Complete — Simple — Tiny

Richard Moore
@ricmoo

Features

Accounts
private keys, mnemonics, any JSON wallet or JSON-RPC

Complete
create wallets, utilities, frameworks and dapps

Ready
node.js, TypeScript and browser ESM / UMD

Tested
20,000 test cases and growing

Tiny
344kb uncompressed / 98kb over-the-wire

Full ENS Support
ENS is a first-class citizen

Documentation
HTML + Markdown

MIT Licensed
including ALL dependencies

Providers
run your own or safely connect to third-parties

ABIv2
structs and nested dynamic types

Modular
pick and choose what matters to you

CLI
test, debug and manage from command-line scripts

Providers

*A generic API for **account-less** blockchain interaction, regardless of backend...*

Basic Providers

- ✦ JsonRpcProvider
- ✦ IpcProvider
- ✦ Web3Provider
- ✦ FallbackProvider

...and easy to add your own!

Third Party Providers

- ✦ EtherscanProvider
- ✦ InfuraProvider
- ✦ NodeSmithProvider
- ✦ AlchemyProvider
- ✦ CloudflareProvider

...and more coming

Default Provider

Safe!
(But how?)

```
const provider = ethers.getDefaultProvider()
```

Signers

A generic API for creating trusted (i.e. signed) <u>account-based</u> data…

Basic Signers

- ✦ JsonRpcSigner
- ✦ LedgerSigner
- ✦ VoidSigner
- ✦ Wallet
- ✦ FireflySigner

…and easy to add your own!

BIP39 + BIP44

English (en) Español (es)

Français (fr) Italiano (it)

日本語 (ja) 한국어 (ko)

中文 (简体) (zh_cn) 中文 (繁體) (zh_tw)

```
Wallet.fromMnemonic(mnemonic [ , path [ , language ] ])
```

async with progress callback!

```
Wallet.fromEncryptedJson(json, password [ , progress ]).then(...)
wallet.encrypt(password [ , progress ]).then(...)
```

PSA: progress bars make encryption "*feel*" faster, so you can still provide security; also mention "decrypting..."

ENS is a First-Class Citizen

“If ENS were the ONLY thing to come out of Ethereum, I would consider Ethereum to have been a huge success.”

```
new Contract("takoyaki.eth", ABI, provider)
```

```
provider.getBalance("donations.ethers.eth")
```

```
ens.setResolver(namehash("mochi.eth"), "resolver.eth")
```

```
wallet.sendTransaction({
    to: "ricmoo.eth",
    value: parseEther("1.0")
})
```

Deploying a Contract

```
const contractFactory = new ContractFactory(ABI, bytecode, signer)

const contract = await contractFactory.deploy(param0, param1)

// We already have the contract address (not mined yet!)
console.log(contract.address)

// The tx is available immediately (not mined yet!)
const tx = contract.deployTransaction;

// Maybe update a database or local storage?
saveInfo(contract.address, tx.hash);

try {
    // Wait until the contract is deployed...
    const receipt = await contract.deployed();
    console.log("Deployed in block:", receipt.blockNumber);

} catch (error) {
    console.log("Failed to deploy in tx: ", error.transactionHash);
}
```

Contracts

Offline or Deferred Signing

```
txObj = await contract.populateTransaction.transfer("ricmoo.com", 100)
```

Introspective Transaction Execution

```
callResponse = await contract.staticCall.transfer("ricmoo.com", 100)
```

Gas Estimation

```
gasEstimate = await contract.estimate.transfer("ricmoo.com", 100)
```

Transaction Execution

```
tx = await contract.transfer("ricmoo.com", 100)
receipt = await tx.wait()
```

Application Binary Interface (ABI) Formats

JSON

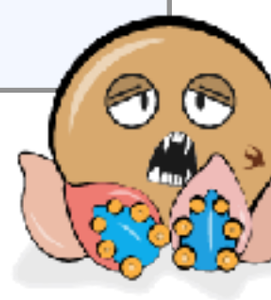
```
{
  "name": "balanceOf",
  "inputs": [
    {
        "name": "owner",
        "type": "address"
    }
  ],
  "outputs": [
    {
      "name": "balance",
      "type": "uint256"
    }
  ],
  "type": "function",
  "mutabilityState": "view",
  "payable": false
}
```

