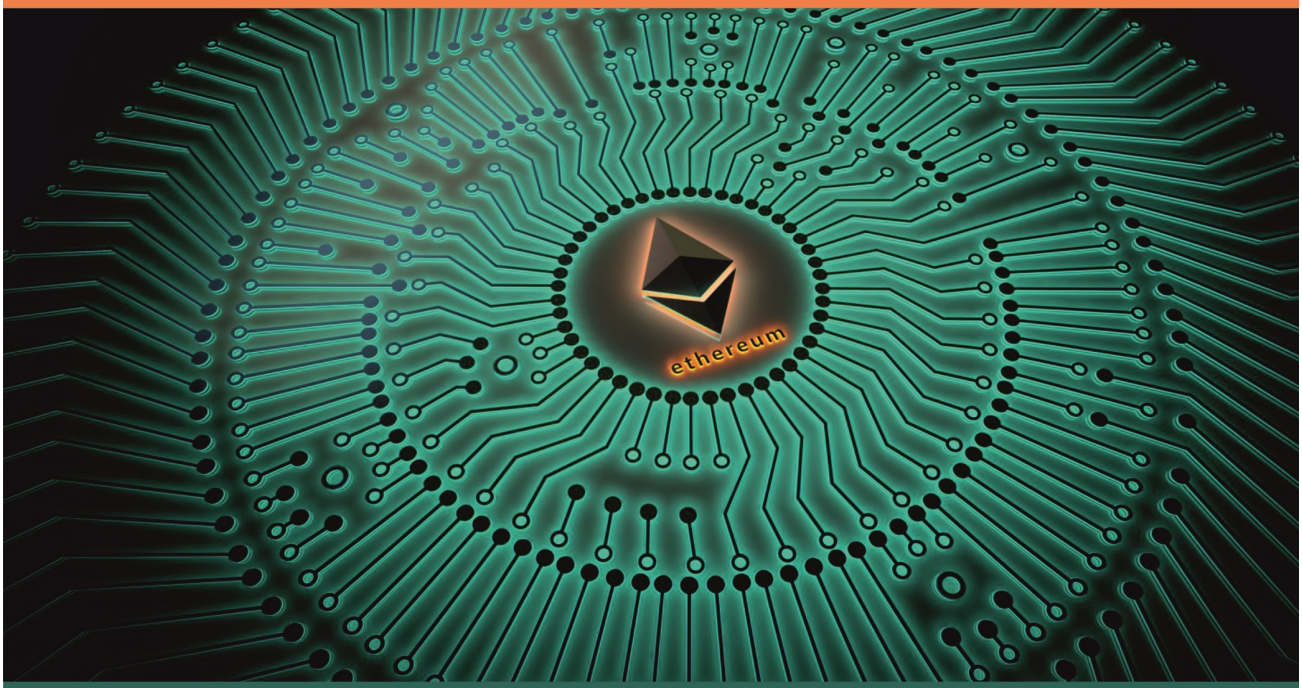
اتریوم و فناوری های مرتبط



Smart Contracts
Solidity Programming
Truffle
IPFS
Oraclize

ترجمه و تالیف

موسی محمدنیا- مریم خدادادی- صفورا سلطانیان


بسم الله الرحمن الرحیم

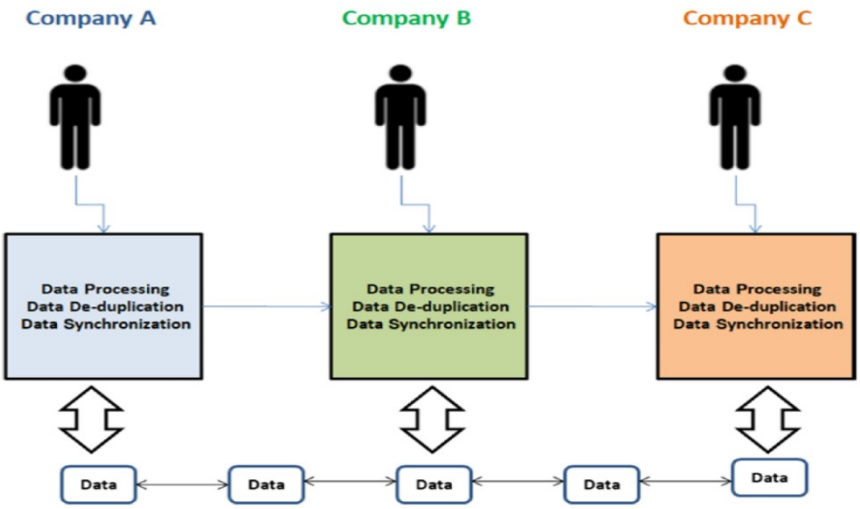
زنجیره‌ی بلوکی اتریوم و فناوری‌های مرتبط

(تصاویر و کدها)

فصل اول

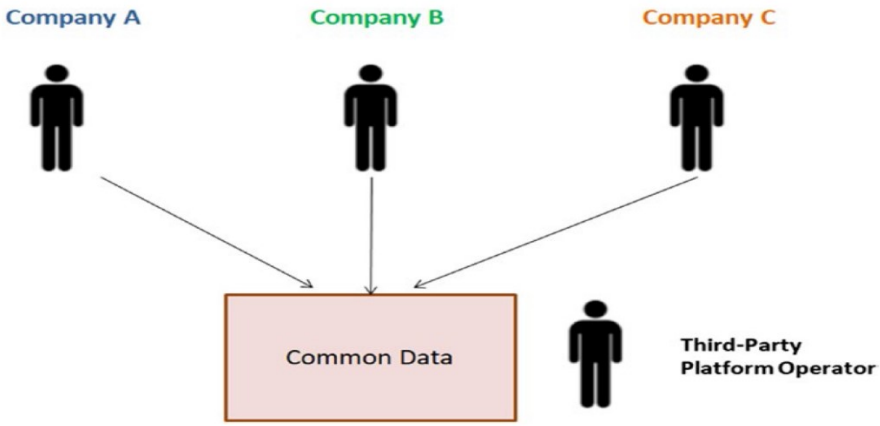
دنیای زنجیره‌ی بلوکی

مدل کاملا توزیع شده



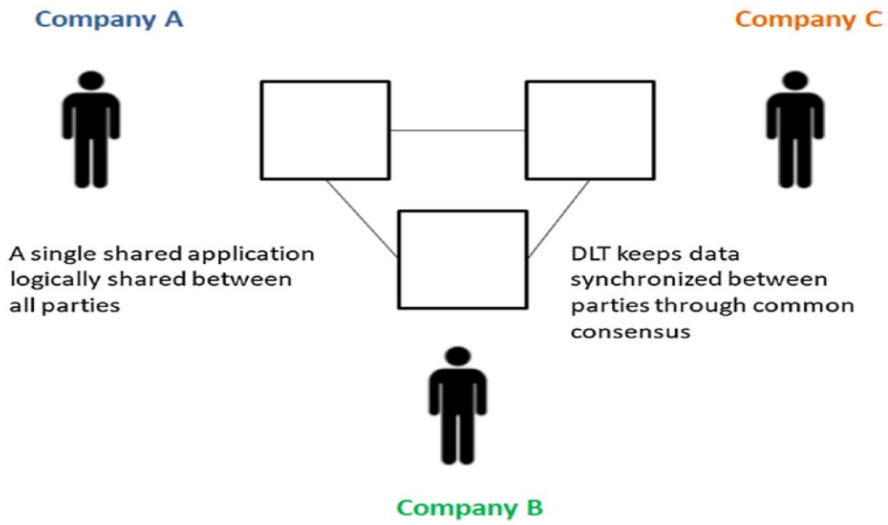
شکل ۱- ۱: سه سازمان در یک حالت توزیع‌شده برای به‌اشتراک‌گذاری داده‌ها کار می‌کنند.

مدل کاملاً متمرکز

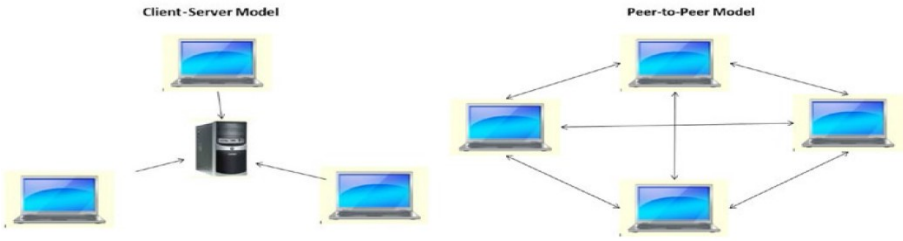


شکل ۱- ۲:. سه سازمان در حالت متمرکز به‌اشتراک‌گذاری داده‌ها کار می‌کنند

DLT یا مدل غیرمتمرکز همتابه‌همتا

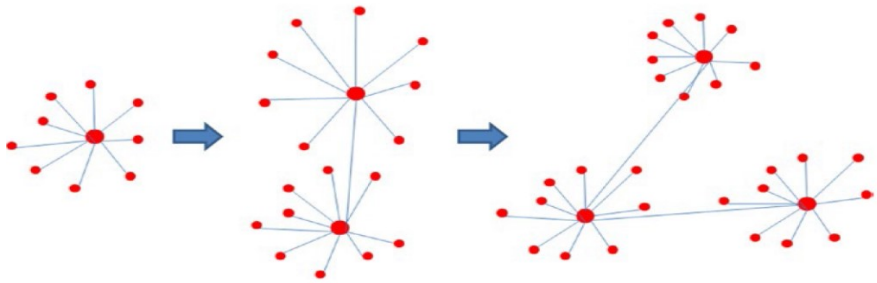


شکل ۱- ۳: سه سازمان در حالت همتابه‌همتا برای به‌اشتراک‌گذاری داده‌ها کار می‌کنند



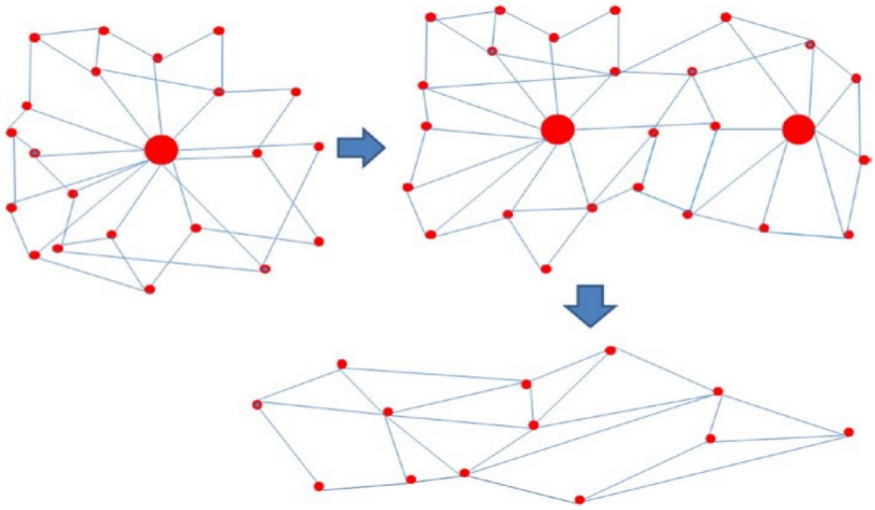
شکل ۱- ٤: مدل مشتری-سرور در مقابل مدل‌های همتابه‌همتا

اکنون می توان بررسی نمود که چگونه می‌توان با تمرکززدایی بیشتر شبکه، این دو الگو را اصلاح کرد (شکل ۱-۵ را ببینید).



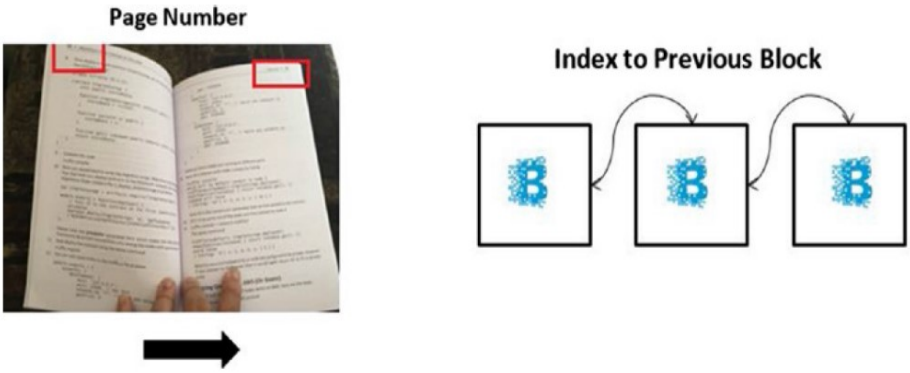
شکل۱- ۵: مدل سرویس دهنده-مشتری در حالت غیرمتمرکزتر

در شکل ۱-٦، نحوه‌ی کار شبکه‌های همتابه‌همتا در صورت عدم‌وجود سرور مرکزی مشاهده می گردد. اکنون که اطمینان حاصل شد چرا به یک فناوری دفترکل توزیع‌شده احتیاج است، درباره‌ی تفاوت‌ها و شباهت‌های زنجیره‌ی بلوکی با DLT بحث می کنیم.

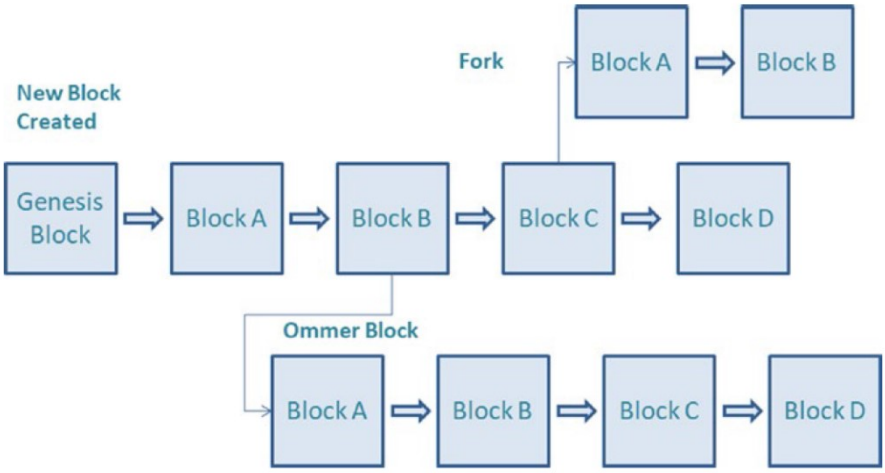


شکل۱- ٦: مدل همتابه‌همتا در حالت کاملاً غیرمتمرکز

تراکنش‌ها و بلوک‌های زنجیره‌ی بلوکی

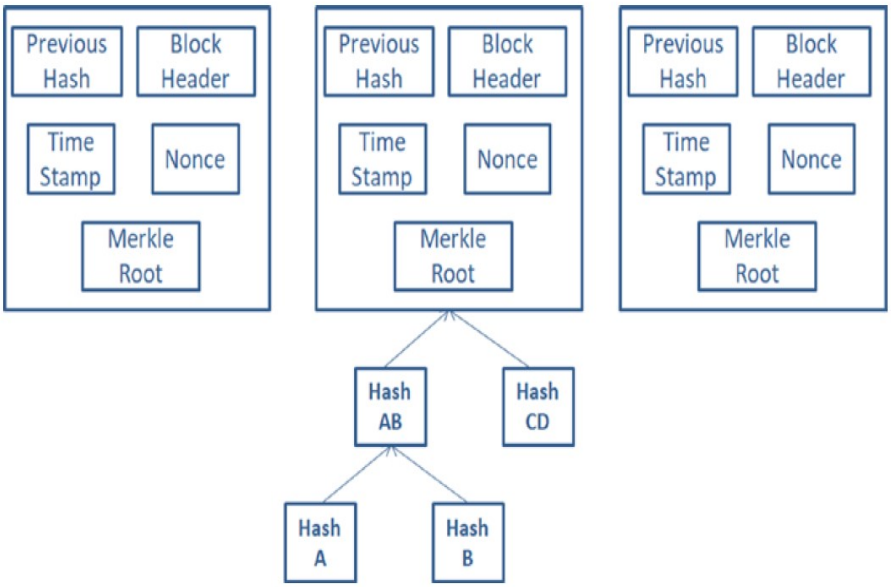


شکل۱– ۷: نمایه‌سازی در زنجیره‌ی بلوکی



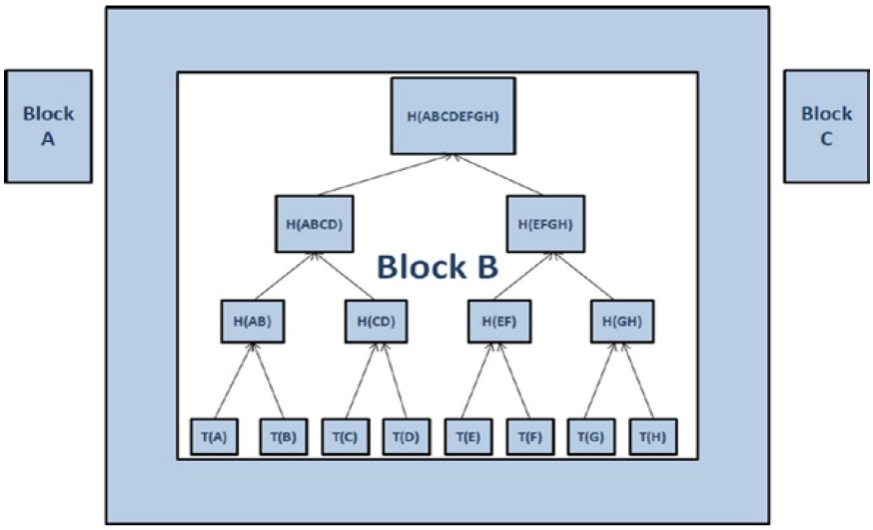
شکل۱– ۸: نحوه تکامل زنجیره‌ی بلوکی.

سرآیند بلوک

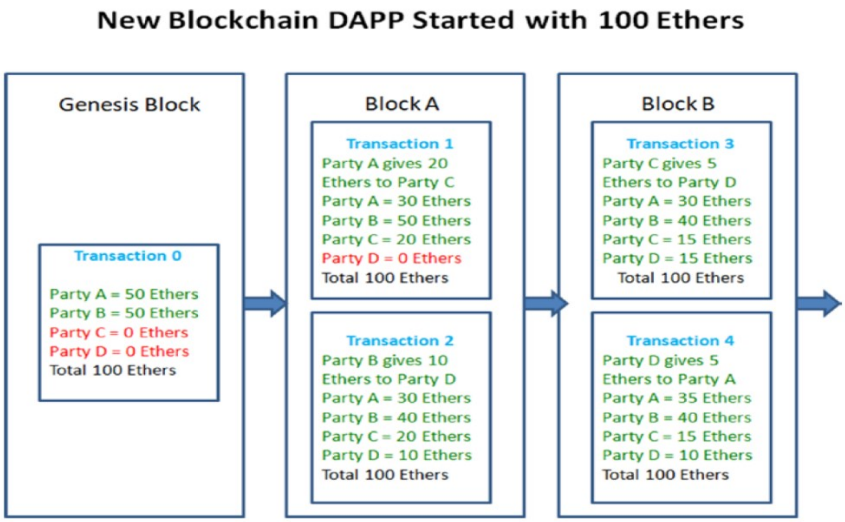


شکل ۱- ۹ : بلوک‌ها در زنجیره‌ی بلوکی

درخت مرکل



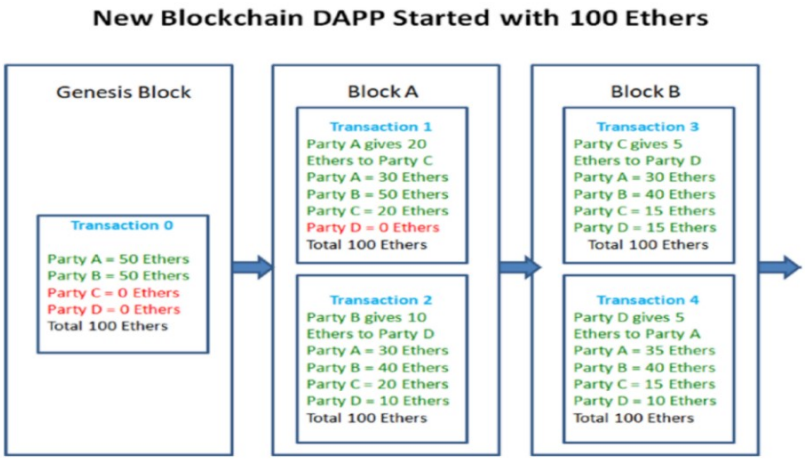
شکل ۱-۱۰: درخت مرکل



شکل ۱-۱۱: ردیابی تراکنش‌ها در زنجیره‌ی بلوکی

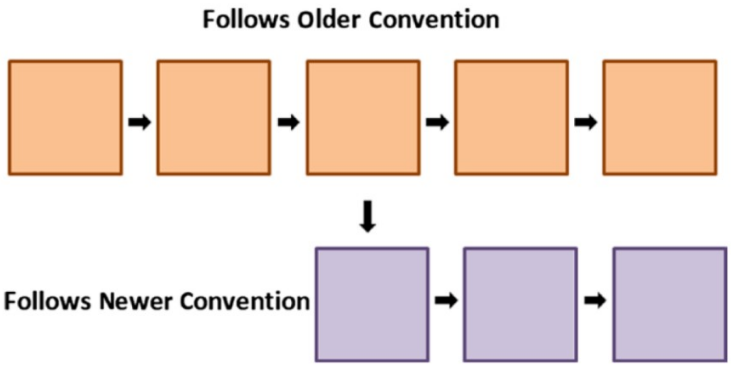
کلیدهای عمومی و خصوصی

شکل ۱-۱۲: کلید های عمومی و خصوصی



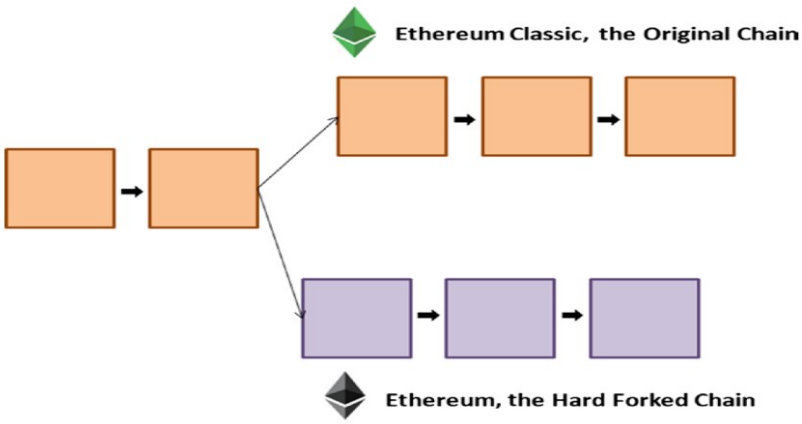
شکل ۱- ۱۳: کلید خصوصی را نمی‌توان از کلید عمومی استخراج کرد.

انشعاب در زنجیره‌ی بلوکی



شکل۱- ١٤: یک انشعاب در زنجیره‌ی بلوکی

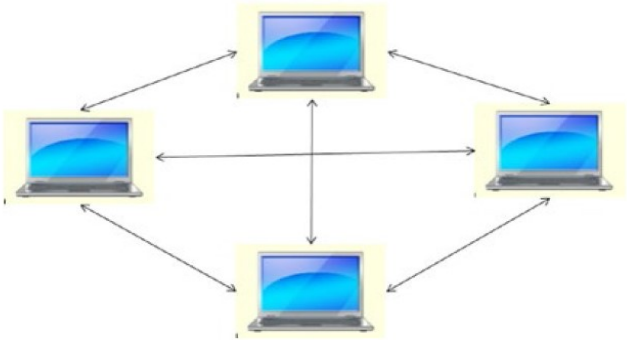
انشعاب‌ها در اتریوم



شکل۱- ١٥: انشعاب‌ها در زنجیره‌ی بلوکی اتریوم

فصل ۲

معماری اتریوم



شکل ۲-۱: شبکه همتابه‌همتا

1 Ethereum = 254.209171 Gas (GAS)

Date (today): 22. May 2021 06:11 AM (GMT)

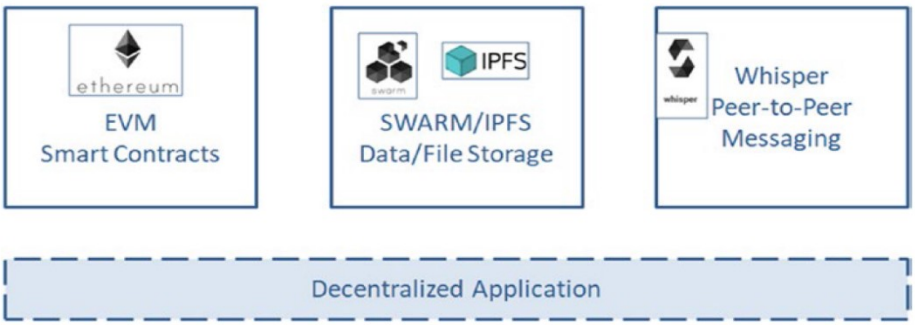
Name	Price USD	Market Cap USD	Available Supply	Volume 24h USD	% 24h	Price Graph (7d)
Ethereum	2210.577	256,368,156,400	115,973,000	52,614,389,640	-18.8086	
Gas	8.696	88,075,373	10,128,400	20,910,600	-17.2159	

1 ETH to GAS (1 Ethereum to Gas) Exchange Calculator

Currency Converter History

شکل ۲-۲: تبدیل Gas به اتر.

اکوسیستم اتریوم کامل



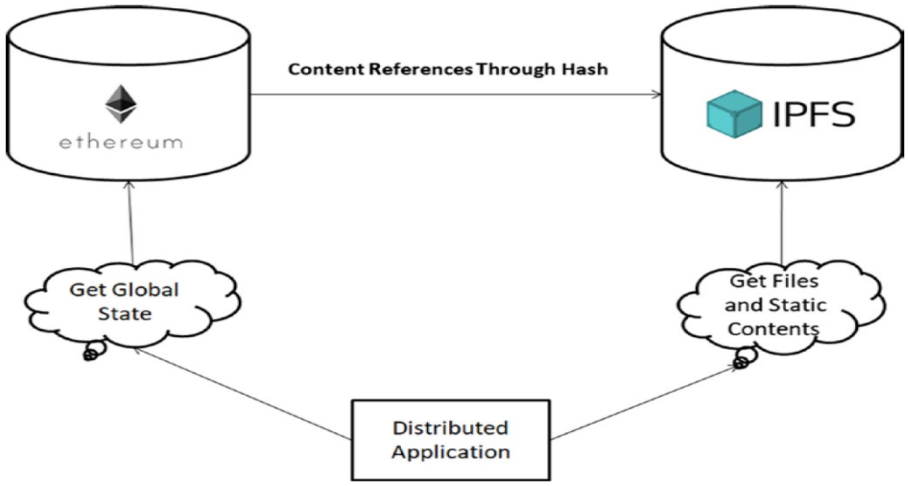
شکل۲-۳: اتریوم اکوسیستم

فایل سیستم بین سیاره‌ای

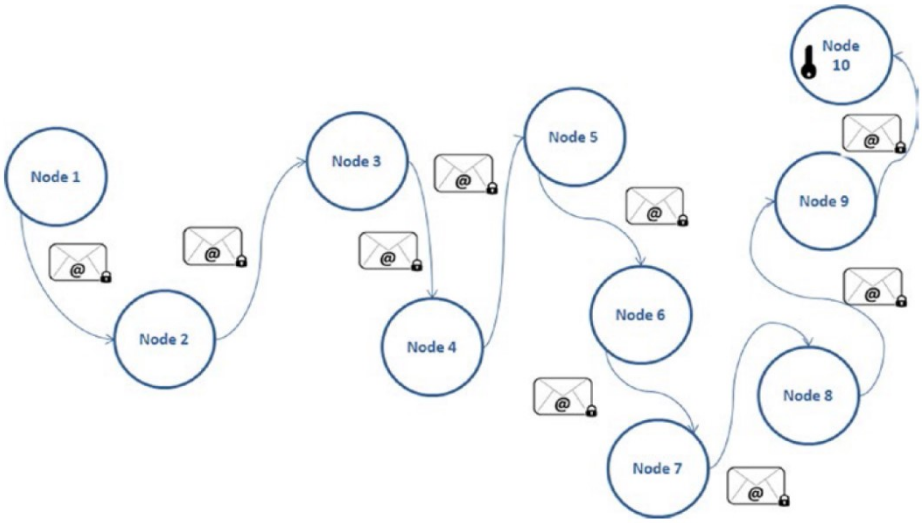
شکل ۲-٤: ذخیره‌سازی داده در IPFS.

