x',
        v: '0x25',
        r: '0x4f4c17305743700648bc4f6cd3038ec6f6af0df73e31757007b7f59df7bee88d',
        s: '0x7e1941b264348e80c78c4027afc65a87b0a5e43e86742b8ca0823584c6788fd0',
        hash: '0xda3be87732110de6c1354c83770aae630ede9ac308d9f7b399ecfba23d923384'
    }
}
```

4.44 call

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

Executes a message call transaction, which is directly executed in the VM of the node, but never mined into the blockchain.

4.44.1 Parameters

1. `Object` - A transaction object, see *web3.eth.sendTransaction*. For calls the `from` property is optional however it is highly recommended to explicitly set it or it may default to *address(0)* depending on your node or provider.
2. `Number|String|BN|BigNumber` - (optional) If you pass this parameter it will not use the default block set with *web3.eth.defaultBlock*. Pre-defined block numbers as `"earliest"`, `"latest"` and `"pending"` can also be used.
3. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.44.2 Returns

`Promise` returns `String`: The returned data of the call, e.g. a smart contract functions return value.

4.44.3 Example

```
web3.eth.call({
    to: "0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe", // contract address
    data: "0xc6888fa1000000000000000000000000000000000000000000000000000000000000000300000000000000000000000000000000000000000000000000000000000000003"
})
.then(console.log);
> "0x000000000000000000000000000000000000000000000000000000000000000a"
```

4.45 estimateGas

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

Executes a message call or transaction and returns the amount of the gas used.

4.45.1 Parameters

1. `Object` - A transaction object, see *web3.eth.sendTransaction* with the difference that for calls the `from` property is optional as well.
2. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.45.2 Returns

`Promise` returns `Number` - the used gas for the simulated call/transaction.

4.45.3 Example

```
web3.eth.estimateGas({
    to: "0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe",
    data: "0xc6888fa10000000000000000000000000000000000000000000000000000000000000003"
})
.then(console.log);
> "0x0000000000000000000000000000000000000000000000000000000000000015"
```

4.46 getPastLogs

```
web3.eth.getPastLogs(options [, callback])
```

Gets past logs, matching the given options.

4.46.1 Parameters

1. `Object` - The filter options as follows:

- `fromBlock` - `Number|String`: The number of the earliest block (`"latest"` may be given to mean the most recent and `"pending"` currently mining, block). By default `"latest"`.
- `toBlock` - `Number|String`: The number of the latest block (`"latest"` may be given to mean the most recent and `"pending"` currently mining, block). By default `"latest"`.
- `address` - `String|Array`: An address or a list of addresses to only get logs from particular account(s).
- `topics` - `Array`: An array of values which must each appear in the log entries. The order is important, if you want to leave topics out use `null`, e.g. `[null, '0x12...']`. You can also pass an array for each topic with options for that topic e.g. `[null, ['option1', 'option2']]`

4.46.2 Returns

`Promise` returns `Array` - Array of log objects.

The structure of the returned event `Object` in the `Array` looks as follows:

- `address` - `String`: From which this event originated from.
- `data` - `String`: The data containing non-indexed log parameter.
- `topics` - `Array`: An array with max 4 32 Byte topics, topic 1-3 contains indexed parameters of the log.
- `logIndex` - `Number`: Integer of the event index position in the block.
- `transactionIndex` - `Number`: Integer of the transaction's index position, the event was created in.
- `transactionHash` 32 Bytes - `String`: Hash of the transaction this event was created in.
- `blockHash` 32 Bytes - `String`: Hash of the block where this event was created in. `null` if still pending.
- `blockNumber` - `Number`: The block number where this log was created in. `null` if still pending.

4.46.3 Example

```
web3.eth.getPastLogs({
    address: "0x11f4d0A3c12e86B4b5F39B213F7E19D048276DAe",
    topics: ["0x033456732123ffff2342342dd12342434324234234fd234fd23fd4f23d4234"]
})
.then(console.log);

> [{
    data: '0x7f9fade1c0d57a7af66ab4ead79fade1c0d57a7af66ab4ead7c2c2eb7b11a91385',
    topics: ['0xfd43ade1c09fade1c0d57a7af66ab4ead7c2c2eb7b11a91ffdd57a7af66ab4ead7',
↪'0x7f9fade1c0d57a7af66ab4ead79fade1c0d57a7af66ab4ead7c2c2eb7b11a91385']
    logIndex: 0,
    transactionIndex: 0,
    transactionHash:
↪'0x7f9fade1c0d57a7af66ab4ead79fade1c0d57a7af66ab4ead7c2c2eb7b11a91385',
    blockHash: '0xfd43ade1c09fade1c0d57a7af66ab4ead7c2c2eb7b11a91ffdd57a7af66ab4ead7',
    blockNumber: 1234,
    address: '0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe'
},{...}]
```

4.47 getWork

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

Gets work for miners to mine on. Returns the hash of the current block, the seedHash, and the boundary condition to be met ("target").

4.47.1 Parameters

1. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.47.2 Returns

`Promise` returns `Array` - the mining work with the following structure:

- `String` 32 Bytes - at **index 0**: current block header pow-hash
- `String` 32 Bytes - at **index 1**: the seed hash used for the DAG.
- `String` 32 Bytes - at **index 2**: the boundary condition ("target"), 2^256 / difficulty.

4.47.3 Example