Human-Readable

```
"function balanceOf(address owner) view returns (uint)"
```

Human-Readable (minimal)

```
"function balanceOf(address) view returns (uint)"
```



sighash

```
"balanceOf(address)"
```

Converting between formats

```
Fragment.from(anyAbiStyle).format("json" | "full" | "minimal" | "sighash")
```

ABIv2

Use structures and arrays of dynamic types in your contracts...

```
contract UserRegistry {
    struct User {
        string name;
        string email;
    }
    event UserAdded(address indexed addr, User user);
    function addUser(User user) { ... }
    function getUser(address addr) view returns (User user) { ... }
}
```

```
const abi = [
  "function addUser(tuple(string name, string email) user)",
  "function getUser(address addr) view returns (tuple(string name, string email))",
  "event UserAdded(address indexed addr, tuple(string name, string email) user)"
];
const contract = new Contract(address, abi, provider)
const tx = await contract.addUser({ name: "H. Miyazaki", email: "hayao@ghibli.jp" })
const user = await contract.getUser(addr);
```

Utilities

ABI Coder
flexible, ABIv2, Solidity signature parsing

Binary Data
manipulate binary, hex strings, base32, base58 and base64

Unit Conversion
fixed point maths with parsing and formatting

Signatures
flatten, expand, complete and compress various formats

RLP
encode and decode recursive-length prefix data

Crypto Primitives
ECDH, signing, ecrecover, public key maths

Addresses
convert and compute various address formats

Hashing
KECCAK256 + SHA2, namehash, prefix-messages

Strings
safely manipulate and verify UTF-8 and bytes32 strings

Transactions
parse and serialize any transaction format

HDNode
BIP-32 low-level HD mnemonics, xpub / xpriv

Web Fetching
simple API with exponential back-off and timeouts

Command-Line Interface

```
ethers --account - send-ether ricmoo.eth 1.2
ethers --account - sign-message "Hello world"
ethers --account - wrap-ether 1.0
ethers --account - unwrap-ether 1.0
ethers --account - sweep ricmoo.eth
ethers --account - send-token weth.tokens.ethers.eth ricmoo.eth 1.1
```

```
/home/ricmoo> ethers eval 'parseEther("0.123").toString()'
123000000000000000
```

```
/home/ricmoo> ethers eval 'provider.resolveName("takoyaki.eth")'
0xa39548C44a9bC35B3552E96d23F7185791d9893B
```

```
/home/ricmoo> ethers eval 'provider.getBlock(-1).then(b => b.hash)'
0x5334cc6ec6662ca182320d764a395897c25d4b93e0284138133e698802fa24f7
```

...or use the REPL

Command-Line Interface (ENS)