شکل ۲-٥: اشتراک‌گذاری داده در IPFS

شکل ۲-٦: ذخیره‌سازی داده در IPFS



شکل۲-۷: IPFS و اتریوم.



شکل ۲-۸: انتقال پیام از طریق پروتکل Whisper.

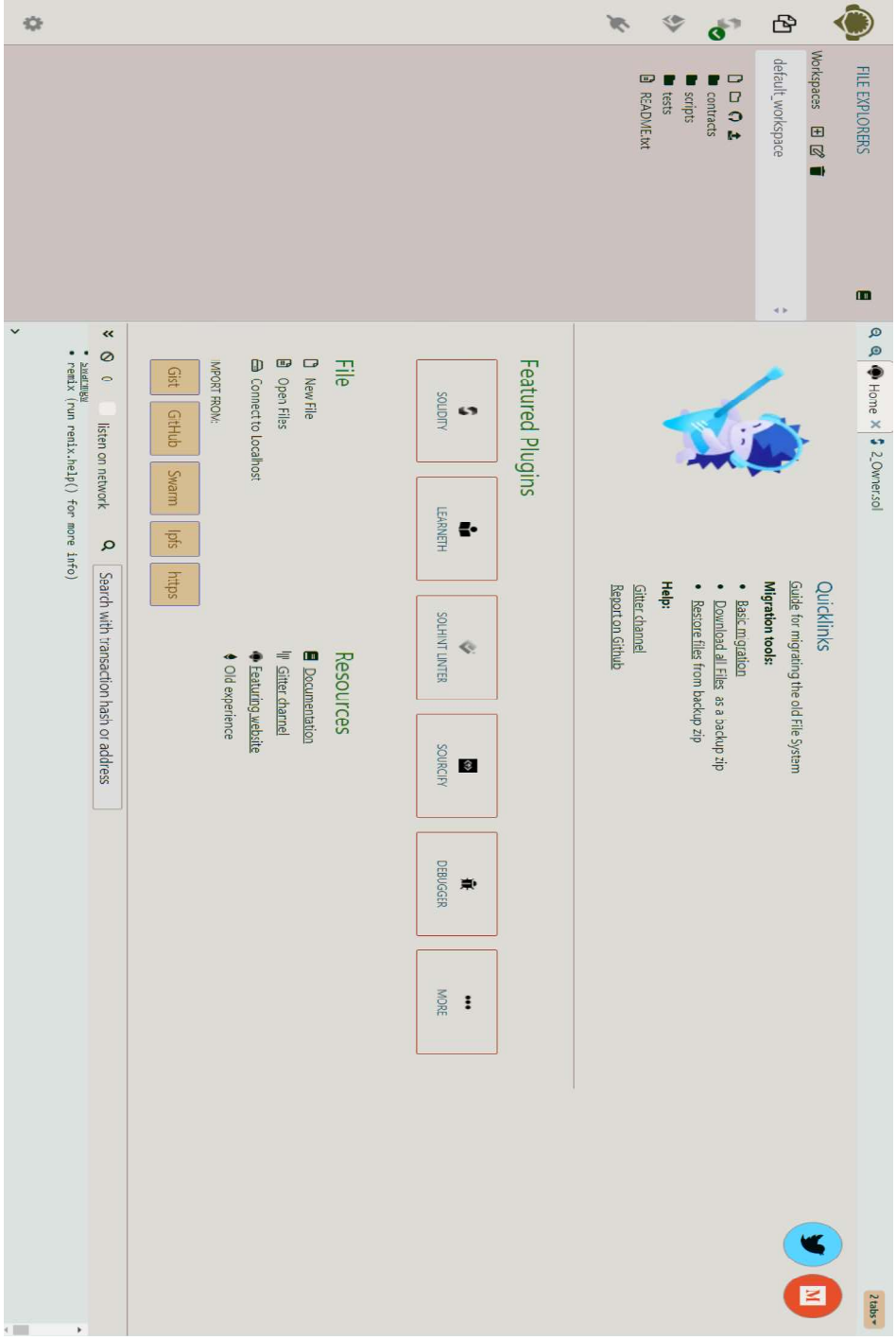
Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

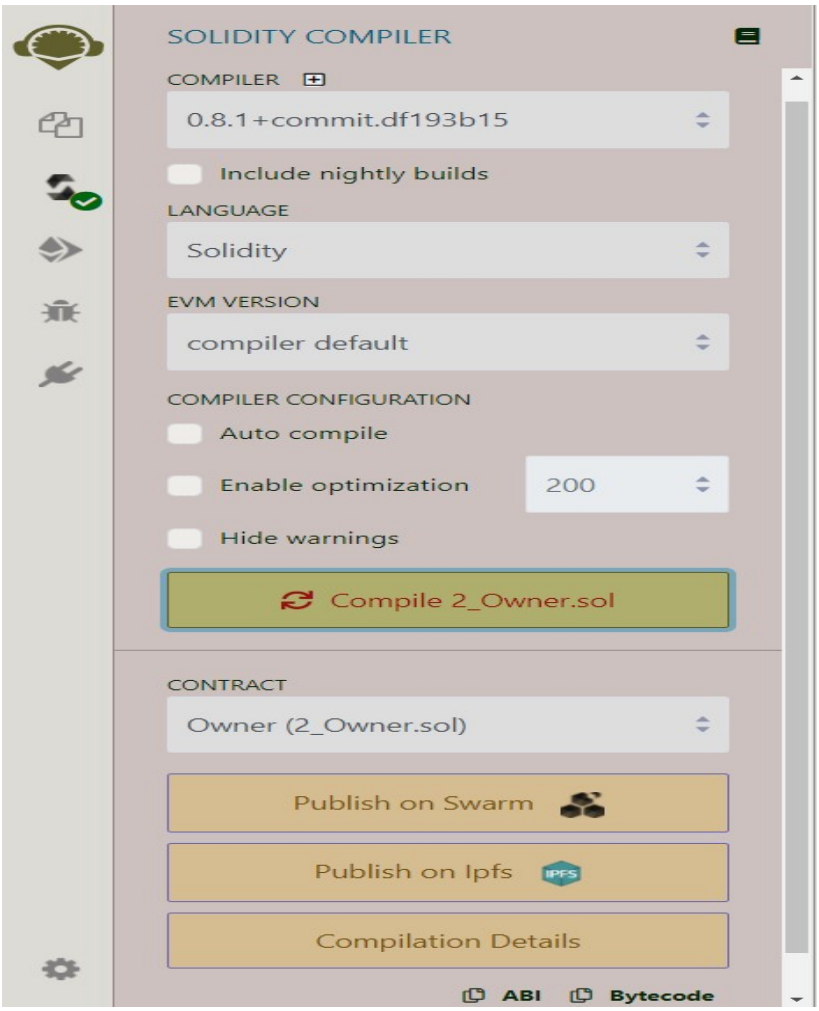
شکل ۲-۹: واحدهای مختلف اتر.

فصل سوم

اصول برنامه‌نویسی پایه‌ی Solidity در اتریوم

شکل۳-۱: مرورگر برخط Remix





شکل ۳-۲: کامپایل برامه در Remix

اگر خطایی رخ دهد، خط موردنظر با علامت ضربدر قرمز رنگ مشخص خواهد شد. (شکل ۳-۳).

```
9   function saveToMemory(-) {
10      string memory myString = "test";
11      assert(bytes(myString).length == 10);
12  }
```

شکل ۳-۳: خطای کامپایلر در Remix

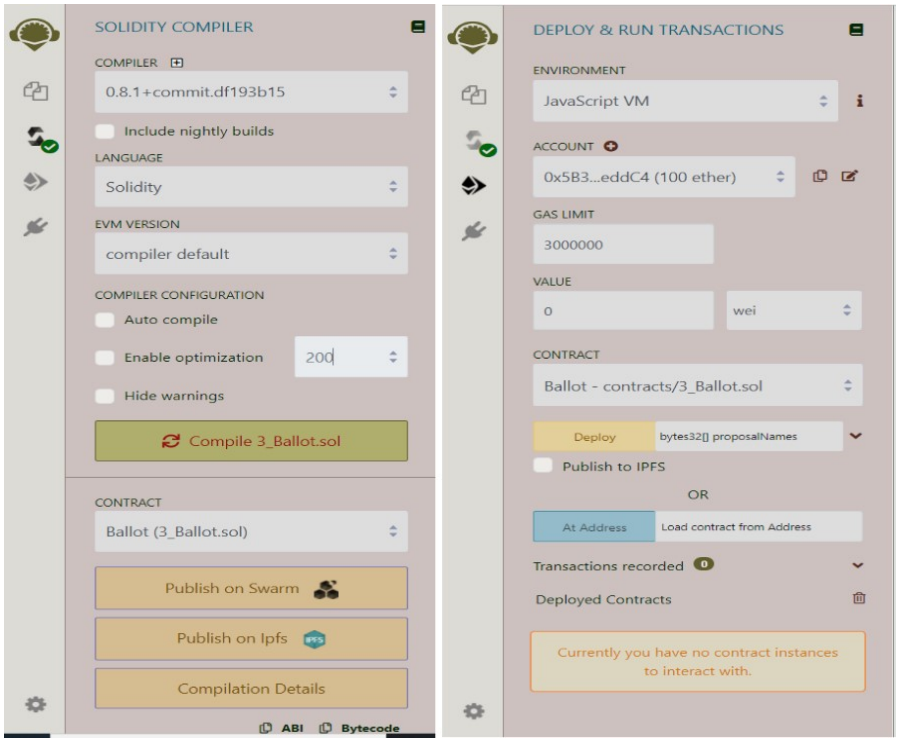
هشدارها نیز با رنگ زرد علامت‌گذاری می‌شوند، اما این هشدارها مانع از اجرای کد نمی‌گردند.

```
9   function saveToMemory() {
10      string memory myString = "test";
11      assert(bytes(myString).length == 10);
12  }
```

شکل ۳-٤: هشدار در برنامه Remix

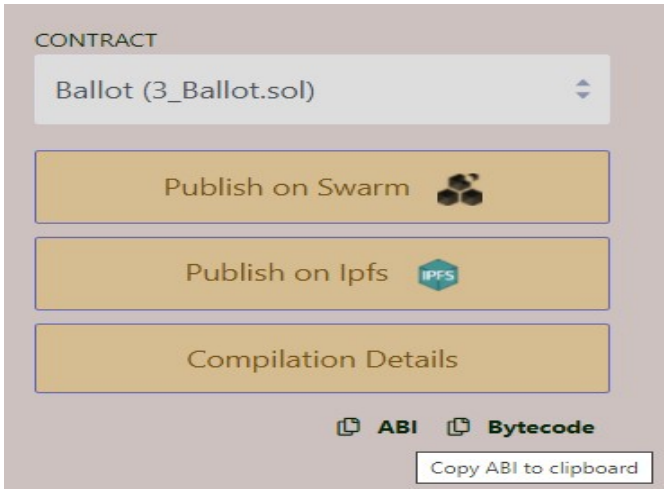
پیاده‌سازیِ قراردادها در Remix

هنگامی‌که قرارداد شما با موفقیت اجرا شد، می‌توانید روی برگه‌ی DEPLOY & RUN TRANSACTIONS در سمت راست کلیک کرده و از منوی گزینه‌ها JavaScript VM را انتخاب کنید (همان حالت پیش‌فرض). در شکل ۳-۵ قرارداد اجرا شده شامل پانزده حساب، با محدودیت gas ۳۰۰۰۰۰۰ برای هرکدام، نشان داده شده است.



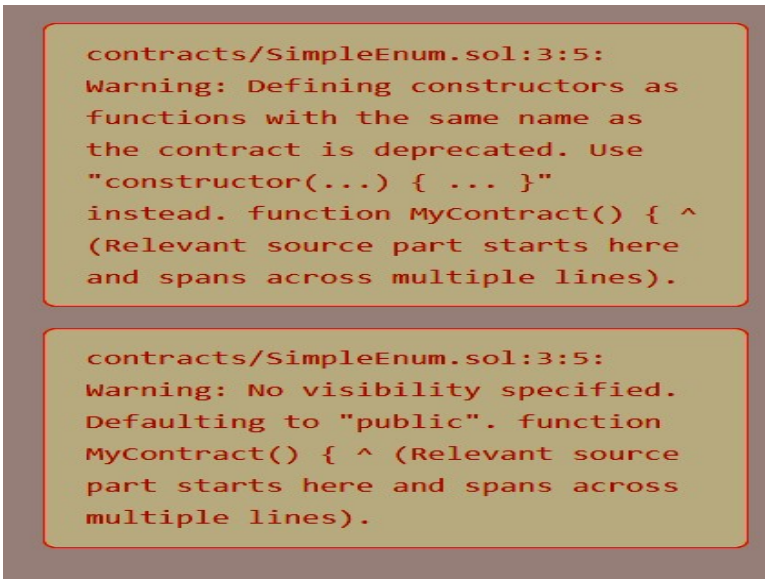
شکل ۳-۵: تصویر سمت چپ: کامپایل کد برنامه‌ی Ballot.sol

تصویر سمت راست: گزینه‌های اجرای همان کد برنامه



شکل ۳-۶: کردن ABI در مرورگر Remix

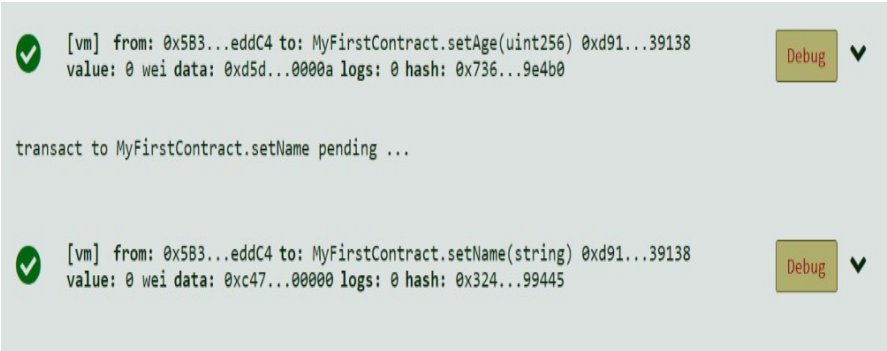
لیست ۳-۱: پرونده‌ی ABI



شکل۳-۷: Warning در مرورگر Remix برای تابع سازنده

شکل ۳-۸: آزمایش عملکرد تنظیم‌کننده در مرورگر Remix

کنسول وب را دقیقاً زیر ویرایشگر کد می‌توانید ببینید (به شکل ۳-۹ نگاه کنید).



شکل ۳-۹: Basic Solidity programming

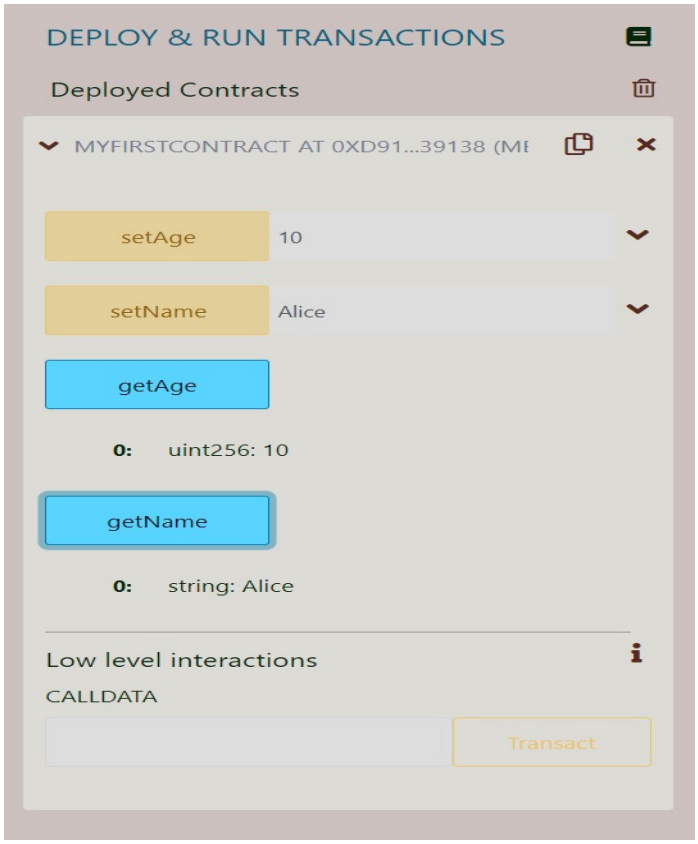
هر بخش را باز کرده و بررسی کنید که آیا مشکلی وجود دارد یا خیر؟ به‌عنوان مثال خصوصیاتی مختلف مانند وضعیت، هزینه Gas و غیره. همچنین به شما امکان رفع اشکال را می‌دهد که در این کتاب بررسی خواهد شد (شکل ۳-۱۰ را ببینید).

status	true Transaction mined and execution succeed
transaction hash	0x3242bb36ee69c33c9923c3effa3ba67cfbf371949bfaf42af2c5d20c3ad99445
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	MyFirstContract.setName(string) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas	3000000 gas
transaction cost	43591 gas
execution cost	21551 gas
hash	0x3242bb36ee69c33c9923c3effa3ba67cfbf371949bfaf42af2c5d20c3ad99445
input	0xc47...00000
decoded input	{ "string newName": "Alice" }
decoded output	{}
logs	[]
value	0 wei

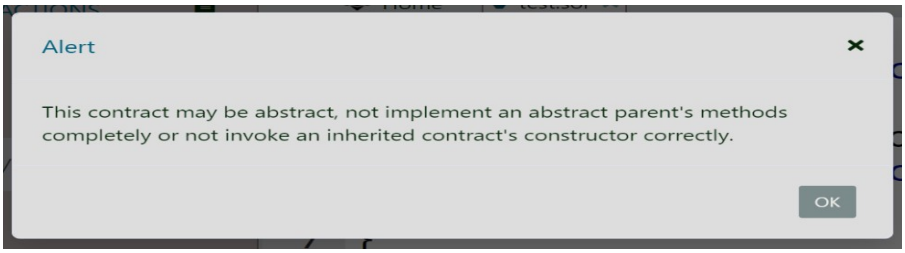
شکل ۳-۱۰: کنسول Remix ویژگی‌های مختلفی را نشان می‌دهد

```
1  pragma solidity ^0.4.0;
2  contract MyFirstContract {
3      string private name;
4      uint private age;
5      function setName(string newName)  {
6          name = newName;
7      }
8      function getName()  view returns (string) {
9          return name;
10     }
11     function setAge(uint newAge)  {
12         age = newAge;
13     }
14     function getAge() view returns (uint) {
15         return age;
16     }
17 }
```

شکل ۳-۱۱: یک قرارداد ساده در Remix

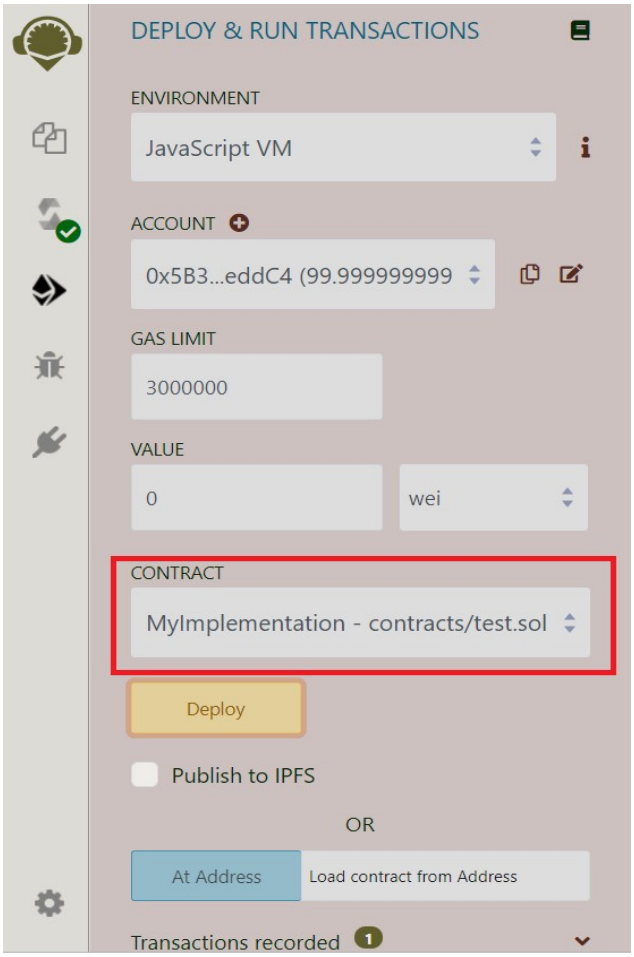


شکل ۳-۱۲: استقرار و آزمایش در Remix

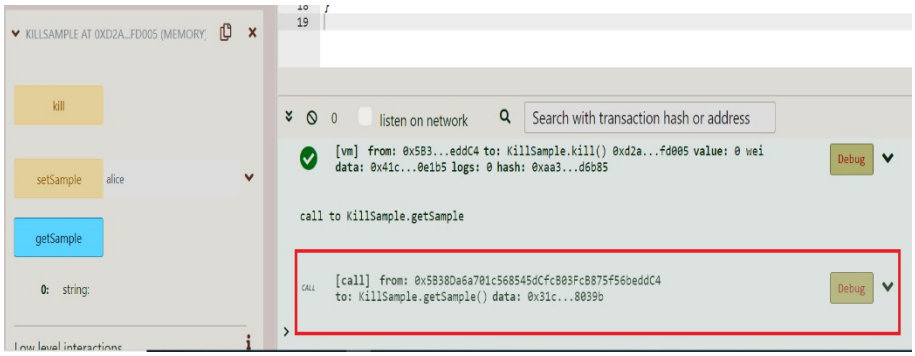


شکل ۳-۱۳: نمایش خطا برای توابع اجرا نشده

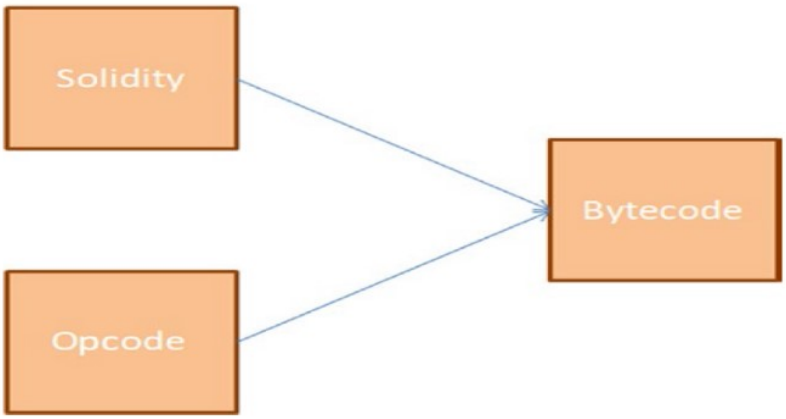
توجه داشته باشید که هنگام اجرا، باید قرارداد پیاده‌ساز را انتخاب کنید زیرا قرارداد انتزاعی قابل اجرا نیست (شکل ۳-۱٤ را ببینید).



شکل ۳-۱٤: در Remix، قرارداد پیاده‌ساز را انتخاب کنید.



شکل ۳-۱۵: Remix خطایی را نشان می‌دهد زیرا قرارداد قبلاً تخریب شده است



شکل ۳-۱۶: رابطه‌ی بین Solidity, opcode, and bytecode

```
WEB3DEPLOY

var proposalNames = /* var of type bytes32[] here */ ;
var ballotContract = new web3.eth.Contract([{"inputs":[{"internalType":"bytes32[]","n
var ballot = ballotContract.deploy({
     data: '0x6080604052348015620000115760008 0fd5b50604051620014b7380380620014b783398
     arguments: [
          proposalNames,
     ]
}).send({
     from: web3.eth.accounts[0],
     gas: '4700000'
   }, function (e, contract){
    console.log(e, contract);
    if (typeof contract.address !== 'undefined') {
         console.log('Contract mined! address: ' + contract.address + ' transactionHa
    }
 })
```

شکل ۳-۱۷: Bytecode در Remix

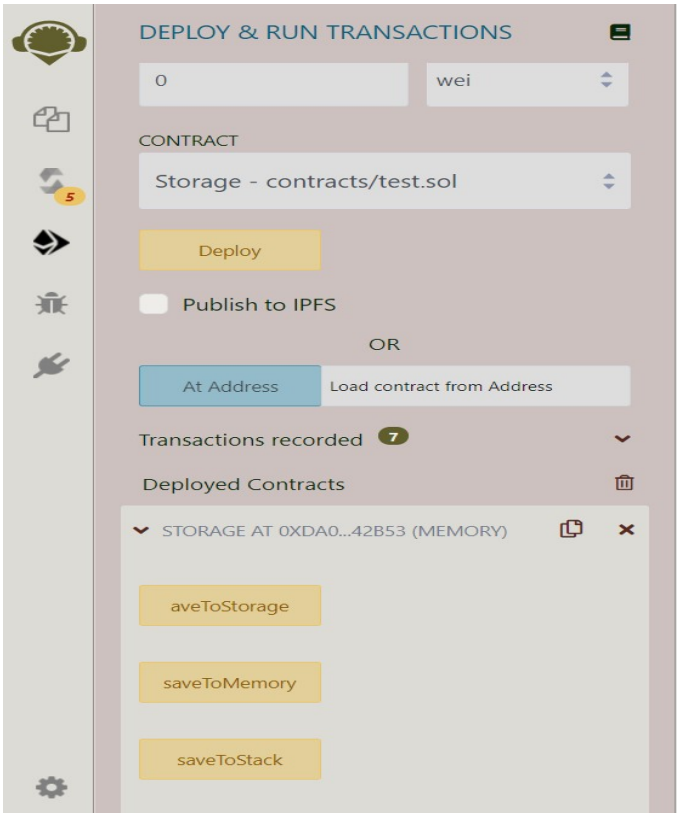
```
Ballot

USERDOC

{ "kind": "user", "methods": {}, "version": 1 }

RUNTIME BYTECODE

{ "generatedSources": [ { "ast": { "nodeType": "YulBlock", "src": "0:11428
```

شکل ۳-۱۸: Opcodes در Remix

لیست ۳-۲۹: Ballot.sol

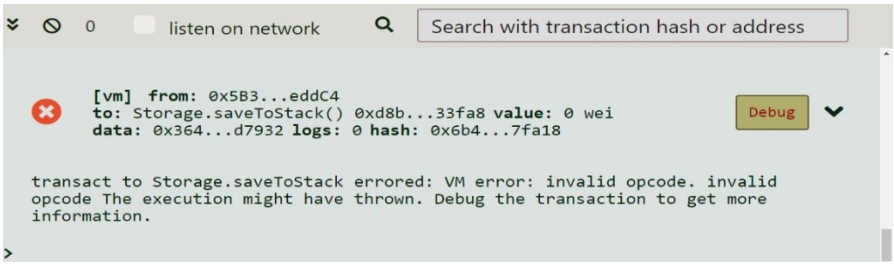
```
pragma solidity ^0.4.0;
contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }
    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;
    /// Create a new ballot with $(_numProposals) different prop
osals.
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }
    /// Give $(toVoter) the right to vote on this ballot.
    /// May only be called by $(chairperson).
    function giveRightToVote(address toVoter) public {
        if (msg.sender != chairperson || voters[toVoter].voted)
return;
        voters[toVoter].weight = 1;
    }
    /// Delegate your vote to the voter $(to).
    function delegate(address to) public {
        Voter storage sender = voters[msg.sender];  // assigns r
eference
        if (sender.voted) return;
        while (voters[to].delegate != address(0) && voters[to].d
elegate != msg.sender)
            to = voters[to].delegate;
        if (to == msg.sender) return;
        sender.voted = true;
        sender.delegate = to;
        Voter storage delegateTo = voters[to];
        if (delegateTo.voted)
            proposals[delegateTo.vote].voteCount += sender.weigh
t;
        else
            delegateTo.weight += sender.weight;
    }
    /// Give a single vote to proposal $(toProposal).
    function vote(uint8 toProposal) public {
        Voter storage sender = voters[msg.sender];
        if (sender.voted || toProposal >= proposals.length) retu
rn;
        sender.voted = true;
```

```
        sender.vote = toProposal;
        proposals[toProposal].voteCount += sender.weight;
    }
    function winningProposal() public constant returns (uint8 _w
inningProposal) {
        uint256 winningVoteCount = 0;
        for (uint8 prop = 0; prop < proposals.length; prop++)
            if (proposals[prop].voteCount > winningVoteCount) {
                winningVoteCount = proposals[prop].voteCount;
                _winningProposal = prop;
            }
    }
}
```



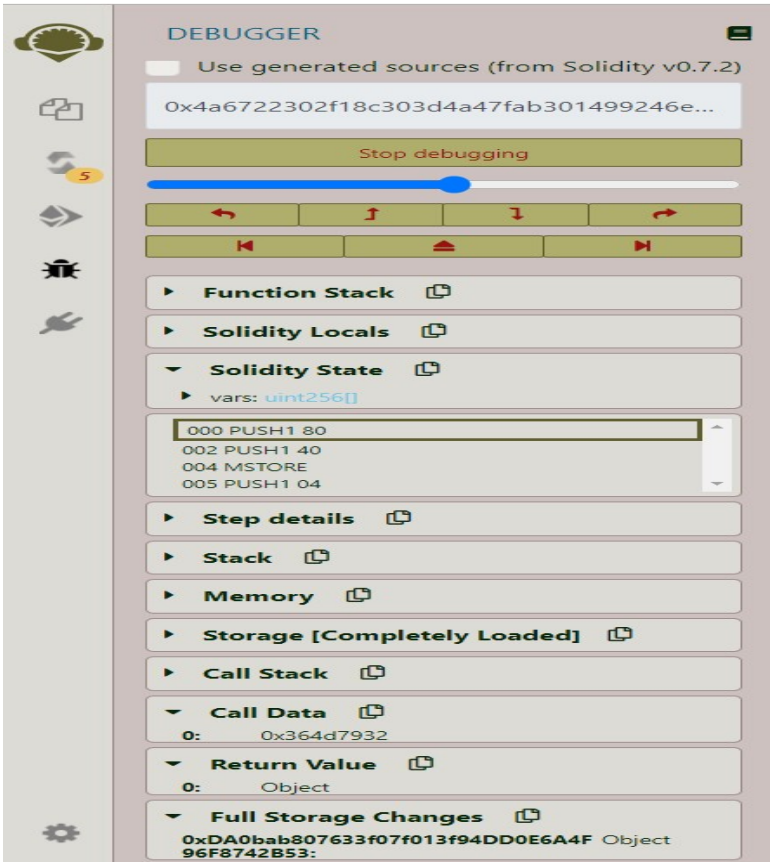
شکل۳-۱۹: استقرار Storage.sol

نام هر سه تابع در سمت راست مشاهده می‌شود. اگر روی مورد اول، saveToStack() کلیک کنیم، یک استثنا در کنسول مشاهده خواهد شد (شکل ۳-۲۰ را ببینید).