```
web3.eth.getWork()
.then(console.log);
> [
  "0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef",
  "0x5EED0000000000000000000000000005EED0000000000000000000000000000",
  "0xd1ff1c01710000000000000000000000d1ff1c01710000000000000000000000"
]
```

4.48 submitWork

```
web3.eth.submitWork(nonce, powHash, digest, [callback])
```

Used for submitting a proof-of-work solution.

4.48.1 Parameters

1. `String` 8 Bytes: The nonce found (64 bits)
2. `String` 32 Bytes: The header's pow-hash (256 bits)
3. `String` 32 Bytes: The mix digest (256 bits)
4. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.48.2 Returns

`Promise` returns `Boolean` - Returns `TRUE` if the provided solution is valid, otherwise `FALSE`.

4.48.3 Example

```
web3.eth.submitWork([
    "0x0000000000000001",
    "0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef",
    "0xD1FE5700000000000000000000000D1FE5700000000000000000000000000"
])
.then(console.log);
> true
```

4.49 requestAccounts

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

This method will request/enable the accounts from the current environment. This method will only work if you're using the injected provider from a application like Metamask, Status or TrustWallet. It doesn't work if you're connected to a node with a default Web3.js provider (WebsocketProvider, HttpProvidder and IpcProvider).

For more information about the behavior of this method please read EIP-1102: Opt-in account exposure.

4.49.1 Parameters

1. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.49.2 Returns

`Promise<Array>` - Returns an array of enabled accounts.

4.49.3 Example

```
web3.eth.requestAccounts().then(console.log);
> ['0aae0B295369a9FD31d5F28D9Ec85E40f4cb692BAf',
↪'0xde0B295669a9FD93d5F28D9Ec85E40f4cb697BAe']
```

4.50 getChainId

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

Returns the chain ID of the current connected node as described in the EIP-695.

4.50.1 Returns

`Promise<Number>` - Returns chain ID.

4.50.2 Example

```
web3.eth.getChainId().then(console.log);
> 61
```

4.51 getNodeInfo

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

4.51.1 Returns

`Promise<String>` - The current client version.

4.51.2 Example

```
web3.eth.getNodeInfo().then(console.log);
> "Mist/v0.9.3/darwin/go1.4.1"
```

4.52 getProof

```
web3.eth.getProof(address, storageKey, blockNumber, [callback])
```

Returns the account and storage-values of the specified account including the Merkle-proof as described in EIP-1186.

4.52.1 Parameters

1. `String` 20 Bytes: The Address of the account or contract.
2. `Number[] | BigNumber[] | BN[] | String[]` 32 Bytes: Array of storage-keys which should be proofed and included. See web3.eth.getStorageAt.
3. `Number | String | BN | BigNumber`: Integer block number. Pre-defined block numbers as `"earliest"`, `"latest"` and `"pending"` can also be used.
4. `Function` - (optional) Optional callback, returns an error object as first parameter and the result as second.

4.52.2 Returns

`Promise<Object>` - A account object.

- `address` - `String`: The address of the account.
- `balance` - `String`: The balance of the account. See web3.eth.getBalance.
- `codeHash` - `String`: hash of the code of the account. For a simple account without code it will return `"0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470"`.
- `nonce` - `String`: Nonce of the account.
- `storageHash` - `String`: SHA3 of the StorageRoot. All storage will deliver a MerkleProof starting with this rootHash.
- `accountProof` - `String[]`:Array of rlp-serialized MerkleTree-Nodes, starting with the stateRoot-Node, following the path of the SHA3 (address) as key.
- **`storageProof` - `Object[]` Array of storage-entries as requested.**
  - `key` - `String` The requested storage key.

- `value` - `String` The storage value.

4.52.3 Example

```
web3.eth.getProof(
    "0x1234567890123456789012345678901234567890",
    ["0x0000000000000000000000000000000000000000000000000000000000000000",
↪"0x0000000000000000000000000000000000000000000000000000000000000001"],
    "latest"
).then(console.log);
> {
    "address": "0x1234567890123456789012345678901234567890",
    "accountProof": [

↪"0xf90211a090dcaf88c40c7bbc95a912cbdde67c175767b31173df9ee4b0d733bfdd511c43a0babe369f6b12092f49181a
↪",

↪"0xf90211a0395d87a95873cd98c21cf1df9421af03f7247880a2554e20738eec2c7507a494a0bcf6546339a1e7e14eb8fb
↪",

↪"0xf90171a04ad705ea7bf04339fa36b124fa221379bd5a38ffe9a6112cb2d94be3a437b879a08e45b5f72e8149c01efcb7
↪",

↪"0xf851808080a009833150c367df138f1538689984b8a84fc55692d3d41fe4d1e5720ff5483a6980808080808080808080
↪"
    ],
    "balance": 0,
    "codeHash": "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470",
    "nonce": 0,
    "storageHash": "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421
↪",
    "storageProof": [
        {
            "key": "0x0000000000000000000000000000000000000000000000000000000000000000
↪",
            "value": '0',
            "proof": []
        },
        {
            "key": "0x0000000000000000000000000000000000000000000000000000000000000001
↪",
            "value": '0',
            "proof": []
        }
    ]
}
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search