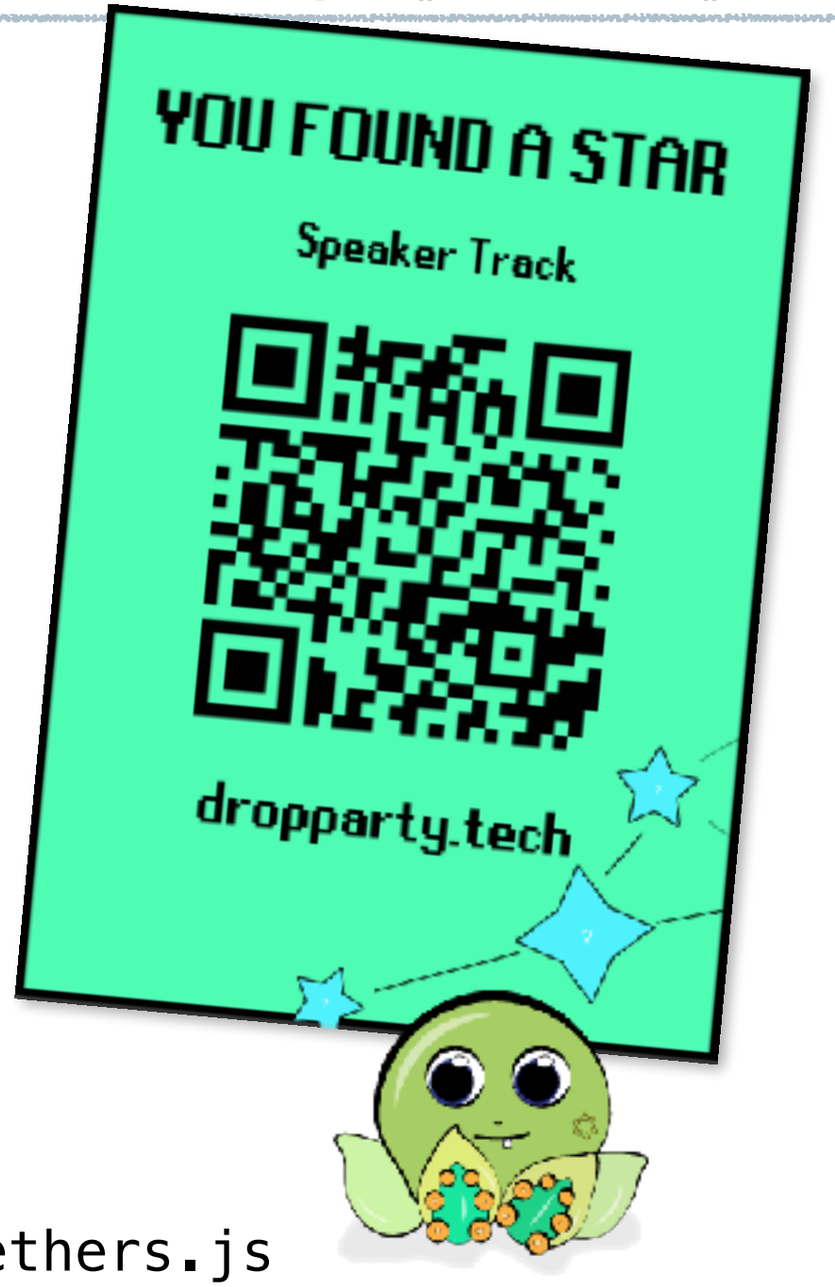
```
Usage:
   ethers-ens COMMAND [ ARGS ] [ OPTIONS ]

COMMANDS
   lookup [ NAME | ADDRESS [ ... ] ]
                                 Lookup an ENS name or address
   commit NAME                   Submit a pre-commitment
   reveal NAME                   Reveal a previous pre-commitment
   set-controller NAME           Set the controller
   set-subnode NAME              Set a subnode owner
   set-resolver NAME             Set the resolver (default: resolver.eth)
   set-addr NAME                 Set the addr record (default: current
   set-text NAME KEY VALUE       Set a text record
   set-email NAME EMAIL          Set the email text record
   set-website NAME URL          Set the website text record
   set-content NAME HASH         Set the IPFS Content Hash
   migrate-registrar NAME        Migrate to the Permanent Registrar
   transfer NAME NEW_OWNER       Transfer registrant ownership
   reclaim NAME                  Reset the controller by the registrant
```

Questions?

Richard Moore

me@ricmoo.com

@ricmoo



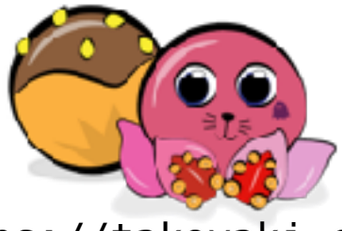
Documentation: https://docs.ethers.io

https://docs-beta.ethers.io

Repository: https://github.com/ethers-io/ethers.js

Related Open-Source Ethereum Projects

https://firefly.city



https://takoyaki.cafe

https://meeseeks.app