شکل ۳-۲۰: Remix console

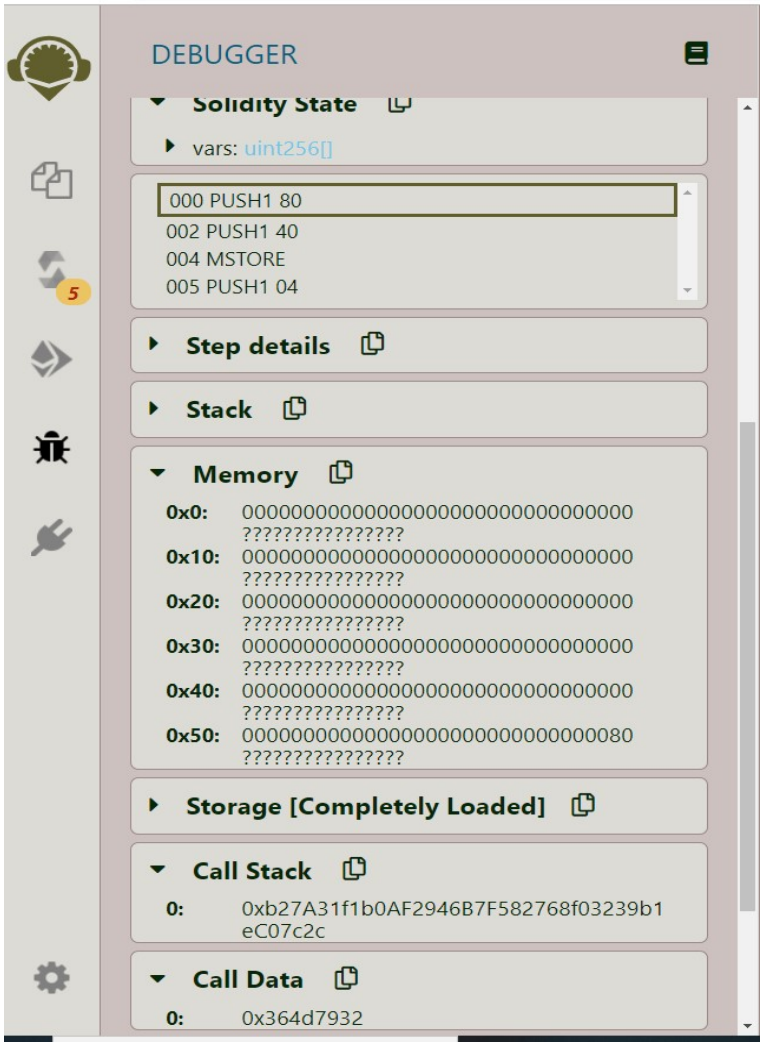
برای راه‌اندازی اشکال‌زدا[1]، روی دکمه اشکال‌زدایی کلیک شود. اکنون در سمت راست، بخش‌های مختلف مشاهده می‌گردد (شکل ۳-۲۱ را ببینید).



شکل ۳-۲۱: کامپایل فایل Storage.sol

[1] Debugger

بخش Transaction یک نوار لغزنده دارد که کمک می‌کند قسمت‌های مختلف کد را مرور کنیم. با کلیک بر روی شماره‌های خط در سمت چپ، می‌توان نقاط توقف[1] را تنظیم کرد و سپس برای مرور کد، یک گام به جلو یا عقب، حرکت کرد. طبق روشی که با آن در حال رفع اشکال هستیم، می‌توان مقادیر Stack، Memory و Storage را مشاهده کرد (شکل ۳-۲۲ را ببینید).



شکل ۳-۲۲: Remix مقادیر Stack، Memory و Storage را در Storage.sol نشان می‌دهد

[1] Break points

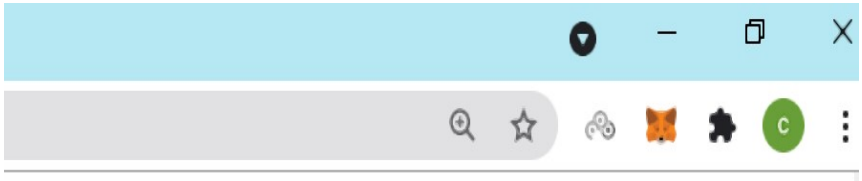
فصل چهارم

استقرار قراردادهای هوشمند

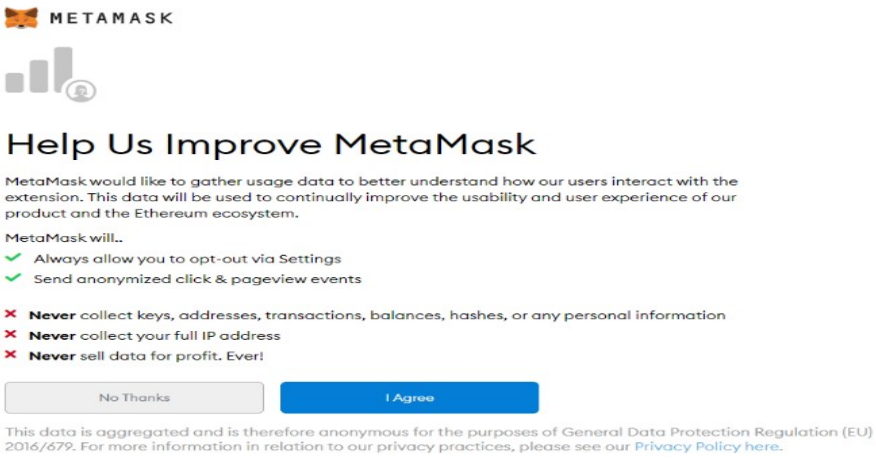
```
ganache-cli
```

```
Administrator: Command Prompt - ganache-cli
6.14.12

C:\Windows\system32>npm install -g ganache-cli
C:\Users\np\AppData\Roaming\npm\ganache-cli -> C:\Users\np\AppData\Roaming\npm\node_modules\ganache-cli\cli.js

> keccak@3.0.1 install C:\Users\np\AppData\Roaming\npm\node_modules\ganache-cli\node_modules\keccak
> node-gyp-build || exit 0


> secp256k1@4.0.2 install C:\Users\np\AppData\Roaming\npm\node_modules\ganache-cli\node_modules\secp256k1
> node-gyp-build || exit 0

+ ganache-cli@6.12.2
added 101 packages from 182 contributors in 9.922s

C:\Windows\system32>ganache-cli
Ganache CLI v6.12.2 (ganache-core: 2.13.2)

Available Accounts
==================
(0) 0x061e0f76aeFe8652C02B32a9Bf2aCFf6ccd01bD9 (100 ETH)
(1) 0xF2eECBE3bf76fb4D1f1B68b6BAF264f0Cd40Abe8 (100 ETH)
(2) 0xdC32d069FB85874259AFf769EC8759E0AeE0E391 (100 ETH)
(3) 0x965c71Ed50B1C60893C21D814FE59453795b8fE6 (100 ETH)
(4) 0x2aE7ce6FF3eF134812FBc7708D982433b78C60cB (100 ETH)
(5) 0x14F339B2F1e269CA78D90EED988229618A76Cc68 (100 ETH)
(6) 0xdE93D442095bf879c2Da9538eD0a1c62b980F14F (100 ETH)
(7) 0xdC07EE6E81D17B9f6Ee501A8B4C4a1dF5C4b3092 (100 ETH)
(8) 0xef8E9440FABa9DfA031010D42ef8386DA16cBB35 (100 ETH)
(9) 0xc04A3f93B8BfB4233cbc336D1F10b0a8E1F610E7 (100 ETH)

Private Keys
==================
(0) 0x998f1d1da8d3825f50a9babc185b40bc543b8d1e729354916ee63acb1888dd9f
(1) 0x68fd19ce41ffcf173bd210c40e162b39e53a7d2612da11accf4fd19010f5ef59
(2) 0xaabef4c6c85d345216f1a612547860c46c62b2d2c642c56f9c350700d322d8fb
(3) 0xd28b3f66a1f742bc57435aa75cb41f7cc3c827de20eaa07c52d743189480f274
(4) 0x6ad1a3aecc3ffb964980caa8e5c31cd30080237fdc5d96ae730a4eb4d9076aea
(5) 0x7ce4e3face3926fb359cd5efc2cede9e7a1576bd908a340ae3058a670a0be19e
(6) 0x17c9667e68b1a6343e8dfeea1269978f4ec43382f144bcd492a17590eaed794b
(7) 0x08b25ff1388acb1dc051e5d151e907a78bee8febd08a2088940a05da4e070712
(8) 0x093bb6fe862c41aba16d21428ea117413ae41ce09dafec8809ef7bdd2cd3c2ba
(9) 0xc133f0eaa7148eca154c140017f93962df476143b98fcf0a4a053f15346cf703

HD Wallet
==================
Mnemonic:      mixed swamp apology movie tuition lonely weapon joy sight fan super come
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
>
```

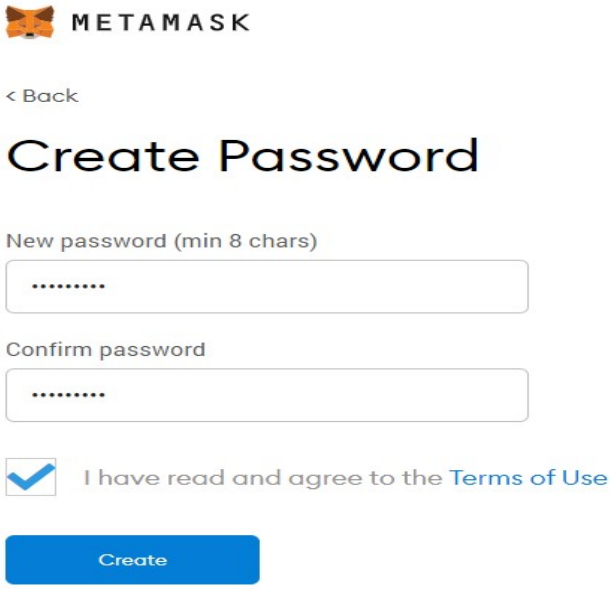
شکل ٤-١: اجرای ganache-cli



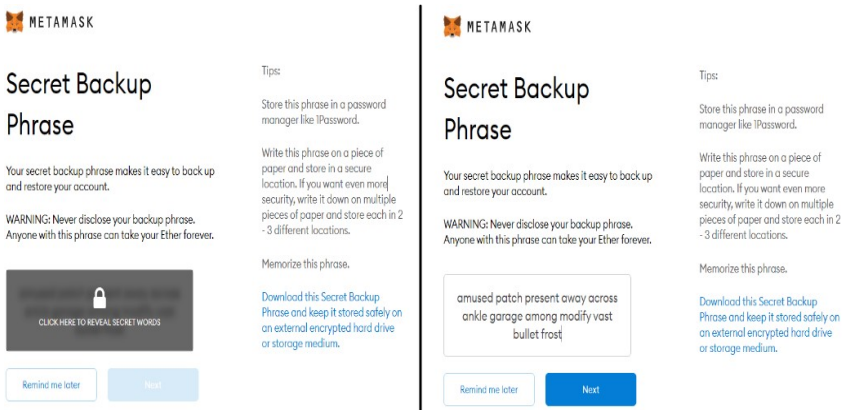
شکل ٤-٢ MetaMask در مرورگر Chrome



شکل ٤-٣: MetaMask pop-up برای شرایط و ضوابط



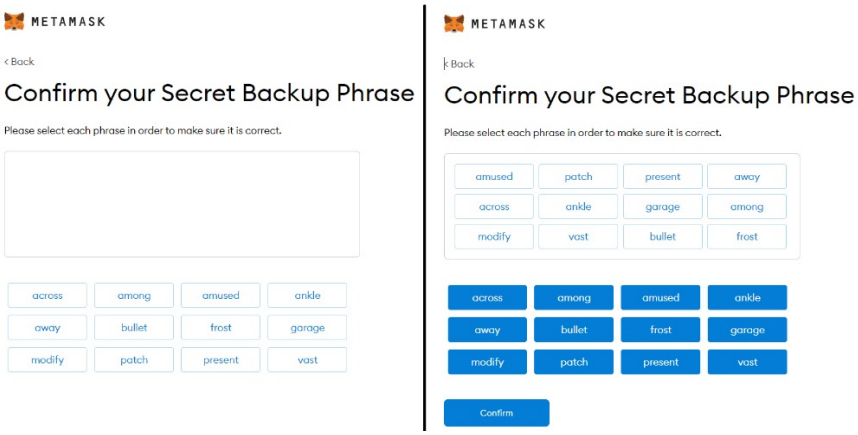
شکل ٤-٤: انتخاب گذرواژه‌ی ورود برای کیف پول MetaMask



(الف) (ب)

شکل ٤-٥: (الف) عبارات بازیابی رمزعبور و (ب) وارد کردن عبارات بازیابی رمزعبور

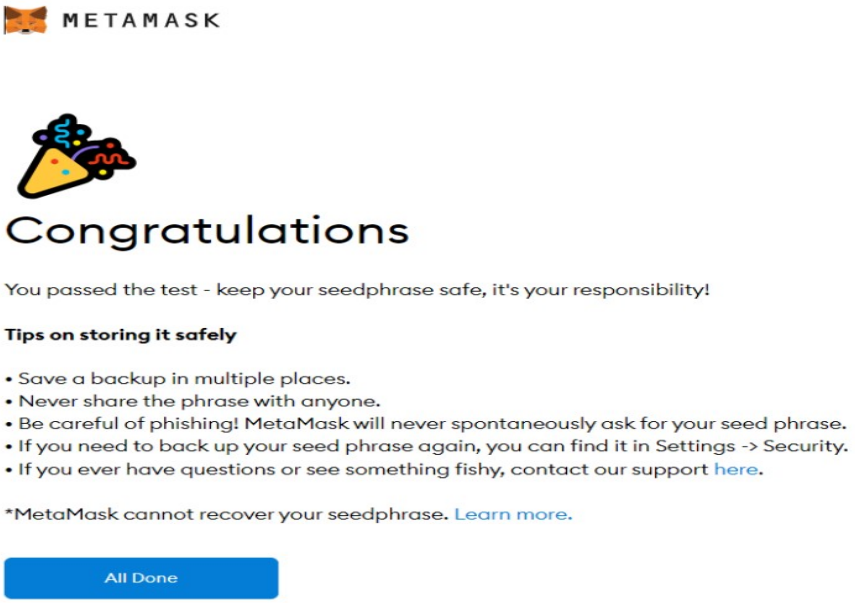
در کادر نشان داده شده عباراتِ بازیابیِ رمز را «به‌ترتیب» انتخاب و سپس روی confirm کلیک کنید (شکل ٤-٦ را ببینید).



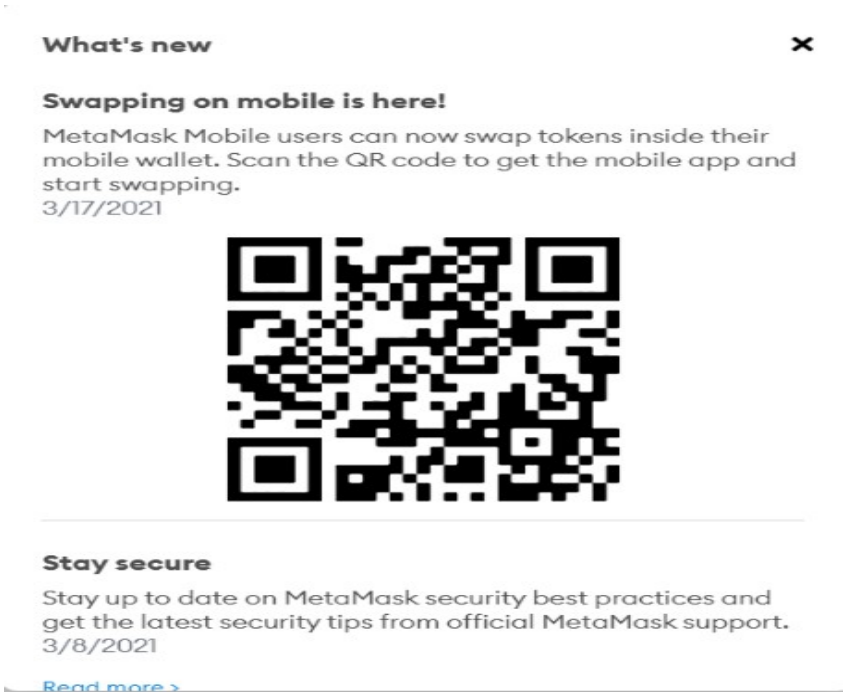
(الف) (ب)

شکل ٤-٦: تایید ترتیب کلمات بازیابی رمزعبور

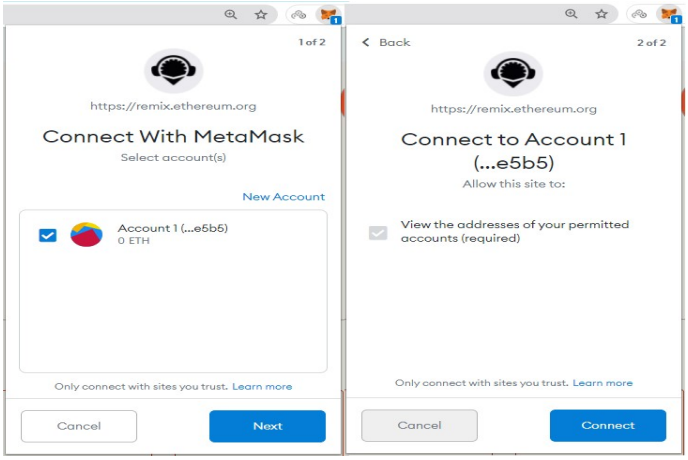
کیف پول شما ایجاد شد (شکل ٤-٧).



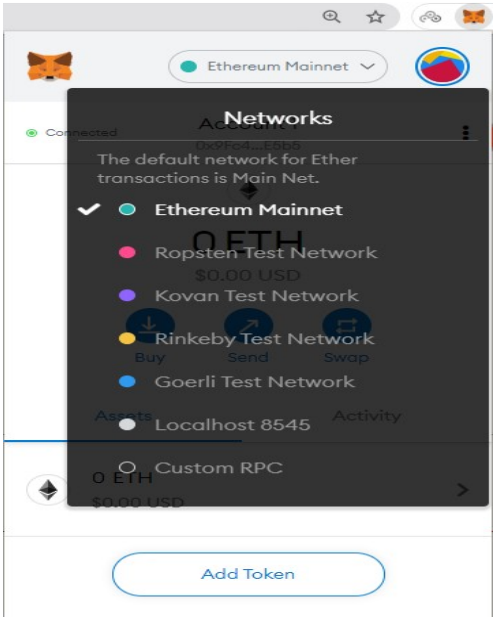
شکل ٤-٧: نصب کیف پول



شکل ٤-٨: اتصال MetaMask به تلفن همراه



شکل ٤-٩: MetaMask pop-up با اتصال شبکه‌ی اصلی پیش‌فرض



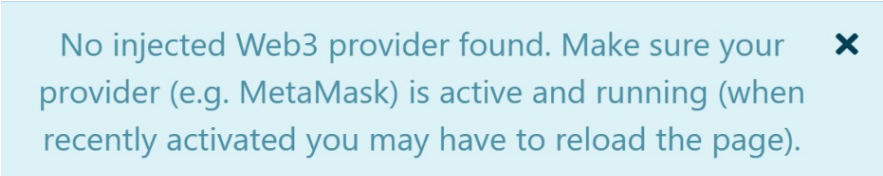
شکل ٤-١٠: انتخاب شبکه آزمایش Ropsten

استقرار قرارداد

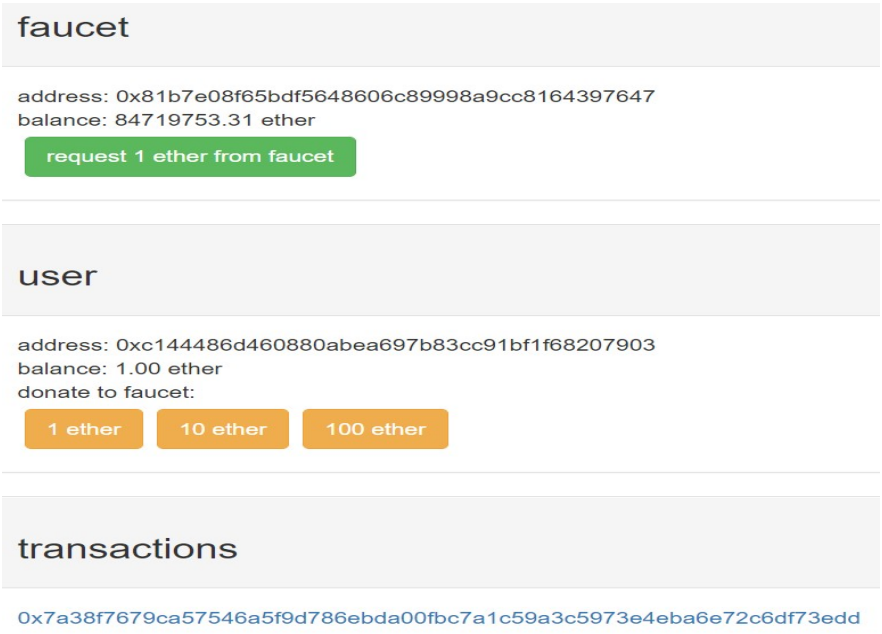
اکنون به مرورگر Remix خود بازگشته و همانند لیست ٤-٣، یک قرارداد ساده بنویسید.

لیست ٤-٣: یک قرارداد ساده

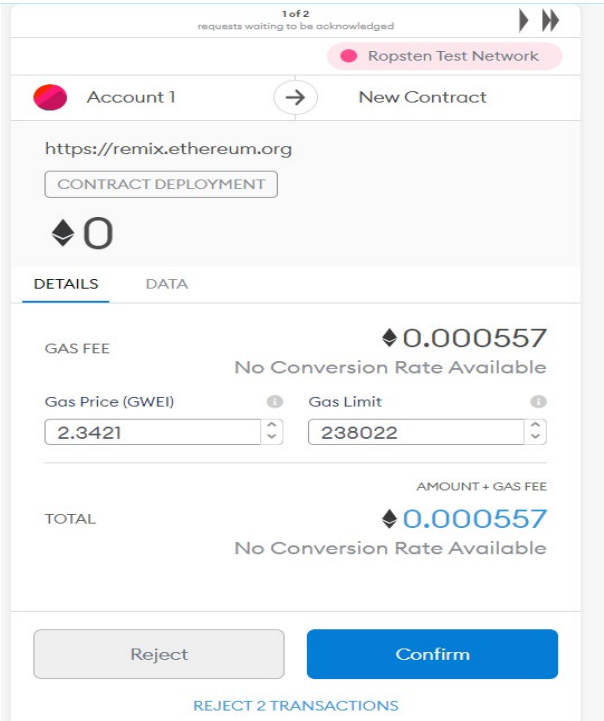
```
pragma solidity ^0.4.0;
contract MyFirstContract {
    string private name;
    uint private age;
    function setName(string newName) public {
        name = newName;
    }
    function getName() public view returns (string) {
        return name;
    }
    function setAge(uint newAge) public {
        age = newAge;
    }
    function getAge() public view returns (uint) {
        return age;
    }
}
```



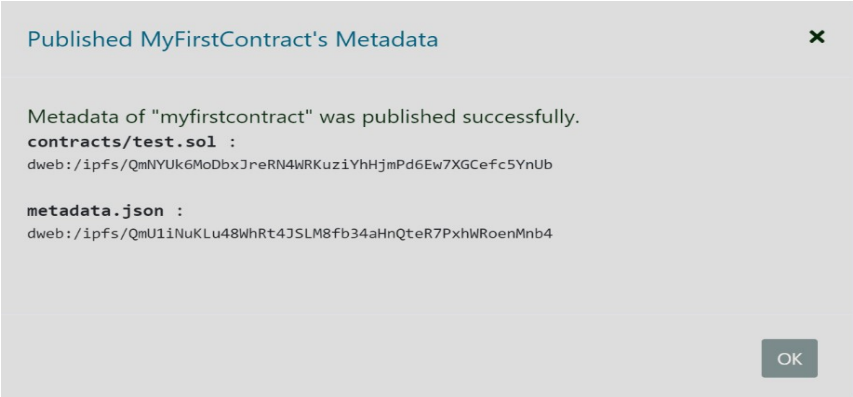
شکل ٤-١١: پیغام هشدار در صورت عدم اجرای MetaMask



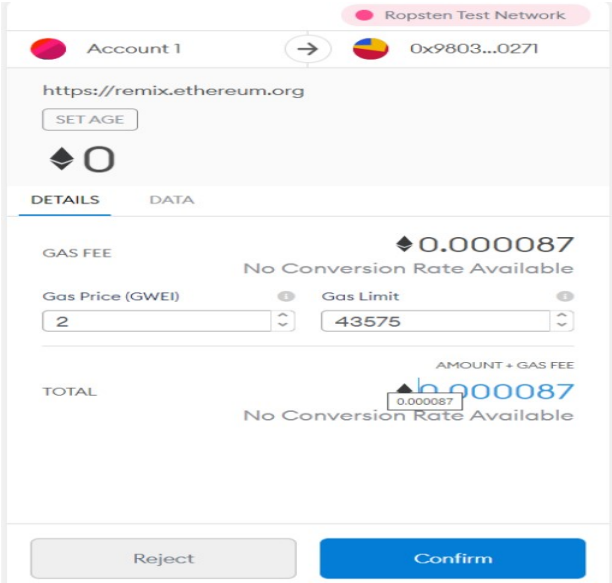
شکل ٤-١٢: وب‌سایت https://faucet.metamask.io/



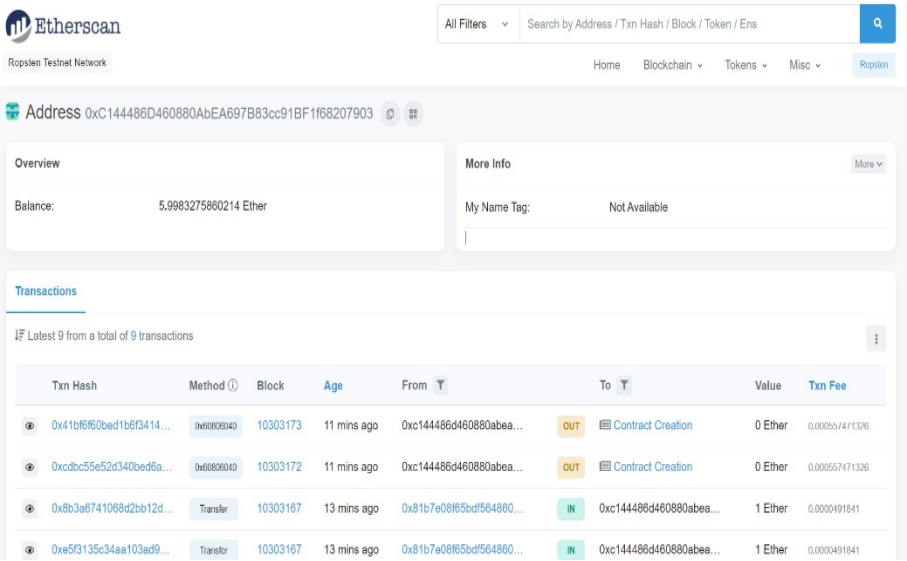
شکل ٤-١٣: بر روی دکمه‌ی ارسال در MetaMask کلیک شود



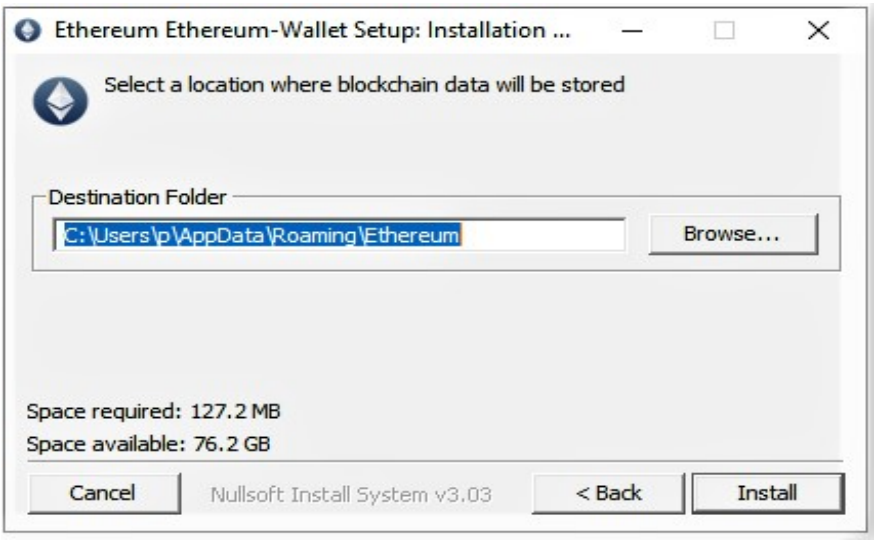
شکل ٤-١٤: استقرار قرارداد



شکل ٤-١٥: اتر در Remix و MetaMask کسر می‌شود



شکل ٤-١٦: گزارش Etherscan



شکل ٤-١٧: نصب کیف پول اتریوم

```
Command Prompt - geth
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\p>geth
INFO [05-25|17:18:09.754] Starting Geth on Ethereum mainnet...
INFO [05-25|17:18:09.758] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [05-25|17:18:09.764] Maximum peer count                       ETH=50 LES=0 total=50
WARN [05-25|17:18:09.771] Sanitizing cache to Go's GC limits       provided=4096 updated=2694
INFO [05-25|17:18:09.794] Set global gas cap                       cap=25,000,000
INFO [05-25|17:18:09.809] Allocated trie memory caches             clean=404.00MiB dirty=673.00MiB
INFO [05-25|17:18:09.825] Allocated cache and file handles         database=C:\Users\p\AppData\Local\Ethereum\geth\chaindata cache=1.31GiB handles=8192
INFO [05-25|17:18:11.116] Opened ancient database                  database=C:\Users\p\AppData\Local\Ethereum\geth\chaindata\ancient readonly=false
INFO [05-25|17:18:11.470] Initialised chain configuration          config="{ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000
: 7280000 Istanbul: 9069000, Muir Glacier: 9200000, Berlin: 12244000, YOLO v3: <nil>, Engine: ethash}"
INFO [05-25|17:18:11.520] Disk storage enabled for ethash caches   dir=C:\Users\p\AppData\Local\Ethereum\geth\ethash count=3
INFO [05-25|17:18:11.525] Disk storage enabled for ethash DAGs     dir=C:\Users\p\AppData\Local\Ethash count=2
INFO [05-25|17:18:11.541] Initialising Ethereum protocol           network=1 dbversion=8
INFO [05-25|17:18:11.861] Loaded most recent local header          number=1,827,244 hash=97fdd6..7dcde9 td=34,035,961,766,293,196,064 age=4y11mo2w
INFO [05-25|17:18:11.867] Loaded most recent local full block      number=0        hash=d4e567..cb8fa3 td=17,179,869,184             age=52y1mo3w
INFO [05-25|17:18:11.872] Loaded most recent local fast block      number=1,823,735 hash=729be7..82da1c td=33,829,036,140,180,720,146 age=4y11mo2w
INFO [05-25|17:18:11.986] Loaded last fast-sync pivot marker       number=12,503,497
INFO [05-25|17:18:12.041] Upgrading chain index                    type=bloombits percentage=30
INFO [05-25|17:18:12.042] Loaded local transaction journal         transactions=0 dropped=0
INFO [05-25|17:18:12.057] Regenerated local transaction journal    transactions=0 accounts=0
INFO [05-25|17:18:12.103] Allocated fast sync bloom                size=1.31GiB
WARN [05-25|17:18:12.167] Unclean shutdown detected                booted=2021-05-11T10:28:06+0430 age=2w6h50m
WARN [05-25|17:18:12.172] Unclean shutdown detected                booted=2021-05-25T17:15:08+0430 age=3m4s
INFO [05-25|17:18:12.176] Starting peer-to-peer node               instance=Geth/v1.10.3-stable-991384a7/windows-amd64/go1.16.3
INFO [05-25|17:18:12.433] New local node record                    seq=7 id=16631aeb24251218 ip=127.0.0.1 udp=30303 tcp=30303
INFO [05-25|17:18:12.445] Started P2P networking                   self=enode://360926fb8b4fac6cd206dd0fb020bef39ae708990c6e81b3cfd247541ba20776fed33e93
INFO [05-25|17:18:12.446] IPC endpoint opened                      url=\\.\pipe\geth.ipc
INFO [05-25|17:18:15.398] Mapped network port                      proto=tcp extport=30303 intport=30303 interface="UPNP IGDv1-IP1"
INFO [05-25|17:18:15.869] Mapped network port                      proto=udp extport=30303 intport=30303 interface="UPNP IGDv1-IP1"
INFO [05-25|17:18:16.013] New local node record                    seq=8 id=16631aeb24251218 ip=100.67.28.73 udp=30303 tcp=30303
INFO [05-25|17:18:20.111] Initializing state bloom                 items=1,035,498 errorrate=0.000 elapsed=8.004s
INFO [05-25|17:18:20.248] Upgrading chain index                    type=bloombits percentage=34
INFO [05-25|17:18:22.973] Looking for peers                        peercount=0 tried=11 static=0
INFO [05-25|17:18:28.115] Initializing state bloom                 items=1,413,503 errorrate=0.000 elapsed=16.008s
INFO [05-25|17:18:28.408] Upgrading chain index                    type=bloombits percentage=39
INFO [05-25|17:18:30.936] Block synchronisation started
WARN [05-25|17:18:30.937] Synchronisation failed, retrying         err="peer is unknown or unhealthy"
INFO [05-25|17:18:33.053] Looking for peers                        peercount=0 tried=20 static=0
```

شکل ٤-١٨: اجرای geth روی خط فرمان

همان‌طور که در شکل ٤-١٩ نشان داده شده است، Chaindata را به مکان پیش‌فرض شما که Geth نصب شده بارگیری می‌کند.

```
INFO [05-25|17:18:09.794] Set global gas cap                       cap=25,000,000
INFO [05-25|17:18:09.809] Allocated trie memory caches             clean=404.00MiB dirty=673.00MiB
INFO [05-25|17:18:09.825] Allocated cache and file handles         database=C:\Users\p\AppData\Local\Ethereum\geth\chaindata cache=1.31GiB handles=8192
INFO [05-25|17:18:11.116] Opened ancient database                  database=C:\Users\p\AppData\Local\Ethereum\geth\chaindata\ancient readonly=false
INFO [05-25|17:18:11.470] Initialised chain configuration          config="{ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000
```

شکل ٤-١٩: Chaindata در مکان پیش‌فرض بارگیری می‌شود

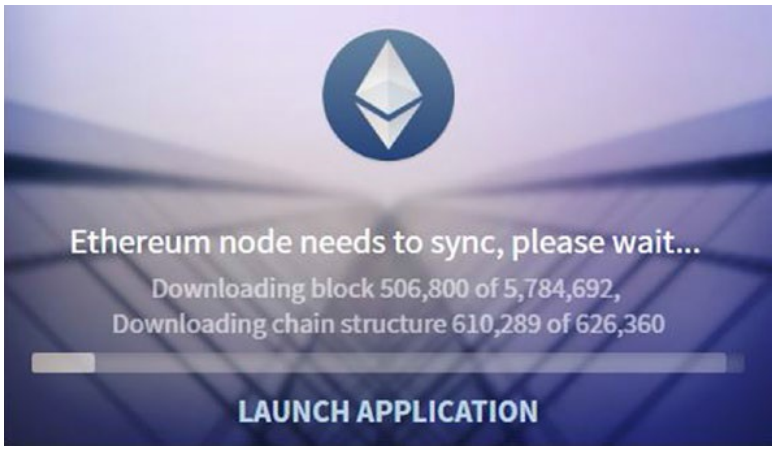
لیست ٤-٤: Genesis.json

```
{
    "config": {
        "chainId": 19763,
        "homesteadBlock": 0,
        "eip150Block": 0,
        "eip155Block": 0,
```

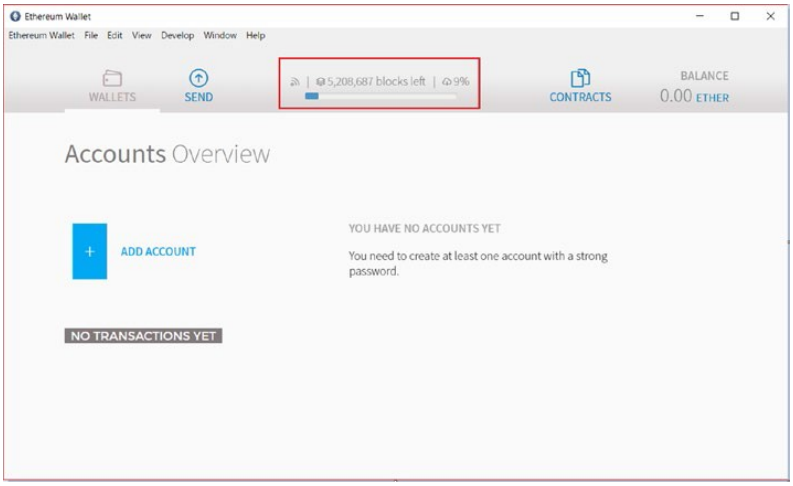
```
            "eip158Block": 0,
            "byzantiumBlock": 0,
            "ethash": {}
        },
        "nonce": "0xdeadbeefdeadbeef",
        "timestamp": "0x0",
        "extraData":
                                "0x0000000000000000000000000000000000000000000000000000000000000000000000000",
        "gasLimit": "0x80000000",
        "difficulty": "0x20000",
        "mixHash":
                                "0x0000000000000000000000000000000000000000000000000000000000000000000000000",
        "coinbase":
                                "0x0000000000000000000000000000000000000000",
        "alloc": {
            "71562b71999873db5b286df957af199ec94617f7": {
                "balance": "0xffffffffffffffffffffffffff"
            }
        },
        "number": "0x0",
        "gasUsed": "0x0",
        "parentHash":
                                "0x0000000000000000000000000000000000000000000000000000000000000000000000000"
    }
```

```
C:\Users\p>geth -datadir=./chaindata
INFO [05-26|11:59:27.040] Starting Geth on Ethereum mainnet...
INFO [05-26|11:59:27.070] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [05-26|11:59:27.081] Maximum peer count                       ETH=50 LES=0 total=50
WARN [05-26|11:59:27.138] Sanitizing cache to Go's GC limits       provided=4096 updated=2694
INFO [05-26|11:59:27.157] Set global gas cap                       cap=25,000,000
INFO [05-26|11:59:27.164] Allocated trie memory caches             clean=404.00MiB dirty=673.00MiB
WARN [05-26|11:59:27.201] Using deprecated resource file C:\Users\p\chaindata\chaindata, please move this file to the 'geth' subdirectory of datadir.
INFO [05-26|11:59:27.212] Allocated cache and file handles         database=C:\Users\p\chaindata\chaindata cache=1.31GiB handles=8192
INFO [05-26|11:59:27.577] Opened ancient database                  database=C:\Users\p\chaindata\chaindata\ancient readonly=false
INFO [05-26|11:59:27.610] Writing default main-net genesis block
INFO [05-26|11:59:27.821] Persisted trie from memory database      nodes=12356 size=1.78MiB time=42.8382ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-26|11:59:27.833] Initialised chain configuration          config="{ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000 EIP155: 2675000 EIP158:
: 7280000 Istanbul: 9069000, Muir Glacier: 9200000, Berlin: 12244000, YOLO v3: <nil>, Engine: ethash}"
INFO [05-26|11:59:27.841] Disk storage enabled for ethash caches   dir=C:\Users\p\chaindata\geth\ethash count=3
INFO [05-26|11:59:27.845] Disk storage enabled for ethash DAGs     dir=C:\Users\p\AppData\Local\Ethash  count=2
INFO [05-26|11:59:27.847] Initialising Ethereum protocol           network=1 dbversion=<nil>
INFO [05-26|11:59:27.850] Loaded most recent local header          number=0 hash=d4e567..cb8fa3 td=17,179,869,184 age=52y1mo3w
```

شکل ٤-٢٠: اتریوم روی یک شبکه خصوصی

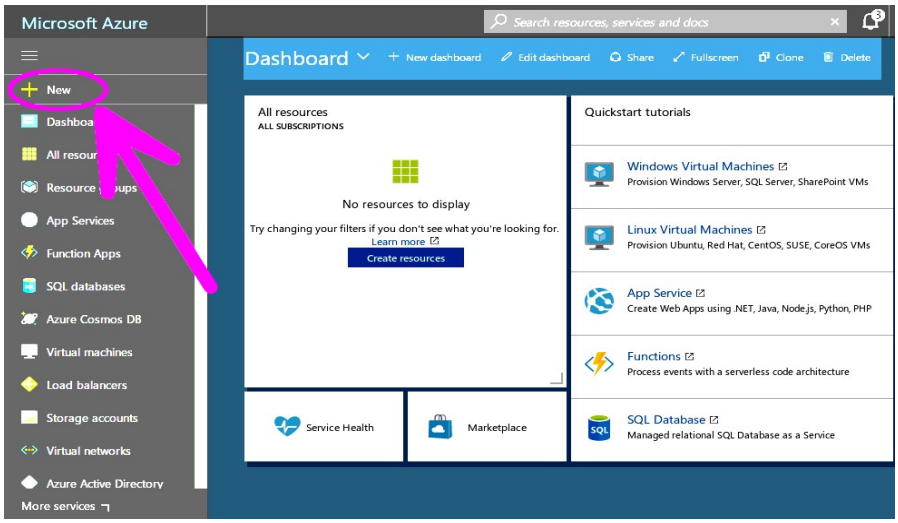


شکل ٤-٢١: شروع Mist



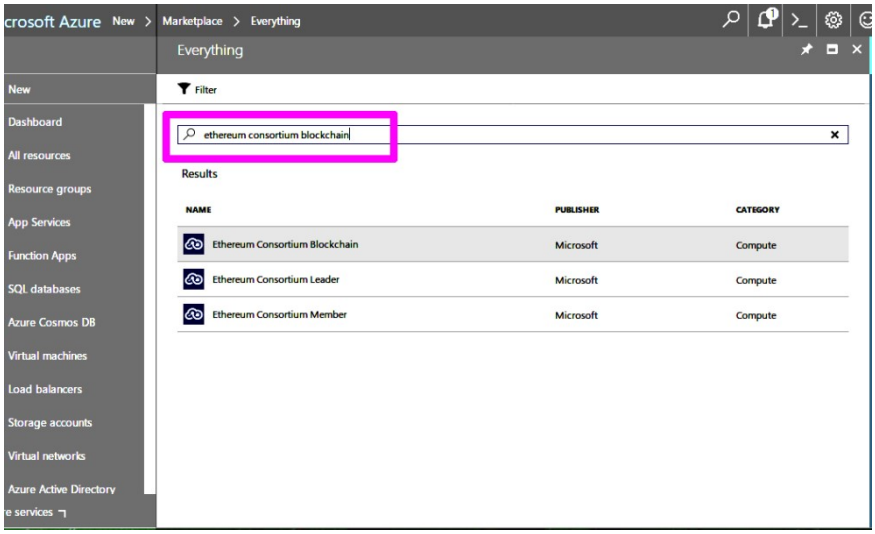
شکل ٤-٢٢: راه‌اندازی برنامه

شکل ٤-٢٣: دستورالعمل‌های بارگیری فایل‌های کلیدی



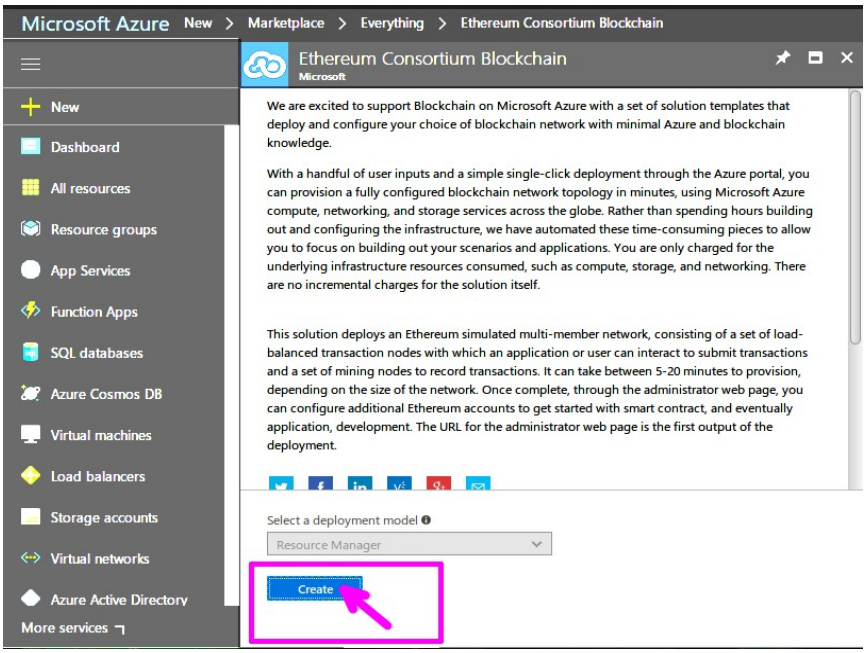
شکل ٤-٢٤: پرتال Azure

۱.در نوار جستجوی منابع، زنجیره‌ی بلوکیِ کنسرسیومِ اتریوم را وارد کنید. ethereum consortium را انتخاب کنید، سپس روی "Create" کلیک کنید.

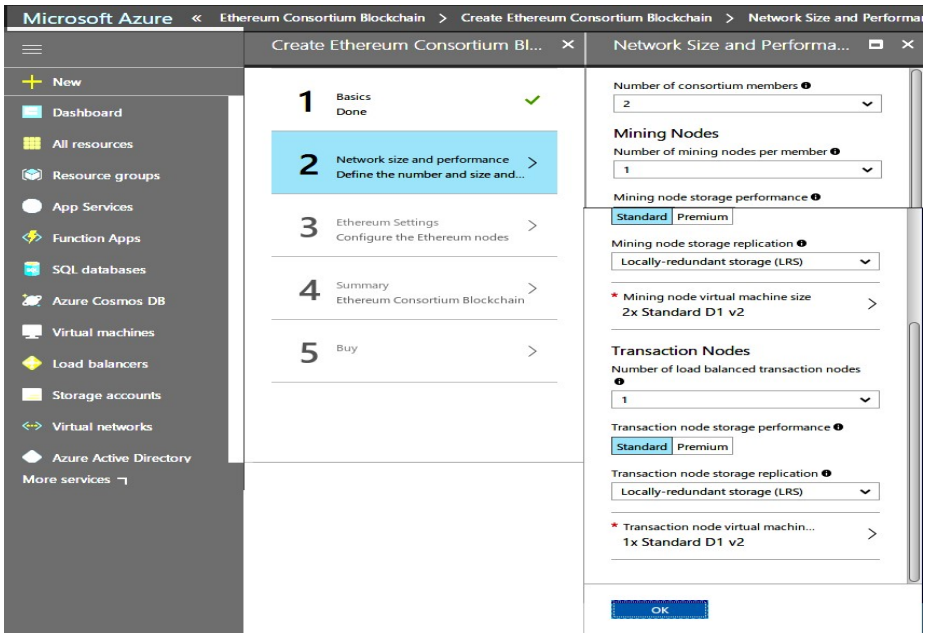


شکل ٤-٢٥: کنسرسیوم اتریوم

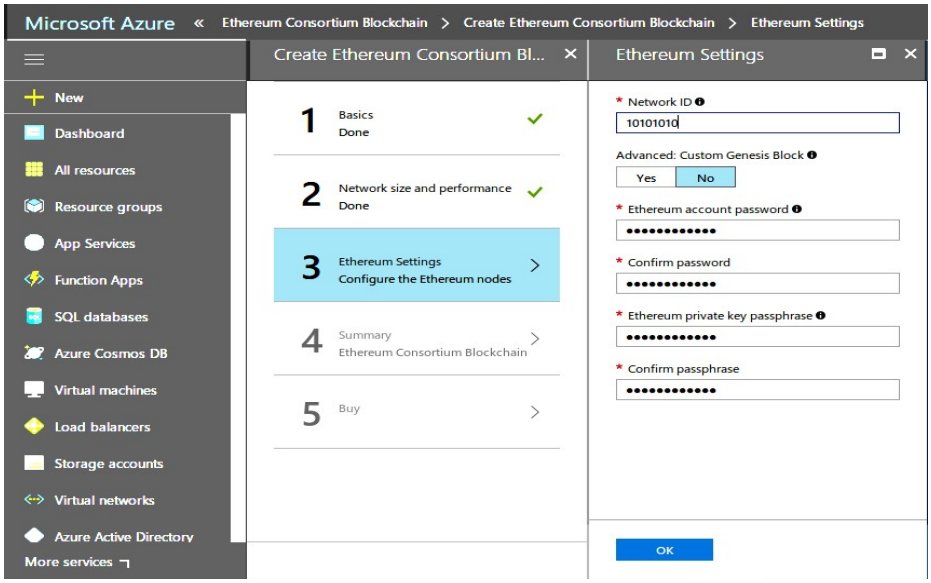
۲. برای تأیید مدل استقرار ، روی create کلیک کنید.



شکل ٤-٢٦: تایید مدل استقرار

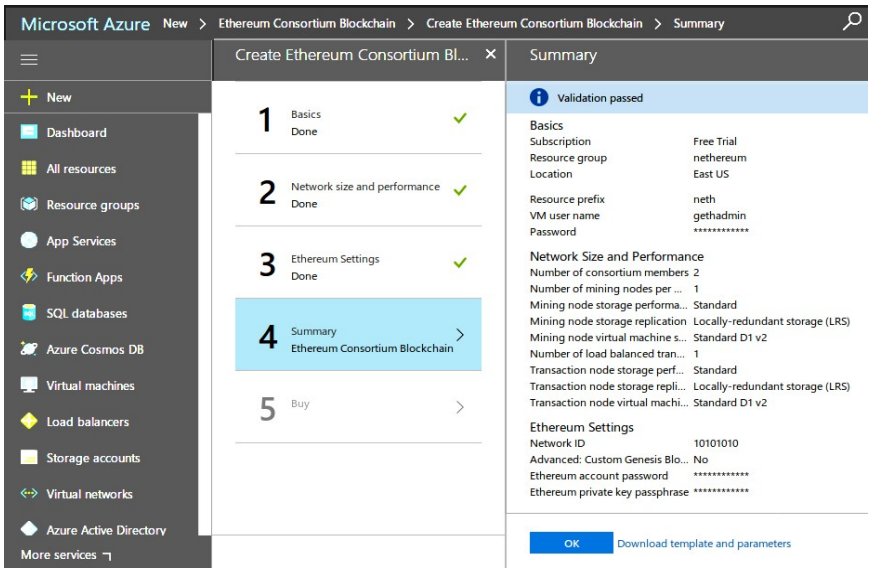


شکل ٤-٢٧: تنظیمات اتریوم

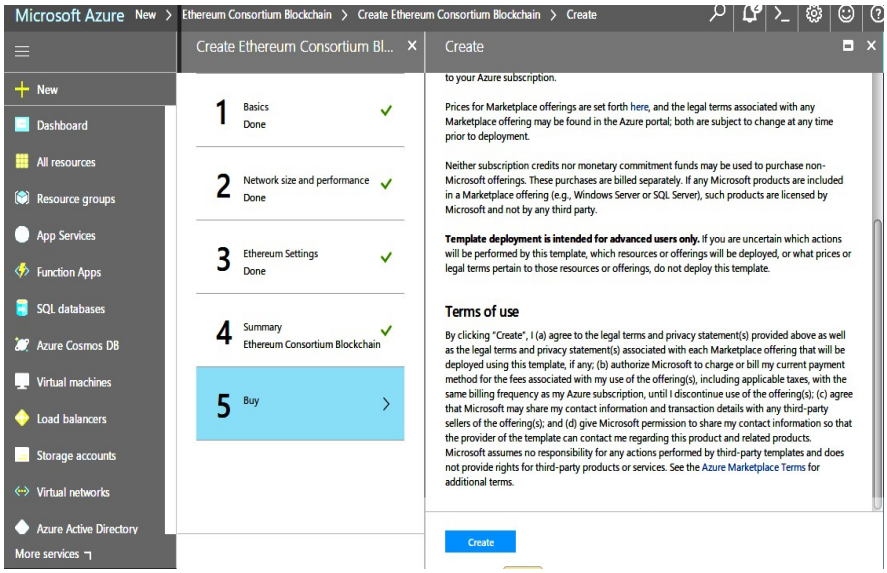


شکل ٤-٢٨: انتخاب شناسه‌ی شبکه و گذروازه

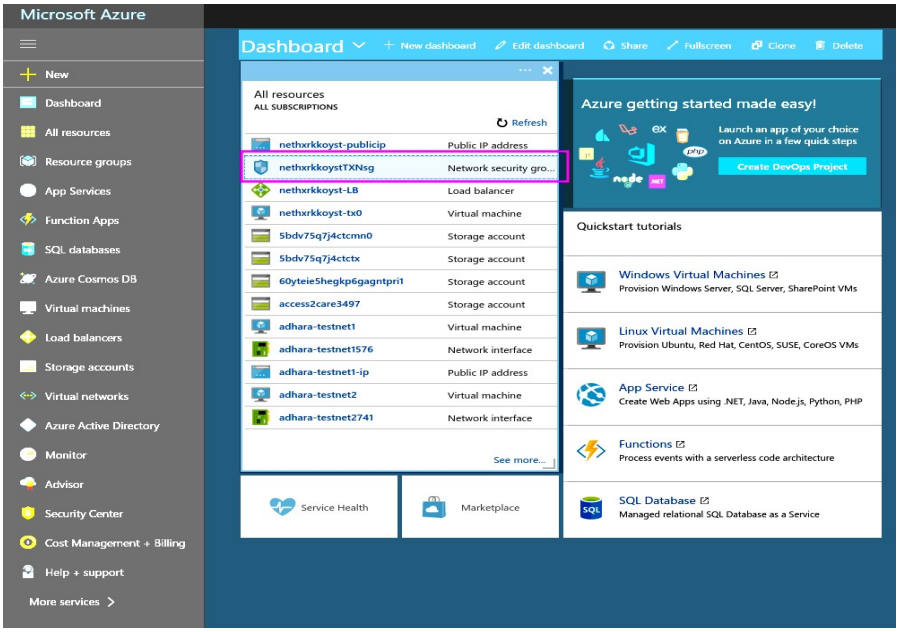
.١



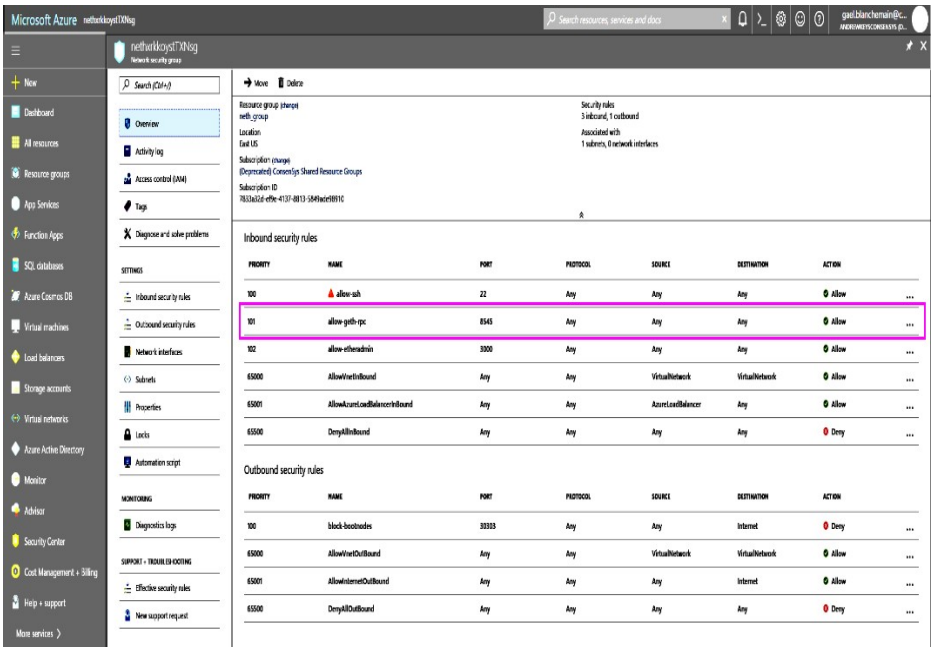
شکل ٤-٢٩: مشخصات زنجیره‌ی بلوکی



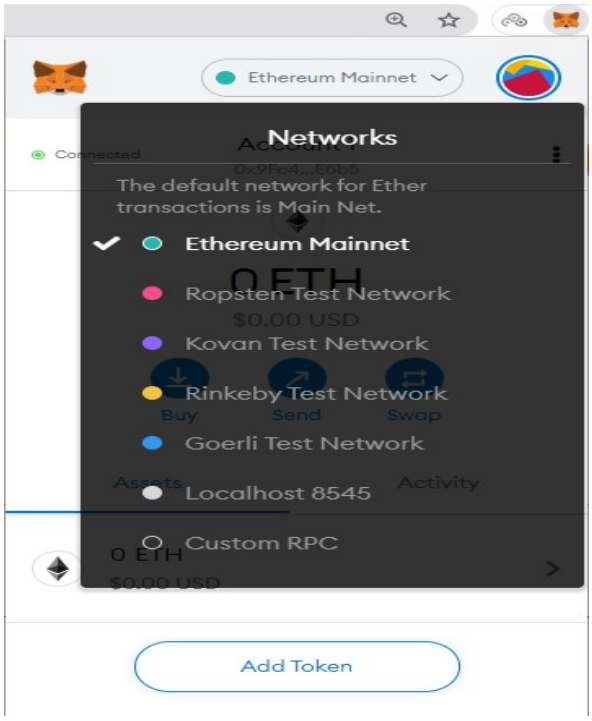
شکل ٤-٣٠: شرایط استفاده از Azure



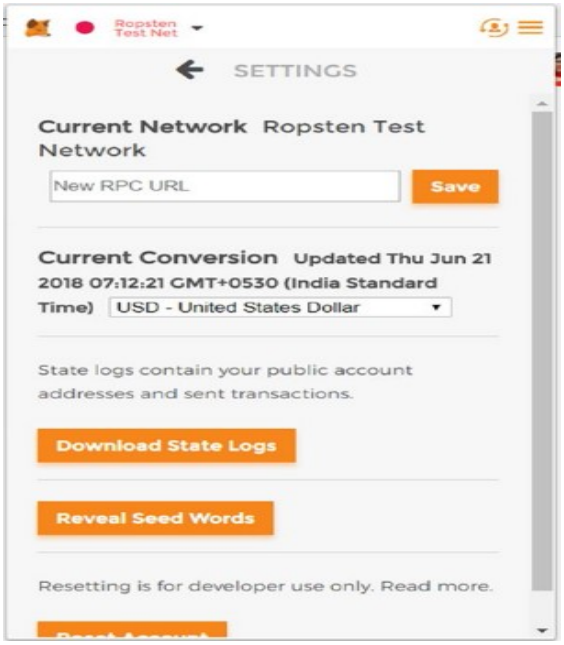
شکل ٤-٣٢: انتخاب Network Security Group



شکل ٤-٣٣: انتخاب پورت مورد استفاده برای تراکنش



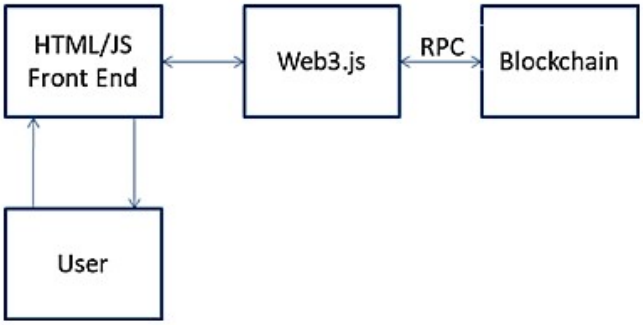
شکل ٤-٣٤: RPC سفارشی در MetaMask


Ropsten Test Net
SETTINGS
Current Network Ropsten Test Network
New RPC URL
Save
Current Conversion Updated Thu Jun 21 2018 07:12:21 GMT+0530 (India Standard Time)
USD - United States Dollar
State logs contain your public account addresses and sent transactions.
Download State Logs
Reveal Seed Words
Resetting is for developer use only. Read more.

شکل ٤-٣٥: URL

فصل پنجم

ادغام با رابط کاربری



شکل ۵-۱: Web3.js

```
root@blockchain-HuronRiver-Platform:/test# mkdir Register
root@blockchain-HuronRiver-Platform:/test# cd Register
root@blockchain-HuronRiver-Platform:/test/Register# npm init -y
Wrote to /test/Register/package.json:

{
  "name": "register",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

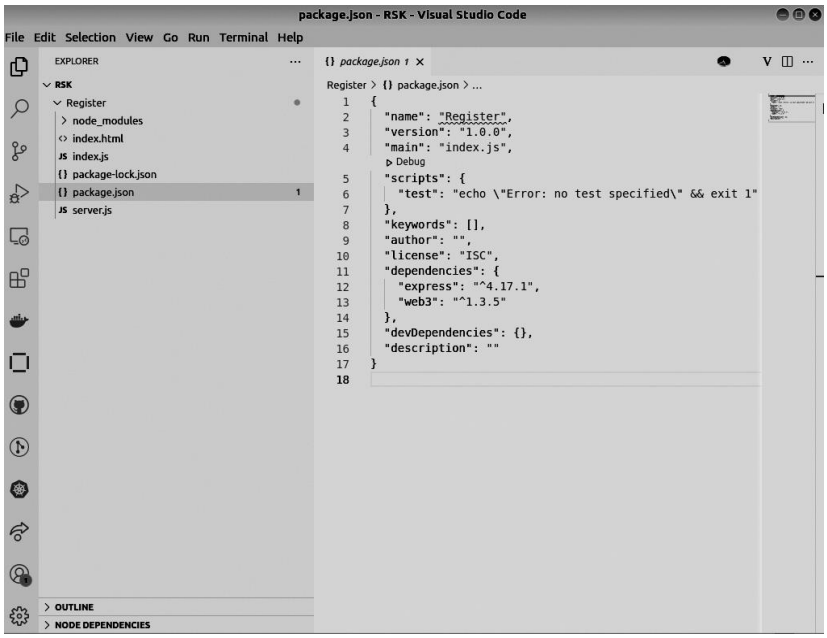
شکل ۵-۲: نصب Express

```
root@blockchain-HuronRiver-Platform:/test/Register# npm install express --save

added 50 packages, and audited 51 packages in 13s

found 0 vulnerabilities
```

شکل ۵-۳: نصب Web3.js

```
root@blockchain-HuronRiver-Platform:/test/Register# npm install web3 --save
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mk
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated ethereumjs-tx@2.1.2: New package name format for new versions: @ethereumjs
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/reques
npm WARN deprecated multicodec@0.5.7: stable api reached
npm WARN deprecated ethereumjs-common@1.5.2: New package name format for new versions: @ethere

added 310 packages, and audited 361 packages in 30s

51 packages are looking for funding
  run `npm fund` for details

21 high severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.

Run `npm audit` for details.
```

شکل ۵-٤: فایل package.json را بررسی کنید



شکل ۵-۵: بررسی وابستگی پرونده‌ها.

لیست ۵-۱: قرارداد Solidity Register.sol

```
pragma solidity ^0.4.18;
contract StudentDetails {
   string fName;
   string lName;
```

```
    string dob;
    function setStudentDetails(string _fName, string _lName,
 string _dob) public {
        fName = _fName;
        lName = _lName;
        dob = _dob;
    }
    function getStudentDetails() public constant returns
 (string, string, string) {
        return (fName, lName, dob);
    }
 }
```

لیست ۵-۲: index.html

```
<!DOCTYPE html>
<html >
  <head>
    <title>Register information at Blockchain</title>

    <link
href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/boot
strap.min.css' rel='stylesheet' type='text/css'>
    <script src="https://code.jquery.com/jquery-
3.1.1.slim.min.js"></script>
    <script
src="./node_modules/web3/dist/web3.min.js"></script>
    <script src="./index.js"></script>
  </head>

  <body class="container">

    <h1 class="page-header">Register information at Blockchain
- RSK network</h1>

    <div class="row">
      <div>
        <h3 class="sub-header">Set information</h3>
        <form class="form-inline" role="form">
          <div class="form-group">
            <table>
              <tr>
                <td><label for="newInfo">Info:</label> </td>
                <td>
                  <input class="form-control" id="newInfo">
                </td>
              </tr>
            </table>
          </div>
          <a href="#" onclick="registerSetInfo()" class="btn
btn-primary">Set</a>
        </form>
      </div>
    </div>
    <div class="row">
```

```
      <div>
        <h3 class="sub-header">Get last information saved</h3>
        <form class="form-inline" role="form">
          <a href="#" onclick="registerGetInfo()" class="btn
btn-primary">Get</a>
          <div class="form-group">
            <table>
              <tr>
                <td>Info:</td>
                <td>
                  <label id="lastInfo">
                </td>
              </tr>
            </table>
          </div>
        </form>
      </div>
    </div>

  </body>
</html>
```

برای ایجاد فایل index.js همانند قبل، دستور gedit index.js را در خط فرمان نوشته و کدهای نشان داده شده در لیست ۵-۳ را جای گذاری کنید.

لیست ۵-۳: index.js

```
// Source code to interact with smart contract

// web3 provider with fallback for old version
window.addEventListener('load', async () => {
  // New web3 provider
  if (window.ethereum) {
      window.web3 = new Web3(ethereum);
      try {
          // ask user for permission
          await ethereum.enable();
          // user approved permission
      } catch (error) {
          // user rejected permission
          console.log('user rejected permission');
      }
  }
  // Old web3 provider
  else if (window.web3) {
      window.web3 = new Web3(web3.currentProvider);
      // no need to ask for permission
  }
  // No web3 provider
  else {
      console.log('No web3 provider detected');
  }
});
console.log (window.web3.currentProvider)
```

```
// contractAddress and abi are setted after contract deploy
var contractAddress = '0xc864D0fef177A69aFa8E302A1b90e450910A4
c3E';
var abi = JSON.parse( '[{"constant":true,"inputs":[],"name":"g
etInfo","outputs":[{"name":"","type":"string"}],"payable":fals
e,"stateMutability":"view","type":"function"},{"constant":fals
e,"inputs":[{"name":"_info","type":"string"}],"name":"setInfo"
,"outputs":[],"payable":false,"stateMutability":"nonpayable","
type":"function"}]' );

//contract instance
contract = new web3.eth.Contract(abi, contractAddress);

// Accounts
var account;

web3.eth.getAccounts(function(err, accounts) {
  if (err != null) {
    alert("Error retrieving accounts.");
    return;
  }
  if (accounts.length == 0) {
    alert("No account found! Make sure the Ethereum client is
configured properly.");
    return;
  }
  account = accounts[0];
  console.log('Account: ' + account);
  web3.eth.defaultAccount = account;
});

//Smart contract functions
function registerSetInfo() {
  info = $("#newInfo").val();
  contract.methods.setInfo (info).send( {from: account}).then(
 function(tx) {
    console.log("Transaction: ", tx);
  });
  $("#newInfo").val('');
}

function registerGetInfo() {
  contract.methods.getInfo().call().then( function( info ) {
    console.log("info: ", info);
    document.getElementById('lastInfo').innerHTML = info;
  });
}
```

این قسمت با استفاده از کیف پول تزریق‌شده، در مورد اینجا، Metamask، به گره‌ی RSK Local متصل می‌شود:

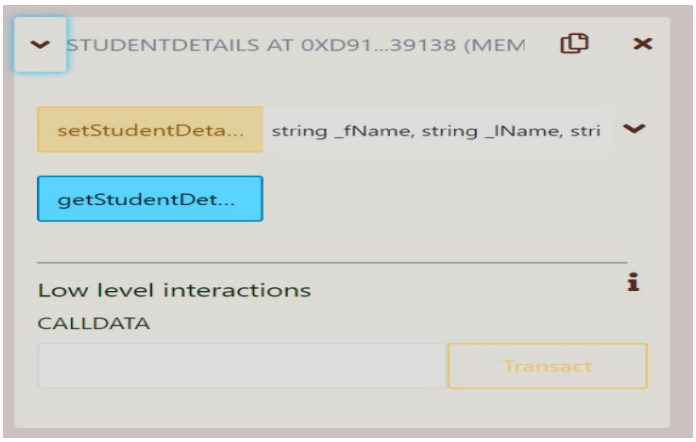
لیست ۵-٤:اتصال کیف پول به شبکه‌ی RSK Local

```
// web3 provider with fallback for old version
window.addEventListener('load', async () => {
  // New web3 provider
  if (window.ethereum) {
      window.web3 = new Web3(ethereum);
      try {
          // ask user for permission
          await ethereum.enable();
          // user approved permission
      } catch (error) {
          // user rejected permission
          console.log('user rejected permission');
      }
  }
  // Old web3 provider
  else if (window.web3) {
      window.web3 = new Web3(web3.currentProvider);
      // no need to ask for permission
  }
  // No web3 provider
  else {
      console.log('No web3 provider detected');
  }
});
```

خط زیر را در کد index.js بروز رسانی کرده و در آن آدرس قرارداد خود را جایگذاری کنید:

```
var contractAddress = '0xc864D0fef177A69aFa8E302A1b90e450910A4
c3E';
```

برای پیدا کردن آدرس قرارداد خود، همان‌طور که در شکل ۵-۷ نشان داده شده است، وقتی‌که قرارداد با کلیک روی دکمه‌ی Deploy استقرار یافت، می‌توان آن را از برگه‌ی Run در مرورگر Remix کپی کرد.



شکل ۵-۷: رونوشت آدرس قرارداد را بنویسید

سپس در پوشه‌ی Register، یک فایل server.js ایجاد کرده و کد زیر را در آن جایگذاری کنید.

لیست ۵-۵: server.js

```
var express = require('express');
var app = express();
app.use(express.static(__dirname));
app.listen('3300');
console.log('Running at\nhttp://localhost:3300');
```

اجرای سرور

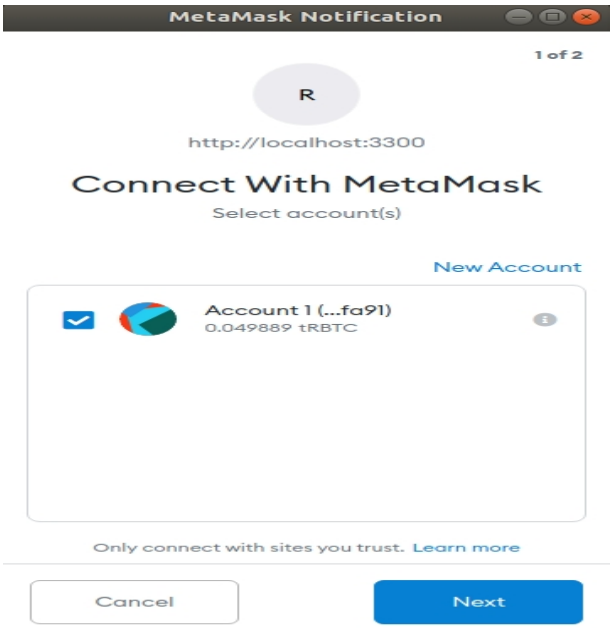
آخرین مرحله اجرای سرور Express است. دستور زیر را در ترمینال وارد کنید (شکل ۵-۸).

```
node server.js
```

```
root@blockchain-HuronRiver-Platform:/test/Register# node server.js
Running at
http://localhost:3300
```

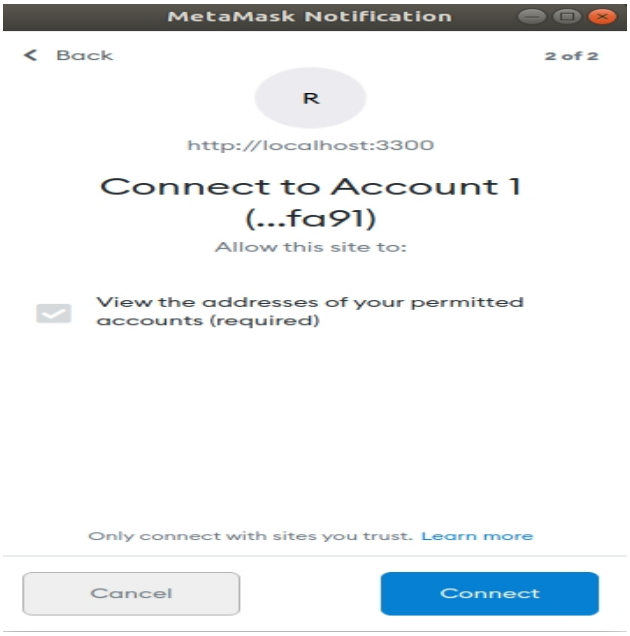
شکل ۵-۸: اجرای سرور

هنگامی‌که کیف پول Metamask ظاهر شد، بر روی دکمه‌ی next کلیک کنید (شکل ۵-۹).



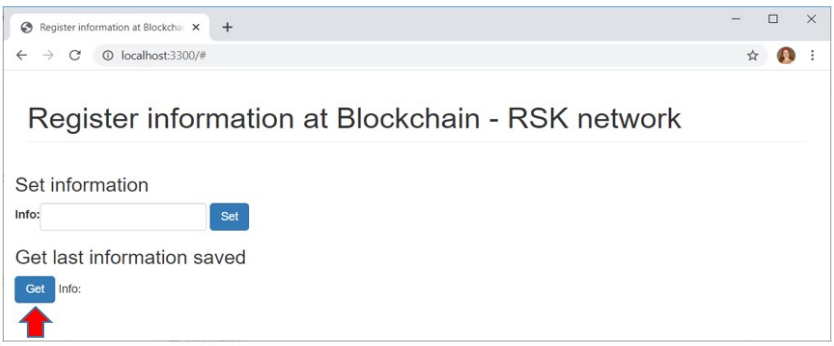
شکل ۵-۹: انتخاب اکانت کاربری برای اتصال به کیف پول

در صفحه‌ی بعد روی connect کلیک کنید (شکل ۵-۱۰).

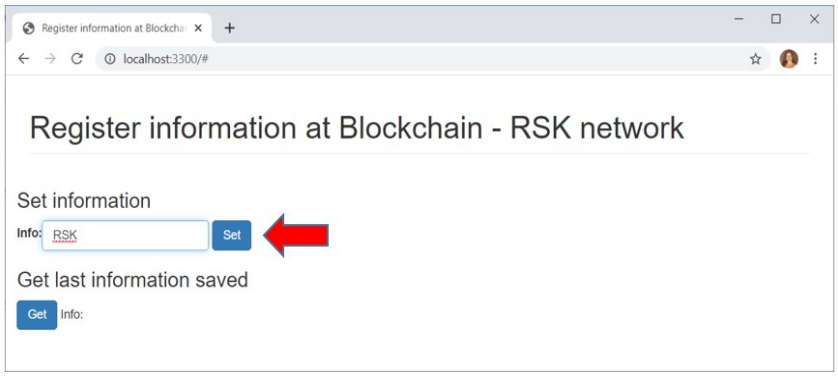


شکل ۵-۱۰ اتصال به کیف پول

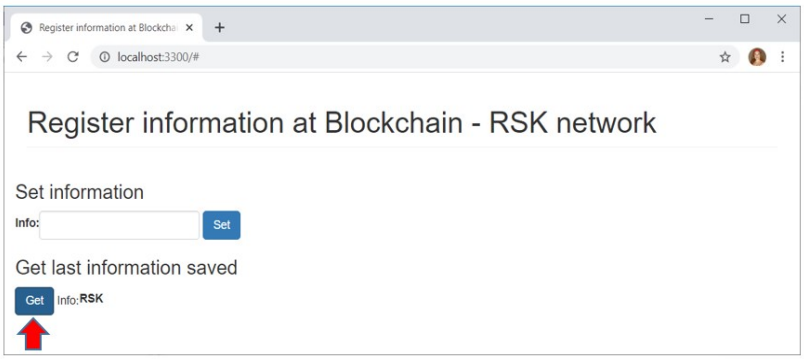
تعامل با قرارداد هوشمند



شکل ۵-۱۱: گرفتن اطلاعات از طریق دکمه Get



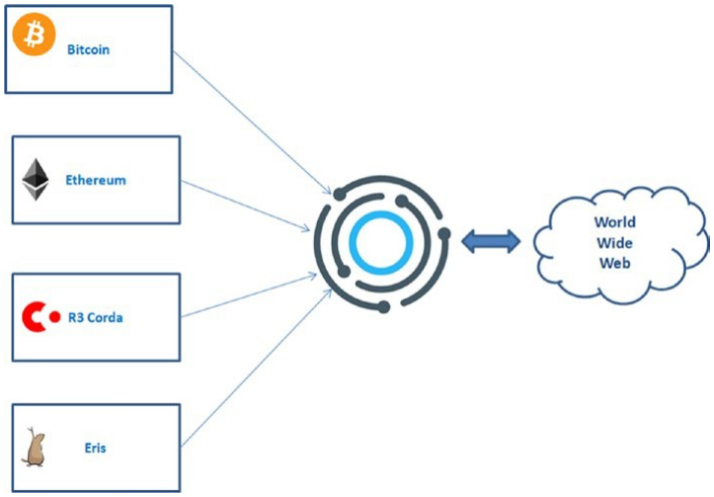
شکل ۵-۱۲: پنجره‌ی Set



شکل ۵-۱۳: بررسی اطلاعات در پنجره getInfo

فصل ششم

برنامه‌نویسی پیشرفته در Oraclize و IPFS و بهترین تمرین‌ها در این زمینه‌ها



شکل ٦-١: Oraclize

لیست ٦-١: USDRate.sol

```
pragma solidity ^0.4.0;
import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";
contract USDRate is usingOraclize {
    uint public price;
    event Log USDRate(string text);
    function USDRate() {
        USDRate("USDRate Contract created.");
        update();
    }
    function getPrice() constant returns (uint) {
       return price;
    }
}
```

لیست ٦-٢: Payable function

```
  function update() payable {
     Log("Oraclize query was sent, waiting for the answer..");
     oraclize_query("URL","json(https://min-api.cryptocompare.
  com/data/price?fsym=ETH&tsyms=USD).USD");
  }
```

لیست ٦-٣: Callback function

```
   function __callback(bytes32 _myid, string _result) {
      require (msg.sender == oraclize_cbAddress());
      Log(_result);
      price = parseInt(_result, 2); // let's save it as $ cents
```

```
}
```

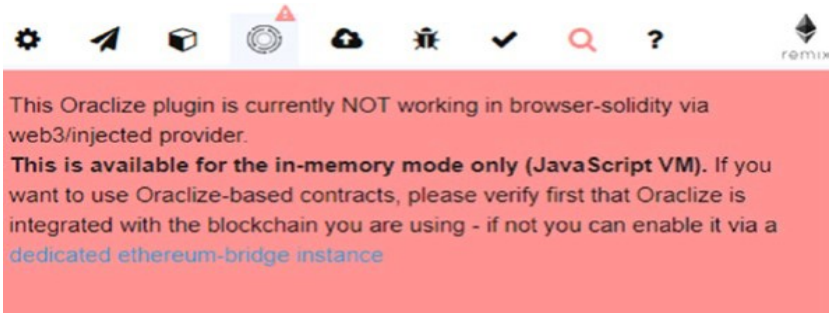
لیست ٦-٤:USDRate.sol

```
pragma solidity ^0.4.0;
import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";
contract USDRate is usingOraclize {
    uint public price;
    event LogUSDRate(string text);
    constructor() {
        emit LogUSDRate("USDRate Contract created.");
        update();
    }
    function update() payable {
        emit LogUSDRate("Oraclize query was sent, waiting for
the answer..");
        oraclize_query("URL","json(https://min-api.
cryptocompare.com/data/price?fsym=ETH&tsyms=USD).USD");
    }
    function __callback(bytes32 _myid, string _result) {
        require (msg.sender == oraclize_cbAddress());
        emit LogUSDRate(_result);
        price = parseInt(_result, 2); // let's save it as $
cents
    }
    function getPrice() constant returns (uint) {
       return price;
    }
}
```

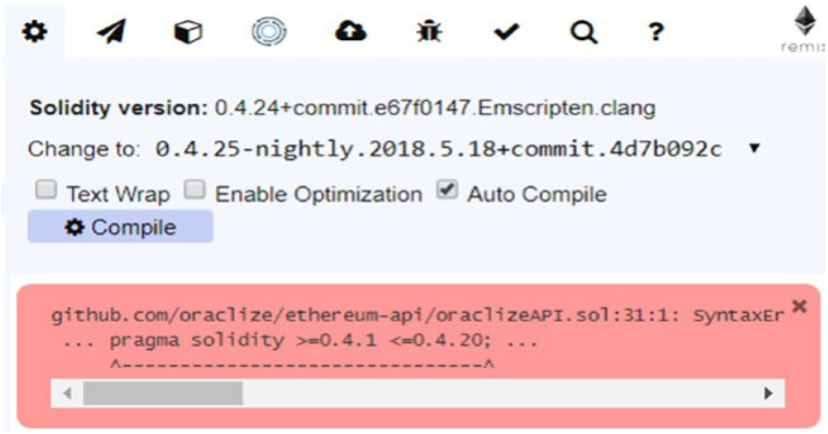
README.md | DieselPrice.sol | KrakenPriceTicker.sol | WolframAlpha.sol | YoutubeViews.sol

```
/*
   Diesel Price Peg

   This contract keeps in storage a reference
   to the Diesel Price in USD
*/


pragma solidity ^0.4.0;
import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";

contract DieselPrice is usingOraclize {

    uint public DieselPriceUSD;

    event newOraclizeQuery(string description);
    event newDieselPrice(string price);

    function DieselPrice() {
        update(); // first check at contract creation
    }

    function __callback(bytes32 myid, string result) {
        if (msg.sender != oraclize_cbAddress()) throw;
        newDieselPrice(result);
        DieselPriceUSD = parseInt(result, 2); // let's save it as $ cents
        // do something with the USD Diesel price
    }

    function update() payable {
        newOraclizeQuery("Oraclize query was sent, standing by for the answer..");
        oraclize_query("URL", "xml(https://www.fueleconomy.gov/ws/rest/fuelprices).fuelPrices.diesel");
    }

}
```

شکل ٦-٢: قراردادها در آدرس https://dev.oraclize.it/



شکل ٦-٣: هشدار Oraclize.



شکل ٦-٤: مشکلات کامپایل در Oraclize.

```
Python encrypted_queries_tools.py -e -p
044992e9473b7d90ca54d2886c7addd14a61109af202f1c95e218b0c99eb06
0c7134c4ae46345d0383ac996185762f04997d6fd6c393c86e4325c469741e
64eca9 "
```

این رشته‌ی خروجی رمزگذاری شده است:

```
BEIGVzv6fJcFiYQNZF8ArHnvNMAsAWBz8Zwl0YCsy4K/RJTN8ERHfBWtSfYHt+
uegdD1wtXTkP30sTW+3xR3w/un1i3caSO0Rfa+wmIMmNHt4aOS
```

سپس می‌تواند به‌عنوان آرگومان برای پرس‌وجوی Oraclize استفاده شود.

```
oraclize_query("URL","AzK149Vj4z65WphbBPiuWQ2PStTINeVp5sS9PSwq
Zi8NsjQy6jJLH765qQu3U/bZPNeEB/bYZJYBivwmmREXTGjmKJk/62ikcO6mIM
QfB5jBVVUOqzzZ/A8ecWR2nOLv0CKkkkFzBYp2sW1H31GI+SQzWV9q64WdqZsA
"a4gXqHb6jmLkVFjOGI0JvrAZh6T5lyeLPSmaslI
```

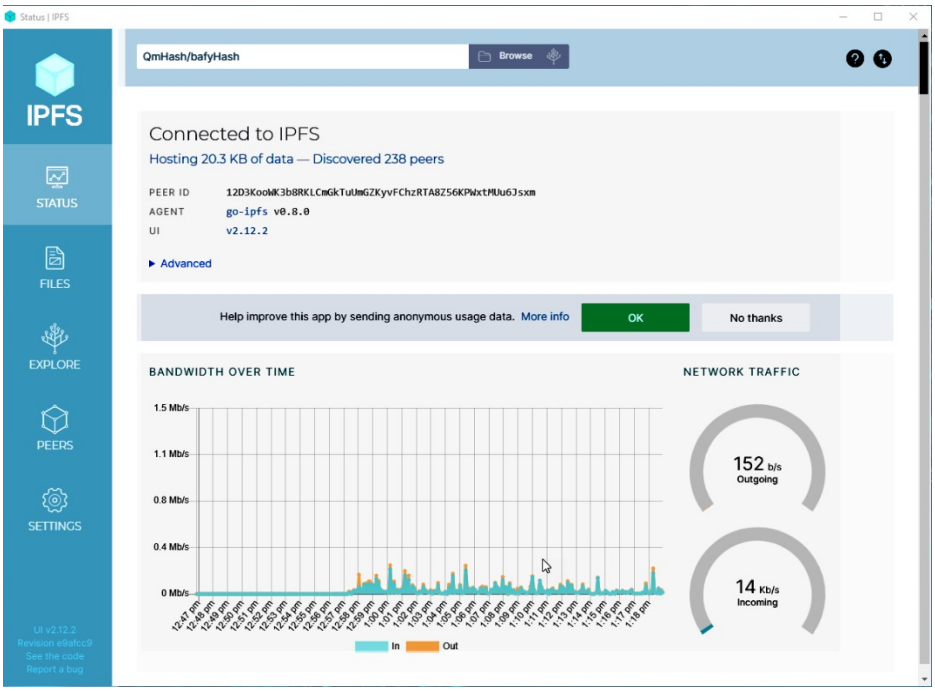
لیست ٦-٥: پرس‌وجوی Oraclize

```
oraclize_query("BEIGVzv6fJcFiYQNZF8ArHnvNMAsAWBz8Zwl0YCsy4K/RJ
TN8ERHfBWtSfYHt+uegdD1wtXTkP30sTW+3xR3w/un1i3caSO0Rfa+wmIMmNHt
4aOS","BNKdFtmfmazLLR/bfey4mP8
v/R5zCIUK7obcUrF2d6CWUMvKKUorQqYZNu1YfRZsGlp/F96CAQhSGomJC7oJa
3PktwoW5J1Oti/y2v4+b5+vN8yLIj1trS7p1l341Jf66AjaxnoFPplwLqE=","
BF5u1td9ugoacDabyfVzoTxPBxGNtmXuGV7AFcO1GLmXkXIKlBcAcelvaTKIbm
aA6lXwZCJCSeWDHJOirHiEl1LtR8lCt+1ISttWuvp
J6sPx3Y/QxTajYzxZfQb6nCGkv+8cczX0PrqKKwOn/Elf9kpQQCXeMglunT
09H2B4HfRs7uuI");
```

لیست ٦-٦: پرس‌وجوی Oracle مبتنی بر زمان

```
oraclize_query(60, "URL",
"json(http://api.fixer.io/latest?symbols=USD,GBP).rates.GBP");
```

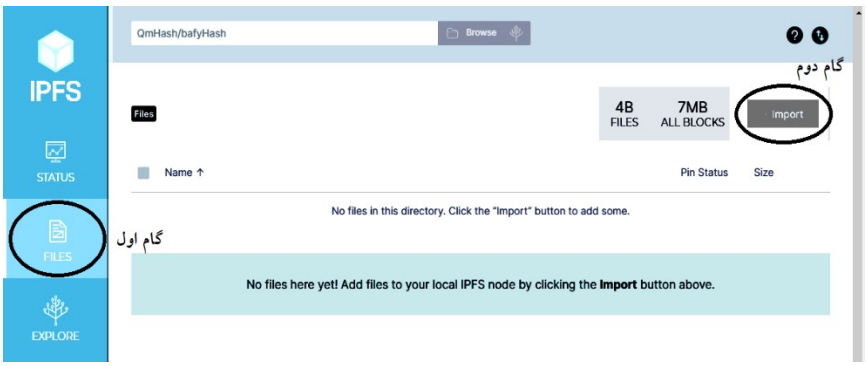
مراحل نصبIPFS



شکل ٦-٥: شماتیکی کلی از فضای IPFS دسکتاپ

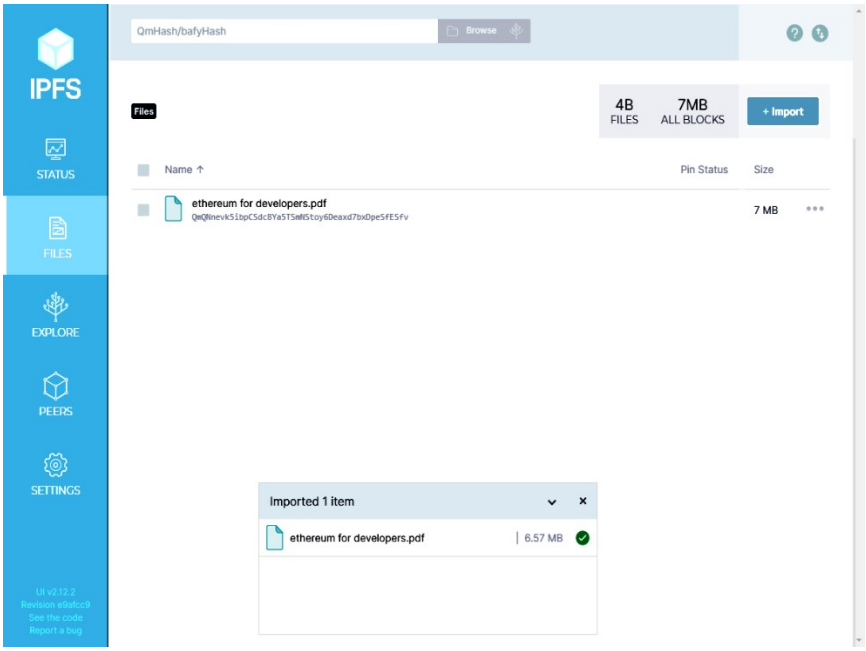
```
Command Prompt
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\p>ipfs version
ipfs version 0.8.0

C:\Users\p>
```

شکل ٦-٦: اطمینان از نصب IPFS بر روی سیستم با استفاده از خط فرمان



شکل ٦-٧: بارگذاری فایل در IPFS.

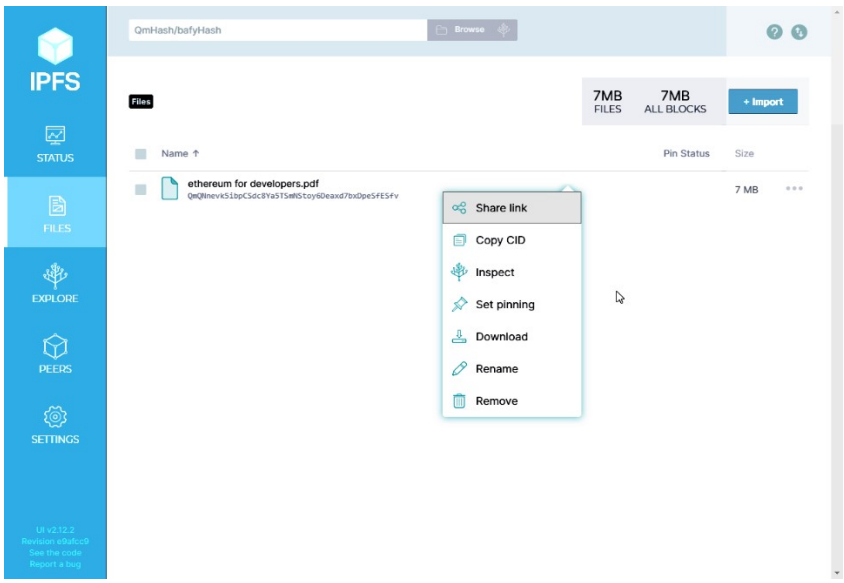
۱. با انتخاب فایل مورد نظر شما، فایل در فضای IPFS بارگذاری شده و به شما یک CID می‌دهد. برای درک بهتر این مرحله شکل (٦-٨) را مشاهده کنید.



شکل ٦–٨: فایل بارگذاری شده در IPFS که به آن یک CID اختصاص داده شده است.

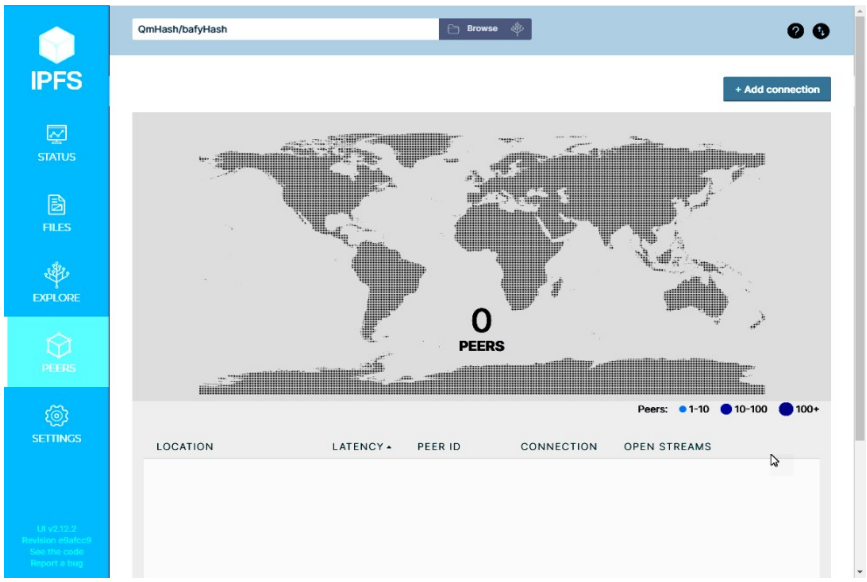
۲. با کلیک راست بر روی فایل می‌توانید به CID خود دسترسی داشته باشید.

توجه داشته باشید که فایل‌ها در IPFS مبتنی بر محتوا تجزیه، و پس از درخواست، ترکیب می‌شوند و در اختیار شما قرار می‌گیرند.



شکل ٦–٩: دسترسی به CID در IPFS

۳.

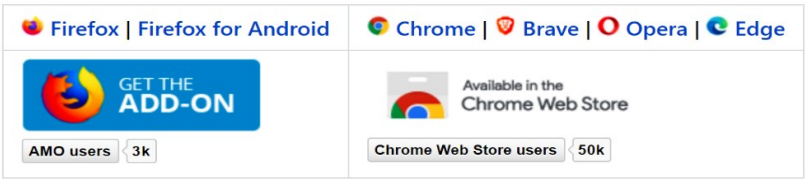


شکل ٦-١٠: نقشه اتصال peer ها در IPFS.

نصب افزونه IPFS

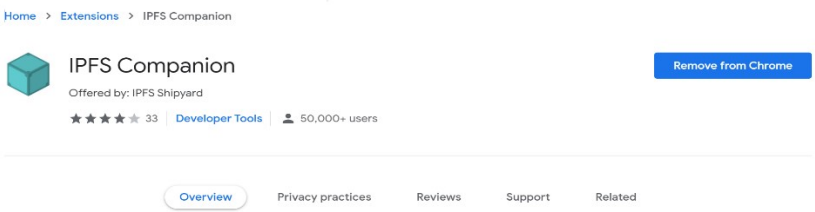
١.برای نصب افزونه ابتدا به آدرسِ https://github.com/ipfs/ipfs-companion مراجعه کنید.

٢.افزونه‌ی web browser مورد نظرتان را انتخاب کنید (شکل ٦-١١).



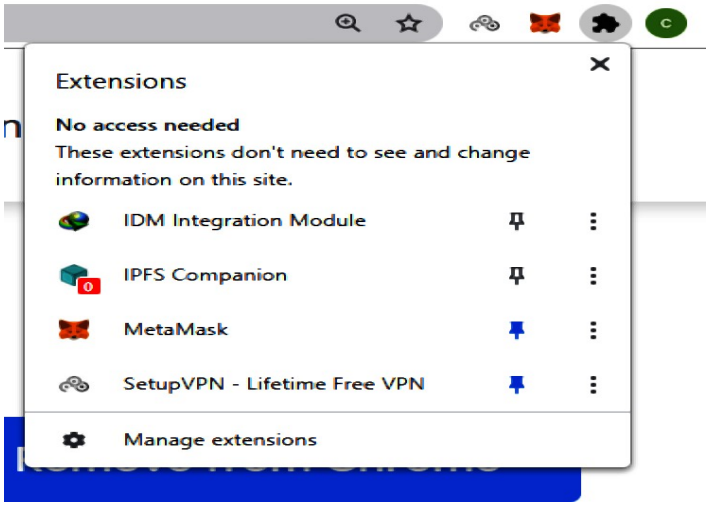
شکل ٦-١١: نصب افزونه IPFS

٣.مطابق شکل ٦-١٢ گزینه add chrome را انتخاب کنید.



شکل ٦-١٢: مراحل نصب افزونه IPFS به chrome.

٤.اکنون افزونه‌ی IPFS به جستجوگر وب شما اضافه شده است.



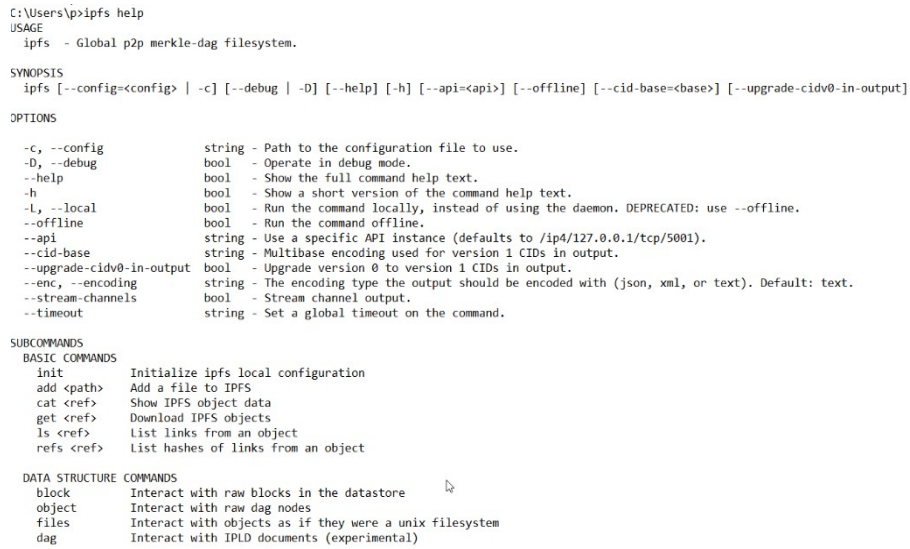
شکل ٦-١٣: افزونه IPFS

```
Command Prompt
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\p>ipfs version
ipfs version 0.8.0

C:\Users\p>ipfs commands
ipfs
ipfs add
ipfs bitswap
ipfs bitswap ledger
ipfs bitswap reprovide
ipfs bitswap stat
ipfs bitswap wantlist
ipfs block
ipfs block get
ipfs block put
ipfs block rm
ipfs block stat
ipfs bootstrap
ipfs bootstrap add
ipfs bootstrap add default
ipfs bootstrap list
ipfs bootstrap rm
ipfs bootstrap rm all
ipfs cat
ipfs cid
ipfs cid base32
```

شکل ٦-١٤: دستورات IPFS.

لیست ٦-٨: دستورات IPFS

```
ipfs
ipfs add
ipfs bitswap
ipfs bitswap ledger
ipfs bitswap reprovide
ipfs bitswap stat
ipfs bitswap wantlist
ipfs block
ipfs block get
ipfs block put
ipfs block rm
ipfs block stat
ipfs bootstrap
ipfs bootstrap add
ipfs bootstrap add default
ipfs bootstrap list
ipfs bootstrap rm
ipfs bootstrap rm all
ipfs cat
ipfs cid
ipfs cid base32
ipfs cid bases
ipfs cid codecs
ipfs cid format
ipfs cid hashes
ipfs commands
ipfs config
ipfs config edit
ipfs config profile
ipfs config profile apply
ipfs config replace
ipfs config show
ipfs daemon
ipfs dag
ipfs dag export
ipfs dag get
ipfs dag import
ipfs dag put
ipfs dag resolve
ipfs dag stat
ipfs dht
ipfs dht findpeer
ipfs dht findprovs
ipfs dht get
ipfs dht provide
ipfs dht put
ipfs dht query
ipfs diag
ipfs diag cmds
```



```
C:\Users\p>ipfs help
USAGE
  ipfs  - Global p2p merkle-dag filesystem.

SYNOPSIS
  ipfs [--config=<config> | -c] [--debug | -D] [--help] [-h] [--api=<api>] [--offline] [--cid-base=<base>] [--upgrade-cidv0-in-output]

OPTIONS

  -c, --config                string - Path to the configuration file to use.
  -D, --debug                 bool   - Operate in debug mode.
  --help                      bool   - Show the full command help text.
  -h                          bool   - Show a short version of the command help text.
  -L, --local                 bool   - Run the command locally, instead of using the daemon. DEPRECATED: use --offline.
  --offline                   bool   - Run the command offline.
  --api                       string - Use a specific API instance (defaults to /ip4/127.0.0.1/tcp/5001).
  --cid-base                  string - Multibase encoding used for version 1 CIDs in output.
  --upgrade-cidv0-in-output   bool   - Upgrade version 0 to version 1 CIDs in output.
  --enc, --encoding           string - The encoding type the output should be encoded with (json, xml, or text). Default: text.
  --stream-channels           bool   - Stream channel output.
  --timeout                   string - Set a global timeout on the command.

SUBCOMMANDS
  BASIC COMMANDS
    init          Initialize ipfs local configuration
    add <path>    Add a file to IPFS
    cat <ref>     Show IPFS object data
    get <ref>     Download IPFS objects
    ls <ref>      List links from an object
    refs <ref>    List hashes of links from an object

  DATA STRUCTURE COMMANDS
    block         Interact with raw blocks in the datastore
    object        Interact with raw dag nodes
    files         Interact with objects as if they were a unix filesystem
    dag           Interact with IPLD documents (experimental)
```

شکل ٦–١٥: راهنمای IPFS.

لیست ٦–٩: خروجی راهنمای ipfs

```
C:\Users\p>ipfs help
USAGE
  ipfs  - Global p2p merkle-dag filesystem.

SYNOPSIS
  ipfs [--config=<config> | -c] [--debug | -D] [--help] [-h] [--
api=<api>] [--offline] [--cid-base=<base>] [--upgrade-cidv0-in-output]
[--encoding=<encoding> | --enc] [--timeout=<timeout>] <command> ...

OPTIONS

  -c, --config                string - Path to the configuration file
to use.
  -D, --debug                 bool   - Operate in debug mode.
  --help                      bool   - Show the full command help text.
  -h                          bool   - Show a short version of the
command help text.
  -L, --local                 bool   - Run the command locally, instead
of using the daemon. DEPRECATED: use --offline.
  --offline                   bool   - Run the command offline.
  --api                       string - Use a specific API instance
(defaults to /ip4/127.0.0.1/tcp/5001).
  --cid-base                  string - Multibase encoding used for
version 1 CIDs in output.
  --upgrade-cidv0-in-output   bool   - Upgrade version 0 to version 1
CIDs in output.
  --enc, --encoding           string - The encoding type the output
should be encoded with (json, xml, or text). Default: text.
  --stream-channels           bool   - Stream channel output.
  --timeout                   string - Set a global timeout on the
command.
```

```
SUBCOMMANDS
  BASIC COMMANDS
    init          Initialize ipfs local configuration
    add <path>    Add a file to IPFS
    cat <ref>     Show IPFS object data
    get <ref>     Download IPFS objects
    ls <ref>      List links from an object
    refs <ref>    List hashes of links from an object

  DATA STRUCTURE COMMANDS
    block         Interact with raw blocks in the datastore
    object        Interact with raw dag nodes
    files         Interact with objects as if they were a unix
filesystem
    dag           Interact with IPLD documents (experimental)

  ADVANCED COMMANDS
    daemon        Start a long-running daemon process
    mount         Mount an IPFS read-only mount point
    resolve       Resolve any type of name
    name          Publish and resolve IPNS names
    key           Create and list IPNS name keypairs
    dns           Resolve DNS links
    pin           Pin objects to local storage
    repo          Manipulate the IPFS repository
    stats         Various operational stats
    p2p           Libp2p stream mounting
    filestore     Manage the filestore (experimental)

  NETWORK COMMANDS
    id            Show info about IPFS peers
    bootstrap     Add or remove bootstrap peers
    swarm         Manage connections to the p2p network
    dht           Query the DHT for values or peers
    ping          Measure the latency of a connection
    diag          Print diagnostics

  TOOL COMMANDS
    config        Manage configuration
    version       Show ipfs version information
    update        Download and apply go-ipfs updates
    commands      List all available commands
    cid           Convert and discover properties of CIDs
    log           Manage and show logs of running daemon

  Use 'ipfs <command> --help' to learn more about each command.

  ipfs uses a repository in the local file system. By default, the
repo is
  located at ~/.ipfs. To change the repo location, set the $IPFS PATH
  environment variable:

    export IPFS_PATH=/path/to/ipfsrepo

  EXIT STATUS

  The CLI will exit with one of the following values:

  0     Successful execution.
  1     Failed executions.

  For more information about each command, use:
  'ipfs <subcmd> --help'
```

اجرای IPFS به کمک خط فرمان

```
Devjani@DESKTOP-UHTA6E3 MINGW64 /f/ethereum/ipfs-example/ethereum-ipfs (master)
$ export IPFS_PATH=.ipfs

Devjani@DESKTOP-UHTA6E3 MINGW64 /f/ethereum/ipfs-example/ethereum-ipfs (master)
$ echo $IPFS_PATH
.ipfs

Devjani@DESKTOP-UHTA6E3 MINGW64 /f/ethereum/ipfs-example/ethereum-ipfs (master)
$ ipfs init
initializing IPFS node at .ipfs
generating 2048-bit RSA keypair...done
peer identity: QmPGx16WcXGWp2bTgusV5xsDgRn83XoHfbN1M7Zq3eCBYv
to get started, enter:

        ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme


Devjani@DESKTOP-UHTA6E3 MINGW64 /f/ethereum/ipfs-example/ethereum-ipfs (master)
$ ipfs config --json API.HTTPHeaders.Access-Control-Allow-Origin '["*"]'

Devjani@DESKTOP-UHTA6E3 MINGW64 /f/ethereum/ipfs-example/ethereum-ipfs (master)
$ ipfs config --json Gateway.HTTPHeaders.Access-Control-Allow-Origin '["*"]'
```

شکل ٦-١٦: دستورات IPFS.

لیست ٦-١٠: دستور IPFS cat

```
$ ipfs init
initializing IPFS node at .ipfs
generating 2048-bit RSA keypair...done
peer identity:QmPGx16WcXGWp2bTgusV5xsDgRn83XoHfbN1M7Zq3eCBYv
to get started, enter:
 ipfs cat /ipfs/
QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
```

توجه داشته باشید که این دستور باید تنها یک‌بار اجرا شود. اگر دوباره آن را اجرا کنید، خطا خواهد داد. اکنون شما باید با استفاده از دستورات زیر، اشتراک منبع متقاطع[1](CORS) را در گره‌ی IPFS فعال کنید. CORS فعال می‌شود تا به همه‌ی درخواست‌ها اجازه داده شود.

```
ipfs config --jsonAPI.HTTPHeaders.Access-Control-Allow-
Origin '["*"]'
ipfs config --jsonGateway.HTTPHeaders.Access-Control-
Allow-Origin '["*"]'
```

توجه: به اشتراک‌گذاری منبع متقاطع، مکانیسمی است که اجازه می‌دهد منابع محدود (به‌عنوان مثال فونت‌ها) در یک صفحه وب از دامنه‌ی دیگری خارج از دامنه‌ای که منبع

[1] Cross-origin resource sharing

اول از آن ارائه شده است، درخواست شود. یک صفحه‌ی وب می‌تواند آزادانه تصاویر متقابل، style sheets، اسکریپت، iframes و ویدئو را در خود جای دهد. اکنون برای اجرای سرور می‌توانید دستور زیر را اجرا کنید (اگر از سیستم‌عامل ویندوز استفاده می‌کنید، این را در حالت bash اجرا کنید):

```
ipfs daemon
```

خروجی شبیه لیست ٦-١١ خواهد بود.

لیست ٦-١١: خروجی دستور ipfs daemon

```
$ ipfs daemon
Initializing daemon...
Swarm listening on /ip4/10.0.75.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.164.245/tcp/4001
Swarm listening on /ip4/169.254.170.245/tcp/4001
Swarm listening on /ip4/169.254.189.121/tcp/4001
Swarm listening on /ip4/169.254.230.188/tcp/4001
Swarm listening on /ip4/172.18.160.1/tcp/4001
Swarm listening on /ip4/192.168.1.6/tcp/4001
Swarm listening on /ip4/192.168.51.129/tcp/4001
Swarm listening on /ip6/:1/tcp/4001
Swarm listening on /p2p-circuit/ipfs/
QmPGx16WcXGWp2bTgusV5xsDgRn83XoHfbN1M7Zq3eCBYv
Swarm announcing /ip4/10.0.75.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.164.245/tcp/4001
Swarm announcing /ip4/169.254.170.245/tcp/4001
Swarm announcing /ip4/169.254.189.121/tcp/4001
Swarm announcing /ip4/169.254.230.188/tcp/4001
Swarm announcing /ip4/172.18.160.1/tcp/4001
Swarm announcing /ip4/192.168.1.6/tcp/4001
Swarm announcing /ip4/192.168.51.129/tcp/4001
Swarm announcing /ip6/:1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

اکنون برای پیدا کردن همه‌ی peerهایی که محتوایی که شما در حافظه IPFS آپلود کرده‌اید را به اشتراک خواهند گذاشت، دستور زیر را اجرا کنید:

```
ipfs swarm peers
```

خروجی، مشابهِ لیست ١٢-٦ خواهد بود.

لیست ٦-١٢: دستور IPFS برای اضافه کردن پرونده به خروجی کنسول

```
$ ipfs swarm peers
```

```
/ip4/100.34.210.63/tcp/14655/ipfs/
QmPRa5sovWPGhSDuEGU2cgfws5ra91bD89xTWmArJxickp
/ip4/100.38.242.117/tcp/24885/ipfs/
QmfGpAZPq1br1G6Q9KcLNqGKLnRjjmXvc6BD5yFfRVQv2y
```

اکنون یک پوشه ایجاد کنید و تصویری به نام E.jpeg را به آن اضافه کنید. به آن پوشه بروید و دستور زیر را اجرا کنید:

```
ipfs add -r
```

خروجی مشابهِ لیست ٦-١٣ است.

لیست ٦-١٣: پاسخ خروجی دستور add

```
$ ipfs add -r .
added        QmbFDRLYyZaaTv3EJmz2QGtUFpvDT9rM7xPQrQkjUuX4qc
images/E.jpeg
added QmUFVahZy2eLbyVqjxK47aoSCDW1MGzoCig5VA4z3sTMAD images
```

حال این فایل را از طریق دستور زیر با مقدار هشی که قبلا نشان داده شد، منتشر کنید:

```
ipfs name publish

QmUFVahZy2eLbyVqjxK47aoSCDW1MGzoCig5VA4z3sTMAD
```

خروجی مشابه لیست ٦-١٤ است.

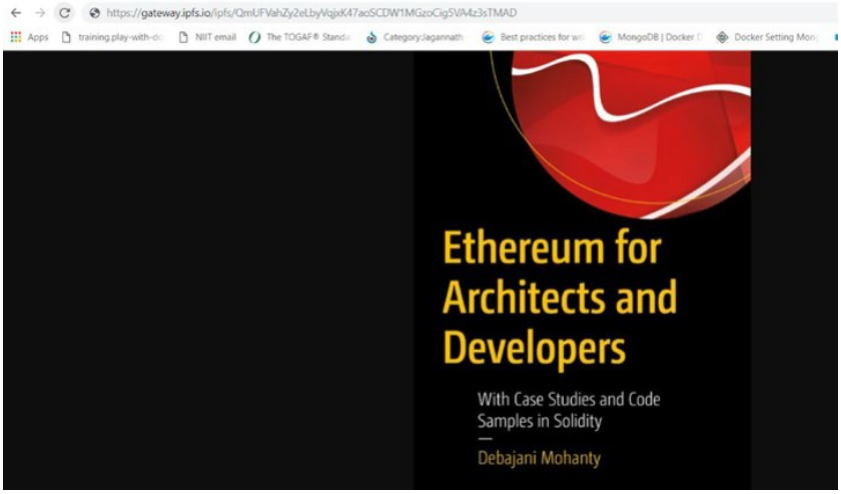
لیست ٦-١٤: دستور IPFS برای انتشار خروجی کنسول پرونده

```
$ ipfs name publish
QmUFVahZy2eLbyVqjxK47aoSCDW1MGzoCig5VA4z3sTMAD
Published to QmPGx16WcXGWp2bTgusV5xsDgRn83XoHfbN1M7Zq3eCBYv:
 /ipfs/QmUFVahZy2eLbyVqjxK47aoSCDW1MGzoCig5VA4z3sTMAD
```

اکنون می‌توانید تصویر مورد نظر را در وب‌سایت IPFS با استفاده از این URL جستجو کنید:

https://gateway.ipfs.io/ipfs/QmUFVahZy2eLbyVqjxK47aoSCDW1MGzoCig5VA4z3sTMAD.

همان‌طور که در شکل ٦-١٧ نشان داده شده است، حاوی مقدار هشی است که توسط IPFS برگردانده شده است. این URL را می‌توان در گره‌های اتریوم به‌صورت رشته‌ای ذخیره کرد و بر اساس نیاز شما قابل دسترسی خواهد بود.



شکل ٦-١٧: اضافه کردن پرونده به حافظه IPFS

لیست ٦-١٥: کد ریسکی

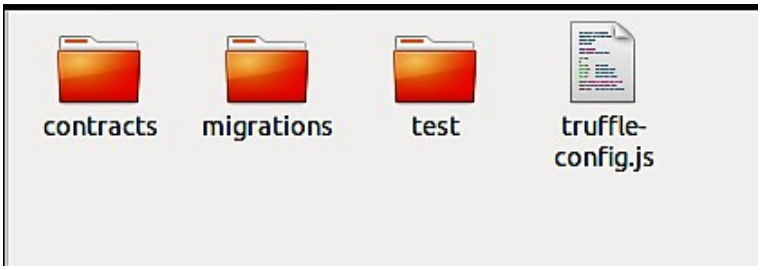
```
// Bad
mapping (address => uint) private userAmount;
function withdrawAmount() public {
uint amountToWithdraw = userAmount[msg.sender];
if (!(msg.sender.call.value(amountToWithdraw)())) { throw; }
        // At this point, the function withdrawAmount() is
called again
        userAmount[msg.sender] = 0;
}
```

لیست ٦-١٦: تمرین خوب برای به‌روزرسانی مقدار در اوایل برنامه

```
// Good
mapping (address => uint) private userAmount;
function withdrawAmount() public {
        uint amountToWithdraw = userAmount[msg.sender];
        userAmount[msg.sender] = 0;
        if (!(msg.sender.call.value(amountToWithdraw)())) {
throw; }
        // At this point, the caller's code is executed, and
can call withdrawAmount() again
}
```

فصل هفتم

Truffleو Embark



شکل ۷-۱: اجرای دستور truffle init و ایجاد پوشه‌های متناظر

لیست۷-۱: قرارداد HelloWorld.sol

```
// We will be using Solidity version 0.5.3
pragma Solidity 0.5.3;

contract HelloWorld {
    string private message = "hello world";

    function getMessage() public view returns(string memory) {
        return message;
    }
    function setMessage(string memory newMessage) public {
        message = newMessage;
    }
}
```

لیست۷-۲: فایل پیکربندی truffle-config.js

```
module.exports={
    networks:{
    },
    mocha:{
    },
    compilers:{
        solc:{
        }
    }
}
```

لیست۷-۳: پیکربندی Compiler

```
compilers:{
    solc:{
        version:"0.5.3",
        docker:false
```

```
            }
    }
```

```
root@thikpad-ThinkPad-E550:/home/thikpad/Ethereum/HelloWorld# truffle compile

Compiling your contracts...
===========================
✓ Fetching solc version list from solc-bin. Attempt #1
✓ Downloading compiler. Attempt #1.
> Compiling ./contracts/HelloWorld.sol
> Artifacts written to /home/thikpad/Ethereum/HelloWorld/build/contracts
> Compiled successfully using:
   - solc: 0.5.3+commit.10d17f24.Emscripten.clang

root@thikpad-ThinkPad-E550:/home/thikpad/Ethereum/HelloWorld#
```

شکل ۷-۲: کامپایل کردن truffle

لیست۷-٤: 2_deploy_contracts.js

```
var HelloWorld = artifacts.require("HelloWorld");
module.exports = function(deployer) {
  deployer.deploy(HelloWorld);
}
```

لیست۷-۵: تنظیمات شبکه

```
module.exports = {
    networks: {
       development: {
          host: "127.0.0.1",
          port: 8545,
          network_id: "*"
    }
       },
    mocha: {
    },

    compilers: {
          solc: {
        version: "0.5.3",
        docker: false
          }
    }
}
```

حال با دستور زیر قرارداد را مستقر می‌کنیم:

```
# truffle migrate
```

بعد از اجرای دستور، خروجی ترمینال به‌صورت زیر (شکل ۷-۳) خواهد بود.

```
root@thikpad-ThinkPad-E550:/home/thikpad/Ethereum/HelloWorld# truffle migrate

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

> Something went wrong while attempting to connect to the network. Check your network configuration.

Could not connect to your Ethereum client with the following parameters:
    - host       > 127.0.0.1
    - port       > 7545
    - network_id > 5777
Please check that your Ethereum client:
    - is running
    - is accepting RPC connections (i.e., "--rpc" option is used in geth)
    - is accessible over the network
    - is properly configured in your Truffle configuration file (truffle-config.js)

Truffle v5.3.6 (core: 5.3.6)
Node v14.16.1
```

شکل ۷-۳: پنجره خروجی با اجرای دستور truffle migrate

```
# ganache-cli -p 8545

# truffle migrate -network development
```

```
root@thikpad-ThinkPad-E550:/home/thikpad/Ethereum/HelloWorld# truffle migrate -network development

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

> Something went wrong while attempting to connect to the network. Check your network configuration.

Could not connect to your Ethereum client with the following parameters:
    - host       > 127.0.0.1
    - port       > 7545
    - network_id > 5777
Please check that your Ethereum client:
    - is running
    - is accepting RPC connections (i.e., "--rpc" option is used in geth)
    - is accessible over the network
    - is properly configured in your Truffle configuration file (truffle-config.js)

Truffle v5.3.6 (core: 5.3.6)
Node v14.16.1
```

شکل ۷-٤: اجرای دستور truffle migrate -network development

تعامل با قرارداد هوشمند


```
{
  "contractName": "HelloWorld",
  "abi": [
    {
      "constant": true,
      "inputs": [],
      "name": "getMessage",
      "outputs": [ ...
      ],
      "payable": false,
      "stateMutability": "view",
      "type": "function",
      "signature": "0xce6d41de"
    },
    {
      "constant": false,
      "inputs": [ ...
      ],
      "name": "setMessage",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function",
      "signature": "0x368b8772"
    }
  ],
```


شکل ۷-٦: کپی مقدار کلید abi

لیست ۷-٦: abi.js

```
var abi = [
  {
    "constant": true,
    "inputs": [],
    "name": "getMessage",
    "outputs": [
      {
        "name": "",
        "type": "string"
      }
    ],
    "payable": false,
    "stateMutability": "view",
    "type": "function",
    "signature": "0xce6d41de"
  },
  {
```

```
    "constant": false,
    "inputs": [
      {
        "name": "newMessage",
        "type": "string"
      }
    ],
    "name": "setMessage",
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function",
    "signature": "0x368b8772"
  }
];
```

لیست ۷-۷: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Hello World</title>
  <script src="https://cdn.jsdelivr.net/gh/ethereum/web3.js@1.
0.0-beta.36/dist/web3.min.js" integrity="sha256-
nWBTbvxhJgjslRyuAKJHK+XcZPlCnmIAAMixz6EefVk=" crossorigin="ano
nymous"></script>
  <script language="javascript" type="text/javascript" src="./
abi.js"></script>
  <script>
    // کد در این قسمت باید اضافه شود
  </script>
</head>
<body>
  <div id="message"></div>
</body>

</html>
```

```
Open ▾                                index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Hello World</title>
  <script src="https://cdn.jsdelivr.net/gh/ethereum/web3.js@1.0.0-beta.36/dist/web3.min.js" integrity="sha256-
nWBTbvxhJgjslRyuAKJHK+XcZPlCnmIAAMixz6EefVk=" crossorigin="anonymous"></script>
  <script language="javascript" type="text/javascript" src="./abi.js"></script>
  <script>
    // this will be where our code will be
        window.addEventListener('load', async () => {

    var contract;
    const contractAddress = "0xE0c2aFD252d248F22FE826f59a7d8B2ACE03ff02";

    const contractMessage = async () => {
      contract = new web3.eth.Contract(abi, contractAddress);
      let message = await contract.methods.getMessage().call();
      return message;
    }

    // Modern dapp browsers...
    if (window.ethereum) {
      window.web3 = new Web3(ethereum);
      try {
        // Request account access if needed
        await ethereum.enable();
        var message = await contractMessage();
        var elm = document.getElementById("message");
        elm.innerHTML = message;
      } catch (error) {
        // User denied account access...
      }
    }
    // Non-dapp browsers...
    else {
```

شکل ۷–۷: کد کامل

```
> transaction hash:    0x1856fdc8b34b7452e37d7128c41eddc6dbd85ba1493b0ec1
147cb70242c30b5
> Blocks: 0            Seconds: 0
> contract address:    0xE0c2aFD252d248F22FE826f59a7d8B2ACE03ff02
> block number:        4
> block timestamp:     1620830577
> account:             0xEc0538314b8597DccF0f44f60678b283bC432d3F
> balance:             99.9799196
```

شکل۷–۸: اجرای دستور truffle deploy

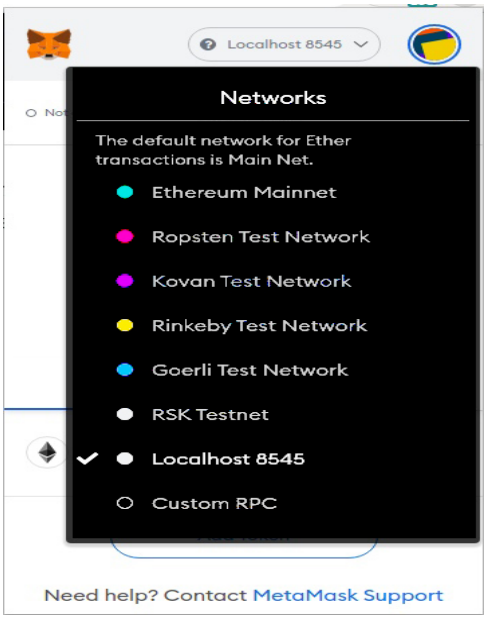
لیست۷–۸: فراخوانیِ async

```
if (window.ethereum) {
    window.web3 = new Web3(ethereum);
    try {
// Request account access if needed
    await ethereum.enable();
    var message = await contractMessage();
    var elm = document.getElementById("message");
    elm.innerHTML = message;
    } catch (error) {
// User denied account access…
    }
} else { // Non-dapp browsers…
    console.log('Non-
Ethereum browser detected. You should consider trying MetaMask
!');
}
```

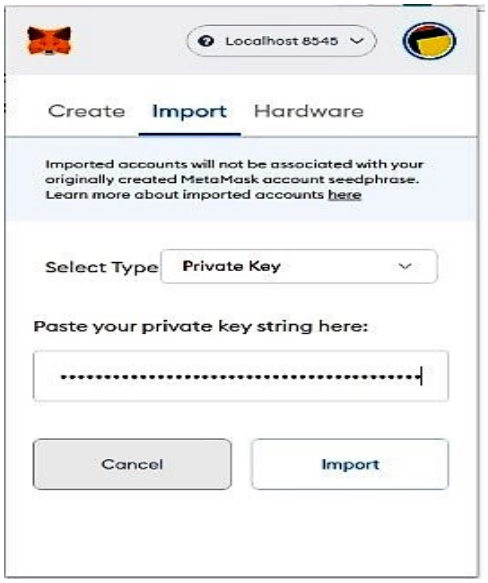
اتصال MetaMask

```
Private Keys
==================
(0) 0x2ca84749427a57a11e5570e57cb7986ca26b350b67794465235ae62092cc8874
(1) 0xc809557af63134c17ab531c5a7a7faccb5613f73baae1ab81abac47cf2c5ca3b
(2) 0xc8e3cdc0db8ee66d2c3057812dd724dc8d0f86adf1ef2602802cdd36ad68c234
(3) 0x1e52f3252bd9d40dda6d176d70aa99434ae0f61900e8c5c82f367a35f61c0f51
(4) 0xca779e08453904b3c86f034af44b4228fda088d320c4bb2df69e7fa07a26b493
(5) 0xd629a42e7267088ed47997d5551c1b82ab2bfae23942a8ee883e4d4734302d16
(6) 0x83981bc5c455b468398d583463e5bc13a01d91f0798ab69f27d10e92093d66b7
(7) 0xaf369431e9e2b7c3fb3a471c4b124e0d3194b905f9bc6777727fa6367363fbca
(8) 0x18ca4007b4bb885c41de0b882f8bc6868649db9b7510df9223e9ac421023878c
(9) 0xf6a993fbaba9cbccdd7f3b0a89e5b7994f15360d16a3cc22ba8501ef796e0931
```

شکل ۷-۹: لیست کلیدهای خصوصی

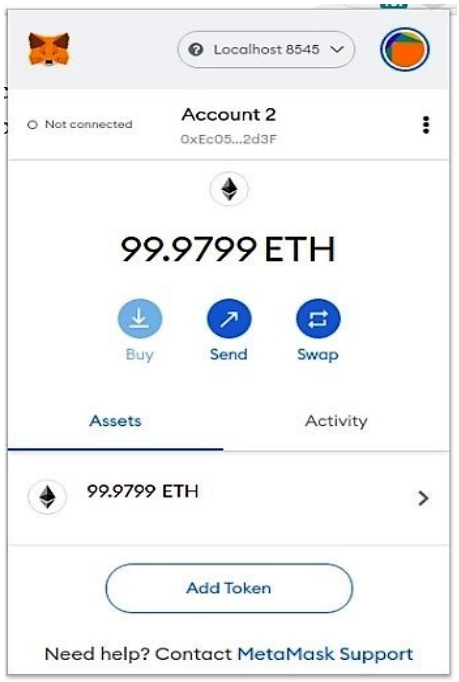


شکل ۷-۱۰: انتخاب شبکه مجازی برای اجرای قرارداد هوشمند



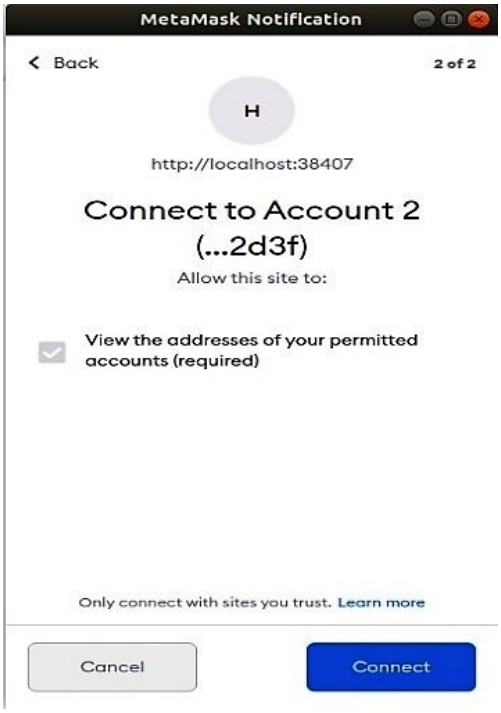
شکل ۷-۱۱: دریافت کلید خصوصی

اکنون یک حساب در MetaMask با حدود ۱۰۰ اتر خواهید داشت (شکل ۷-۱۲). حال باید برنامه‌ی وب خود را باز کنیم.



شکل ۷-۱۲: دریافت ۱۰۰ اتر در کیف پول

شروع برنامه‌ی وب



شکل ۷-۱۳: اتصال کیف پول و قرارداد هوشمند

صفحه‌ی باز شده در explorer به‌صورت زیر خواهد بود (شکل ۷-۱٤).

شکل ۷-۱٤: صفحه‌ی بازه شده در explorer

Embark

```
F:\ethereum\embark>embark demo
Initializing Embark Template....
Installing packages...
Init complete

App ready at embark_demo
-------------------
Next steps:
-> cd embark_demo
-> embark blockchain or embark simulator
open another console in the same directory and run
-> embark run
For more info go to http://embark.status.im

F:\ethereum\embark>_
```

شکل ۷- ۱۵: ایجاد embark_demo

لیست ۷-۹: ایجاد embark_demo

```
F:\ethereum\embark>embark demo
Initializing Embark Template....
Installing packages...
Init complete
App ready at embark_demo
-------------------
Next steps:
-> cd embark_demo
-> embark blockchain or embark simulator
   open another console in the same directory and run
-> embark run
For more info go to http://embark.status.im
```

برای اطلاعات بیشتر به سایت http://embark.status.im مراجعه کنید. همان‌طور که در شکل ۷-۱۶ و لیست ۷-۲ نشان داده شده است، برای اجرای ganache-cli، به پوشه‌ی آن بروید و دستور زیر را اجرا کنید :

```
embark simulator
```

```
Command Prompt - embark simulator

F:\ethereum\embark\embark_demo>embark simulator
Ganache CLI v6.1.8 (ganache-core: 2.2.1)

Available Accounts
==================
(0) 0xb8d851486d1c953e31a44374aca11151d49b8bb3 (~100 ETH)
(1) 0xf6d5c6d500cac10ee7e6efb5c1b479cfb789950a (~100 ETH)
(2) 0xf09324e7a1e2821c2f7a4a47675f9cf0b1a5eb7f (~100 ETH)
(3) 0xfbaf82a227dcebd2f9334496658801f63299ba24 (~100 ETH)
(4) 0x774b5341944deac70199a4750556223cb008949b (~100 ETH)
(5) 0x4801428dad07e7c2401d033d195116011fc4e400 (~100 ETH)
(6) 0xcf08befbc01a5b02ea09d840797d6b4565d4d535 (~100 ETH)
(7) 0x1a2f3b98e434c02363f3dac3174af93c1d690914 (~100 ETH)
(8) 0x4a17f35f0a9927fb4141aa91cbbc72c1b31598de (~100 ETH)
(9) 0xdf18cb4f2005bc52f94e9bd6c31f7b0c6394e2c2 (~100 ETH)

Private Keys
==================
(0) 0xf942d5d524ec07158df4354402bfba8d928c99d0ab34d0799a6158d56156d986
(1) 0x88f37cfbaed8c0c515c62a17a3a1ce2f397d08bbf20dcc788b69f11b5a5c9791
(2) 0xf4ebc8adae40bfc741b0982c206061878bffed3ad1f34d67c94fa32c3d33eac8
(3) 0xca67021a16478270ede4fddd65d0c031c75cd36c13b6a56bcb767928c1c2cf86
(4) 0x9955b1e01b2a7d8c22df41754d48b08dff3c0f3dd79d43e091c6311f97f0605a
(5) 0x130137aa9a7fbc7cadc98c079cda47a999ff41931d9feaab621855beceed71f7
(6) 0xead83d04f741d2b3ab50be1299c18aa1a82c241606861a9a6d3122443496522d
(7) 0xe6e893ac9f1c1db066a8a83a376554084b0a786e4cdcd91559d68bd4a1dac396
(8) 0xf1023ac6c8695f6ceb5331a382be8846bfe078b22c18ad7ef4fc3ea6e1cc59e4
(9) 0x4aef59c2cf29479b2c27a5f208e6b89d65d16f4977988151e135460db8274fdb
```

شکل ۷- ۱٦: شبیه‌ساز Embarc

لیست ۷- ۱۰: Embark simulator

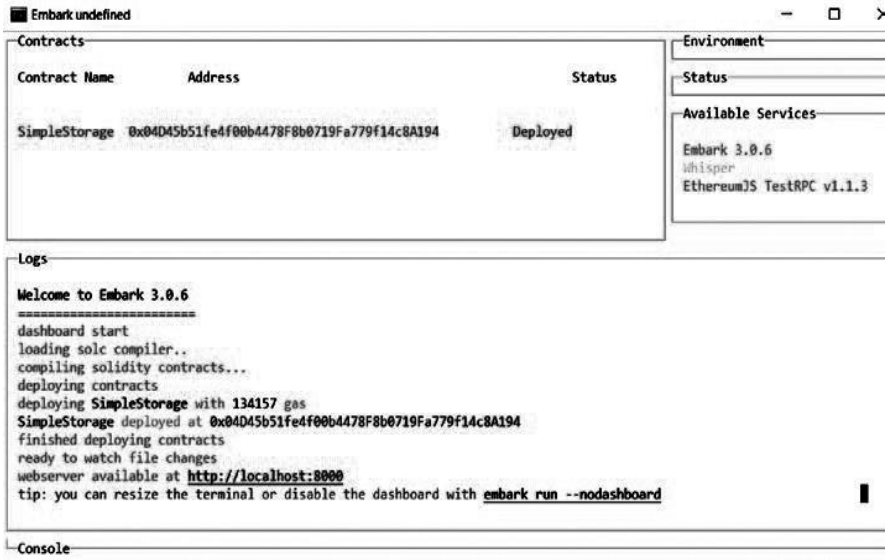
```
F:\ethereum\embark\embark_demo>embark simulator
Ganache CLI v6.1.8 (ganache-core: 2.2.1)
Available Accounts
==================
(0) 0xb8d851486d1c953e31a44374aca11151d49b8bb3 (~100 ETH)
(1) 0xf6d5c6d500cac10ee7e6efb5c1b479cfb789950a (~100 ETH)
(2) 0xf09324e7a1e2821c2f7a4a47675f9cf0b1a5eb7f (~100 ETH)
(3) 0xfbaf82a227dcebd2f9334496658801f63299ba24 (~100 ETH)
```

همچنین، می‌توانید موارد زیر را برای اجرای یک‌گره واقعی اجرا کنید:

```
embark blockChain
```

همان‌طور که در شکل ۷-۱۷ و لیست ۷-۳ نشان داده شده است، در پنجره‌ی دیگری از همان پوشه، کد زیر را اجرا کنید، کنسول را مشاهده خواهید کرد:

```
embark run
```

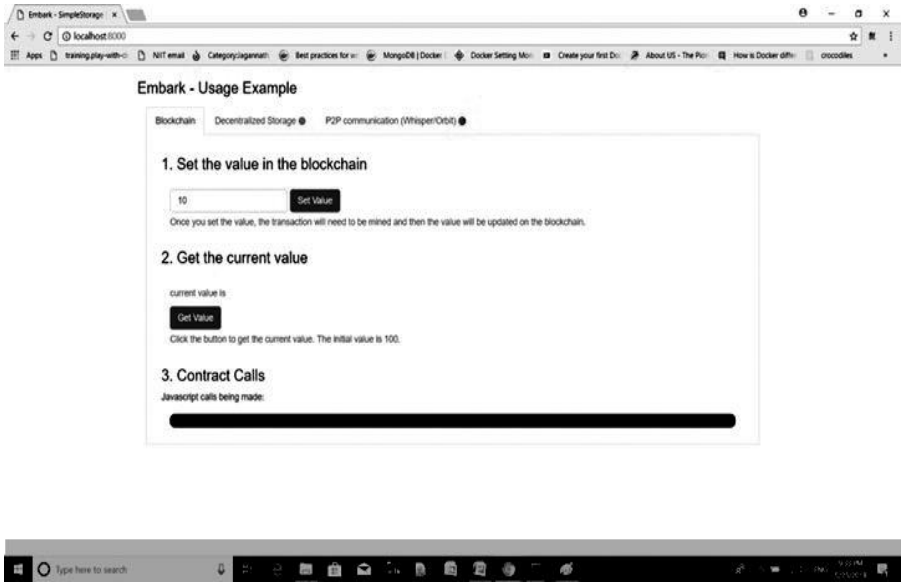


شکل ۷– ۱۷: اجرای Running embark.

لیست ۷–۱۱: Running embark run

```
deploying contracts
|
| deploying SimpleStorage with 134157 gas |
| SimpleStorage deployed at
0x04D45b51fe4f00b4478F8b0719Fa779f14c8A194 |
| finished deploying contracts |
| ready to watch file changes |

| webserver available at http://localhost:8000
```

اکنون سرور به‌صورت محلی روی پورت ۸۰۰۰ در حال اجرا است. http://localhost:8000 را در مرورگر باز کرده و کنسول وب را همان‌طور که در شکل ۷–۱۸ نشان داده شده است بررسی می‌کنیم.



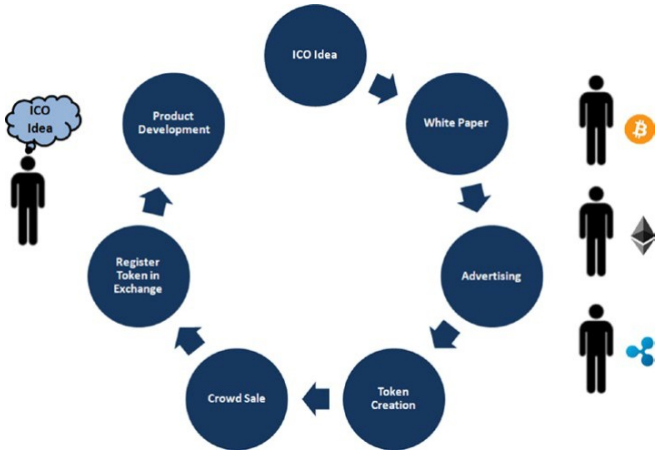
شکل ۷- ۱۸: Embark running on local

فصل هشتم

تکنیک‌ها و استراتژی‌های تست برنامه‌های توزیع‌شده‌ی اتریوم

فصل نهم

موارد استفاده از اتریوم



شکل ۹-۱: نقشه راه ICO

لیست ۹-۱: واسط ERC20.sol

```
pragma solidity ^0.4.0;
interface ERC20 {
    function totalSupply() constant returns
  (uint _totalSupply);
    function balanceOf(address _owner) constant returns
  (uint balance);
    function transfer(address _to, uint _value) returns
  (bool success);
    function transferFrom(address _from, address _to,
  uint _value) returns (bool success);
    function approve(address _spender, uint _value)
  returns (bool success);
    function allowance(address _owner, address _spender)
  constant returns (uint remaining);
    event Transfer(address indexed _from, address indexed _to,
  uint _value);
    event Approval(address indexed _owner, address indexed _
  spender, uint _value);
}
```

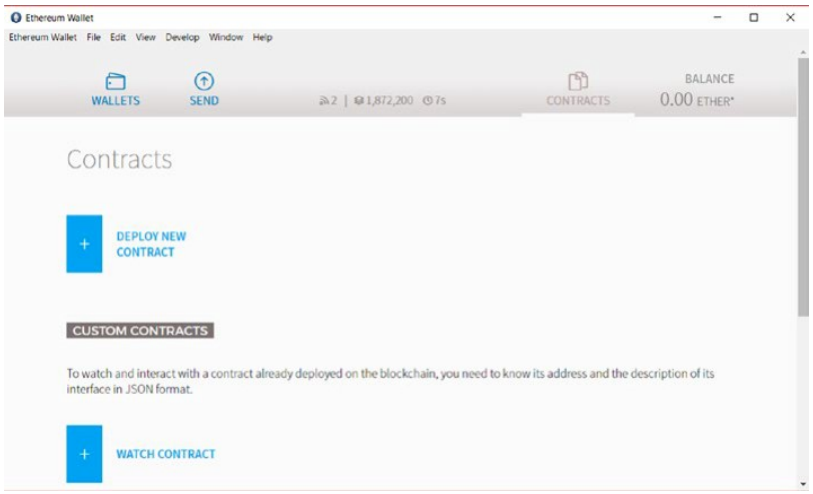
واسطِ MyFirstToken.sol را همان‌طور که در لیست ۹-۲ پیاده‌سازی آن نشان داده شده است، ایجاد می‌شود.

لیست ۹-۲ : MyFirstToken.sol

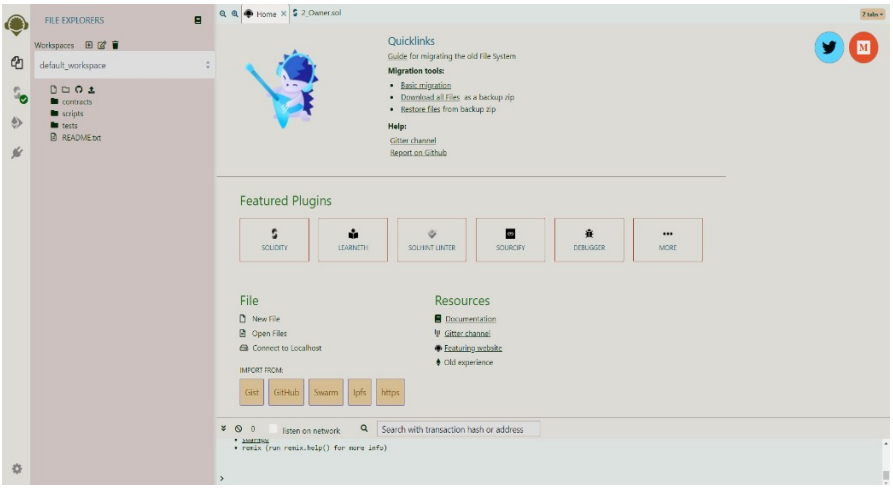
```
import "browser/ERC20.sol";
contract MyFirstToken is ERC20 {
    string public constant symbol = "DMY";
    string public constant name = "My First Token";
    uint8 public constant decimals = 18;
```

```
    uint private constant __totalSupply = 1000;
    mapping (address => uint) private __balanceOf;
    mapping (address => mapping (address => uint)) private __a
llowances;
    function MyFirstToken() {
            __balanceOf[msg.sender] = __totalSupply;
    }
    function totalSupply() constant returns (uint _totalSupply
) {
        _totalSupply = __totalSupply;
    }
    function balanceOf(address _addr) constant returns (uint b
alance) {
        return __balanceOf[_addr];
    }
    function transfer(address _to, uint _value) returns (bool
success) {
        if (_value > 0 && _value <= balanceOf(msg.sender)) {
            __balanceOf[msg.sender] -= _value;
            __balanceOf[_to] += _value;
            return true;
        }
        return false;
    }
    function transferFrom(address _from, address _to, uint _va
lue) returns (bool success) {
        if (__allowances[_from][msg.sender] > 0 &&
            _value > 0 &&
            __allowances[_from][msg.sender] >= _value &&
            __balanceOf[_from] >= _value) {
            __balanceOf[_from] -= _value;
            __balanceOf[_to] += _value;
             __allowances[_from][msg.sender] -= _value;
            return true;
        }
        return false;
    }
    function approve(address _spender, uint _value) returns (b
ool success) {
        __allowances[msg.sender][_spender] = _value;
        return true;
    }
    function allowance(address _owner, address _spender)
  constant returns (uint remaining) {
        return __allowances[_owner][_spender];
    }
}
```

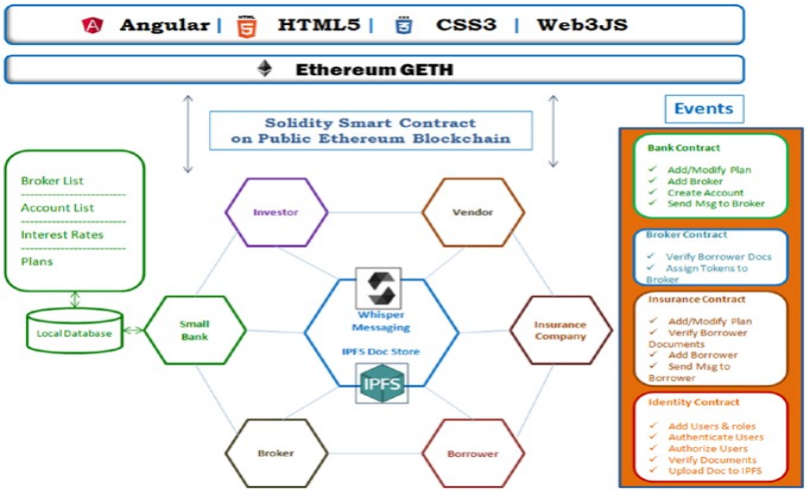
استقرار در Mist



شکل ۹-۲: کیف پول اتریوم مبتنی بر Mist



شکل ۹-۳: مرورگر remix



شکل ۹-٤: نمودار سیستم برای سرمایه‌گذاری خرد Dapp روی اتریوم

لیست ۹-۳: PropertyTransaction.sol

```
pragma solidity ^0.4.18;
contract PropertyTransaction {
    /**This contract is a basic land registry program where On
ly
    the contract initiator can assign properties to people may

be after
    some document verification*/
    Property[] properties; // registered properties
    address landRegistryAdmin; //landRegistryAdmin is the user

who has initiated the contract
    struct Property{
        uint propertyId; // each property has an unique
propertyId
        bytes32[] ownershipHistory; //Property ownership is a
historical data that has a seller, an owner/buyer and a
date of transaction
    }

    /**PropertySold is an event that informs on success or
failure of a transaction*/
    event PropertySold(uint propertyId,
                       string ownership,
                       bool flag,
                       string message);
    constructor() public {
        landRegistryAdmin = msg.sender; // initiate the
    landRegistryAdmin as the contract creator and initiate
    some registered properties
        Property memory property0 = Property(0, new bytes32[](
0));
        properties.push(property0);
```

```
        properties[properties.length-1].ownershipHistory.
    push("Buyer:b0, Seller:s0, DOT:dt0");
        Property memory property1 = Property(1, new bytes32[](
0));
        properties.push(property1);
        properties[properties.length-1].ownershipHistory.
    push("Buyer:b1, Seller:s1, DOT:dt1");
        Property memory property2 = Property(2, new bytes32[](
0));
        properties.push(property2);
        properties[properties.length-1].ownershipHistory.
    push("Buyer:b2, Seller:s2, DOT:dt2");
    }
    /**Land registration authority may alter ownership to any
  of the existing properties*/
    function addNewOwner(uint propertyId, bytes32 ownership)
  public {
    if (msg.sender != landRegistryAdmin) {
            emit PropertySold(propertyId,
                              bytes32ToString(ownership),
                              false,
                              'Only land registry department
              can assign ownership to buyer');
        }
        for (uint i = 0; i < properties.length; i++) {
            if (properties[i].propertyId == propertyId) {
                properties[i].ownershipHistory.push(ownership)
;
                emit PropertySold(propertyId,
                                  bytes32ToString(ownership),
                                  true,
                                  'New ownership added to existing

                property');
                break;
            }
        }
        // propertyId does not exist in record, hence create a

    new transaction and add to registered properties
        Property memory property = Property(properties.length,

    new bytes32[](0));
        properties.push(property);
        properties[properties.length-1].ownershipHistory.
    push(ownership);
        emit PropertySold(propertyId,
                                  bytes32ToString(ownership),
                                  true,
                                  'New Property added');
    }
    function retrievePropertyHistory(uint propertyId) public
  view returns(bytes32[]){
        for (uint i = 0; i < properties.length; i++) {
            if (properties[i].propertyId == propertyId) {
                return properties[i].ownershipHistory;
            }
```

```
        }
    }
    function bytes32ToString(bytes32 x) private pure returns (
string) {
        bytes memory bytesString = new bytes(32);
        uint charCount = 0;
        for (uint j = 0; j < 32; j++) {
            byte char = byte(bytes32(uint(x) * 2 ** (8 * j)));
            if (char != 0) {
                bytesString[charCount] = char;
                charCount++;
            }
        }
        bytes memory bytesStringTrimmed = new bytes(charCount)
;
        for (j = 0; j < charCount; j++) {
            bytesStringTrimmed[j] = bytesString[j];
        }
        return string(bytesStringTrimmed);
    }
}
```

ضمیمه

زبان برنامه‌نویسی Solidity

نصب کردن Node.js

ابتدا مطمئن شوید که Node.js را در دستگاه CentOS خود نصب دارید. اگر نصب نیست، با استفاده از دستورات زیر آن را نصب کنید.

```
# First install epel-release
$sudo yum install epel-release
# Now install nodejs
$sudo yum install nodejs
# Next install npm (Nodejs Package Manager)
$sudo yum install npm
# Finally verify installation
$npm --version
```

اگر مراحل را به‌درستی انجام داده باشید خروجی زیر را مشاهده خواهید کرد:

```
3.10.10
```

نصب کردن solc

پس از نصب Node.js، می‌توان با دستور زیر کامپایلر Solidity را نصب نمود.

```
$sudonpm install -g solc
```

دستور فوق، برنامه‌ی solcjs را نصب می‌کند و آن را از طریق سیستم در سراسر برنامه[1] در دسترس قرار می‌دهد. اکنون می‌توانید نسخه‌ی کامپایلر Solidity خود را با صدور دستور زیر بررسی کنید:

```
$solcjs -version
```

اگر همه چیز خوب پیش برود، با اجرای دستور بالا، در صفحه نمایش، خروجی زیر را مشاهده خواهید کرد:

```
0.5.2+commit.1df8f40c.Emscripten.clang
```

[1] Global Level

اکنون آماده‌ی استفاده از solcjs هستید که ویژگی‌های کمتری نسبت به کامپایلر استاندارد Solidity دارد اما نقطه‌ی شروع خوبی برای شما خواهد بود.

روش دوم: استفاده از Doker Image

برای شروع برنامه‌نویسی Solidity می‌توان از یک Docker Image استفاده کرد که در زیر مراحل آن بررسی می‌شود.

```
$docker pull ethereum/solc:stable
```

پس از بارگیری تصویر Docker، می‌توانیم آن را با استفاده از دستور زیر تأیید کنیم:

```
$docker run ethereum/solc:stable-version
```

با این کار خروجی زیر مشاهده می‌شود:

```
$docker run ethereum/solc:stable -version

solc, the Solidity compiler commandlineinterfaceVersion:
0.5.2+commit.1df8f40c.Linux.g++
```

روش سوم: نصب Binary Packages

در مثال زیر، پارامتر uintstoredData از نوع uint تعریف شده‌است و از دو تابع set و get برای دریافت، تغییر یا بازیابی مقدار متغیر استفاده می‌شود.

```
pragma Solidity >=0.4.0 <0.6.0;
contract SimpleStorage {
   uint storedData;
   function set(uint x) public {
      storedData = x;
   }
   function get() public view returns (uint) {
      return storedData;
   }
}
```

کلمات کلیدی

کلمات کلیدیِ رزروشده‌ی برنامه‌نویسیِ Solidity در جدول ۱ نشان داده شده است. توجه کنید که از کلمات کلیدی نمی‌توان برای تعریف متغیرها در قراردادهای هوشمند استفاده کرد.

جدول ۱: کلمات رزروشده در زبان برنامه‌نویسیِ Solidity.

abstract	after	alias	apply
auto	case	catch	copyof
defualt	define	final	immutable
implements	in	inline	let
macro	match	mutable	null
of	override	partial	promise
Reference	relocatable	sealed	sizeof
static	supports	switch	try
typedef	typeof	unchecked	public

اجرای اولین برنامه به زبان برنامه‌نویسیِ Solidity

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   constructor() public{
   }
   function getResult() public view returns(uint){
      uint a = 1;
      uint b = 2;
      uint result = a + b;
      return result;
   }
}
```

خروجی به شکل زیر نمایش داده خواهد شد:

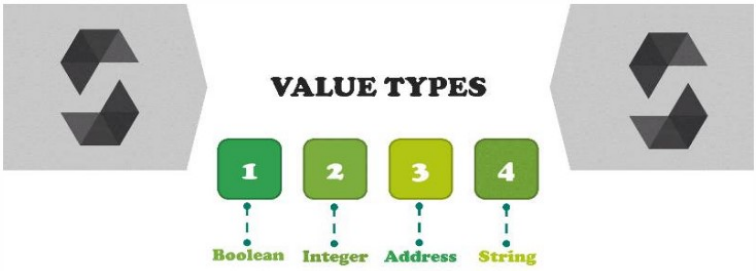
```
0: uint256: 3
```

Comments

```
function getResult() public view returns(uint){
   // This is a comment. It is similar to comments in C++
   /*
      * This is a multi-line comment in Solidity
      * It is very similar to comments in C Programming
   */
   uint a = 1;
```

```
uint b = 2;
uint result = a + b;
return result;}
```

نوع و ساختار داده‌ها



جدول ۲: انواع داده که در زبان برنامه‌نویسی Solidity پشتیبانی می‌شود.

نوع داده	کلمه کلیدی	مقدار
بولین	bool	True/false
عدد صحیح	int/uint	مجموع اعداد صحیح باعلامت و بدون-علامت
عدد صحیح	int8 to int256	مجموع اعداد باعلامت از ۸ بیت تا ۲۵۶ بیت
عدد صحیح	uint8 to uint256	مجموع اعداد بی‌علامت از ۸ بیت تا ۲۵۶ بیت
عدد اعشاری ثابت	fixed/unfixed	مجموع اعداد اعشاری ثابت با علامت و بدون علامت
عدد اعشاری شناور	fixed/unfixed	مجموع اعداد اعشاری شناور با و بدون علامت
عدد اعشاری ثابت	fixedMxN	عدد اعشاری با علامت اعشاری که M نشان‌دهنده‌ی تعداد بیت‌های گرفته‌شده بر اساس نوع داده و N نشان‌دهنده‌ی نقاط اعشاری است. M باید بر ۸ قابل تقسیم باشد و از ۸ به ۲۵۶ برسد. N می‌تواند از ۰ تا ۸۰ باشد. به‌عنوان‌مثال fixed128x18
	unifixedM×N	عدد اعشاری بی علامت اعشاری، که M نشان‌دهنده‌ی تعداد بیت‌های گرفته شده بر

عدد اعشاری ثابت		اساس نوع داده و N نشان‌دهنده‌ی نقاط اعشاری است. M باید بر ۸ قابل تقسیم باشد و از ۸ به ۲۵۶ برسد. N می‌تواند از ۰ تا ۸۰ باشد. به‌عنوان‌مثال fixed128x18
رشته	string	برای مشخص کردن آن از نمادهای " ، ' استفاده می‌شود.
آدرس	address	محل ذخیره‌سازی آدرس *اتریوم*

آدرس‌ها

متغیرها

۱.

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   uint storedData;       // State variable
   constructor() public {
      storedData = 10;    // Using State variable
   }
}
```

۲.

```
pragma Solidity ^0.5.0;
contract Solidity Test {
   uint storedData; // State variable
   constructor() public {
      storedData = 10;
   }
   function getResult() public view returns(uint){
      uint a = 1; // local variable
      uint b = 2;
      uint result = a + b;
      return result; //access the local variable
   }
}
```

۳.

```
pragma Solidity  ^0.5.0;
contract LedgerBalance {
   mapping(address => uint) public balances;

   function updateBalance(uint newBalance) public {
      balances[msg.sender] = newBalance;
   }
}
contract Updater {
```

```
    function updateBalance() public returns (uint) {
        LedgerBalance ledgerBalance = new LedgerBalance();
        ledgerBalance.updateBalance(10);
        return ledgerBalance.balances(address(this));
    }
}
```

جدول ۳: متغیرهای ویژه در فضای نامی سراسری

نام	مقداری را که بازمی‌گرداند
Blockhash(uint blockNumber) returns 32bytes	هش بلوک
block.coinbase (address payable)	آدرس استخراج کننده بلوک فعلی
block.difficulty (uint)	سختی بلوک فعلی
block.gaslimit (uint)	محدودیت gas بلوک فعلی
block.number (uint)	شماره بلوک فعلی
block.timestamp (uint)	مهر زمانی بلوک فعلی
gasleft() returns (uint256)	باقیمانده gasها
msg.data (bytes calldata)	کامل کردن داده‌های فراخوانی
msg.sender (address payable)	فرستنده پیام(فراخواننده کنونی)
msg.sig (bytes4)	چهاربایت اول فراخوانده داده (تابع شناسه)
msg.value (uint)	تعداد wei ارسال شده با یک پیام
now (uint)	مهرزمانی بلوک کنونی
tx.gasprice (uint)	مبلغ gas تراکنش
tx.origin (address payable)	فرستنده تراکنش

ساختارهای داده در زبان Solidity

آرایه‌ها

```
pragma Solidity  ^0.5.0;
contract test {
   function testArray() public pure{
      uint len = 7;

      //dynamic array
      uint[] memory a = new uint[](7);

      //bytes is same as byte[]
      bytes memory b = new bytes(len);

      assert(a.length == 7);
      assert(b.length == len);

      //access array variable
      a[6] = 8;

      //test array variable
      assert(a[6] == 8);

      //static array
      uint[3] memory c = [uint(1) , 2, 3];
      assert(c.length == 3);
   }
}
```

ساختمان داده

```
pragma Solidity ^0.5.0;

contract test {
   struct Book {
      string title;
      string author;
      uint book_id;
   }
   Book book;
```

```
    function setBook() public {
        book = Book('Learn Java', 'TP', 1);
    }
    function getBookId() public view returns (uint) {
        return book.book_id;
    }
}
```

تعریف آرایه‌ای از یک struct در برنامه‌نویسی Solidity بسیار مفید است زیرا همانند یک پایگاه داده در زنجیره‌ی بلوکی اتریوم عمل می‌کند. برای این‌که بتوانیم این موضوع را تشریح کنیم، ابتدا ساختمان داده‌ای به نام Person که شامل دو داده‌ی age و name به‌ترتیب از نوع عدد صحیح بی علامت و رشته است را ایجاد کرده و سپس از آن یک آرایه‌ی عمومی به نام people طراحی شده که در ادامه کد آن را مشاهده می‌کنید.

```
struct Person {
uint age;
string name;
}
Person [] public people;
```

اکنون می‌خواهیم یک Person جدید ایجاد کرده و به آرایه‌ی people اضافه کنیم. ابتدا یک Person جدید به نام satoshi ساخته و آن را مقداردهی می‌کنیم. در ادامه به کمک دستور push آن را به آرایه‌ی خود اضافه می‌کنیم.

```
// create a New Person:
Person satoshi =Person (172, "Satoshi");
// Add that person to the Array:
people.push(satoshi)
```

البته می‌توان این دو مرحله را باهم ترکیب و آن را به شکل زیر بازنویسی کرد:

```
people.push(Person(16,"Vitalik"));
```

نگاشت

```
pragma Solidity  ^0.5.0;
contract LedgerBalance {
    mapping(address => uint) public balances;
```

```
   function updateBalance(uint newBalance) public {
      balances[msg.sender] = newBalance;
   }
}
contract Updater {
   function updateBalance() public returns (uint) {
      LedgerBalance ledgerBalance = new LedgerBalance();
      ledgerBalance.updateBalance(10);
      return ledgerBalance.balances(address(this));
   }
}
```

خروجی برنامه‌ی فوق به‌صورت زیر خواهد بود:

```
   "0": "uint256: 10"
```

برای درک بهترِ نحوه‌ی استفاده از نگاشت‌ها، برنامه‌ی زیر را در نظر بگیرید:

```
pragma Solidity  0.5.1;
contract MyContract {
   uint256 peopleCount = 2;
   mapping (uint => Person) public people;
   struct Person{
   uint _id;
   string _firstName;
   string _lastName;
   {
   function addPerson (string memory _firstName, string memory
   _lastName) public {
   peopleCount += 1;
   people[peopleCount] =Person (peopleCount, _firstName, _lastN
   ame);
   }}
   }
```

Enums

```
pragma Solidity  ^0.5.0;
contract test {
   enum FreshJuiceSize{ SMALL, MEDIUM, LARGE {
   FreshJuiceSize choice;
   FreshJuiceSize constant defaultChoice = FreshJuiceSize.MEDI
UM;
   function setLarge() public {
      choice = FreshJuiceSize.LARGE;
   }
   function getChoice() public view returns (FreshJuiceSize) {
      return choice;
   }
   function getDefaultChoice() public pure returns (uint) {
      return uint(defaultChoice);
   }
```

```
}
```

خروجی برنامه زیر به‌صورت زیر است:

```
uint8: 2
uint256: 1
```

تعیین سطوح دسترسی به متغیرها[1]

```
pragma Solidity  ^0.5.0;
contract C {
   uint public data = 30;
   uint internal iData= 10;
     function x() public returns (uint) {
      data = 3; // internal access
      return data;
   }
}
contract Caller {
   C c = new C();
   function f() public view returns (uint) {
      return c.data(); //external access
   }
}
contract D is C {
   function y() public returns (uint) {
      iData = 3; // internal access
      return iData;
   }
   function getResult() public view returns(uint){
      uint a = 1; // local variable
      uint b = 2;
      uint result = a + b;
      return storedData; //access the state variable
   }}
```

عملگرهای زبان Solidity

[1] Variable Scope

برای درک بهتر نحوه‌ی کار با عملگرها، برنامه زیر را مرور کنید.

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   uint storedData;
   constructor() public{
      storedData = 10;
   }
   function getResult() public view returns(string memory){
      uint a = 1; // local variable
      uint b = 2;
      uint result = (a > b? a: b);  //conditional operation
      return integerToString(result);
   }
   function integerToString(uint _i) internal pure
      returns (string memory) {
      if (_i == 0) {
         return "0";
      }
      uint j = _i;
      uint len;
      while (j != 0) {
         len++;
         j /= 10;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;
      while (_i != 0) {
         bstr[k--] = byte(uint8(48 + _i % 10));
         _i /= 10;
      }
      return string(bstr);
   }
}
```

خروجی برنامه برابر است با:

```
0: string: 2
```

حلقه‌ها و عبارت‌های شرطی

.

حلقه‌ها

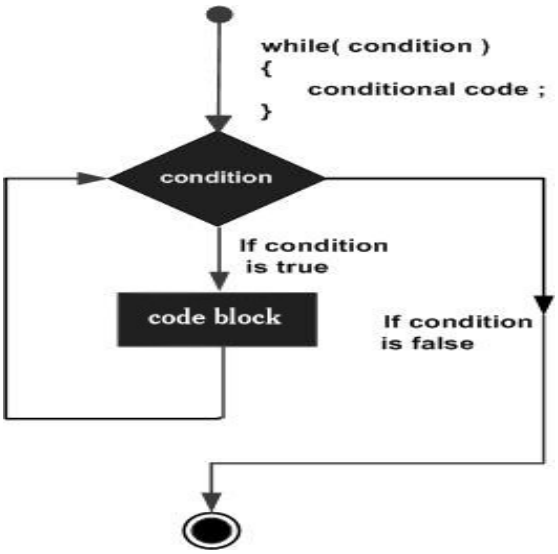
- حلقه‌ی while: دستور while به‌صورت زیر نوشته می‌شود:

```
while (expression) {
   Statement(s) to be executed if expression is true
}
```

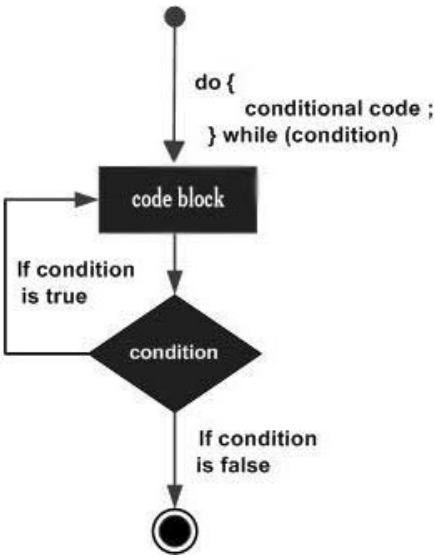
در ادامه برای درک بهتر عملکرد while کد زیر را بررسی کنید. که خروجی برنامه، عدد ۱۲ خواهد بود.

```
pragma Solidity  ^0.5.0;

contract Solidity Test {
   uint storedData;
   constructor() public{
      storedData = 10;
   }
   function getResult() public view returns(string memory){
      uint a = 10;
      uint b = 2;
      uint result = a + b;
      return integerToString(result);
   }
   function integerToString(uint _i) internal pure
      returns (string memory) {

      if (_i == 0) {
         return "0";
      }
      uint j = _i;
      uint len;

      while (j != 0) {
         len++;
         j /= 10;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;

      while (_i != 0) { // while loop
         bstr[k--] = byte(uint8(48 + _i % 10));
         _i /= 10;
      }
      return string(bstr);
   }
}
```

فلوچارت حلقه while به‌صورت زیر است:



- حلقه‌ی do-while: فلوچارت این حلقه در شکل زیر نشان داده شده است.



دستور do-while به‌صورت زیر استفاده می‌شود:

```
do {
   Statement(s) to be executed;
```

```
} while (expression);
```

برای درک بهتر عملکرد حلقه‌ی do-while، برنامه‌ی زیر که خروجی ۱۲ را تولید می‌کند بررسی کنید:

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   uint storedData;
   constructor() public{
      storedData = 10;
   }
   function getResult() public view returns(string memory){
      uint a = 10;
      uint b = 2;
      uint result = a + b;
      return integerToString(result);
   }
   function integerToString(uint _i) internal pure
      returns (string memory) {

      if (_i == 0) {
         return "0";
      }
      uint j = _i;
      uint len;

      while (j != 0) {
         len++;
         j /= 10;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;

      do {                    // do while loop
         bstr[k--] = byte(uint8(48 + _i % 10));
         _i /= 10;
      }
      while (_i != 0);
      return string(bstr);
   }
}
```

- حلقه‌ی for: برای استفاده از حلقه‌ی for باید از دستور زیر استفاده کرد:

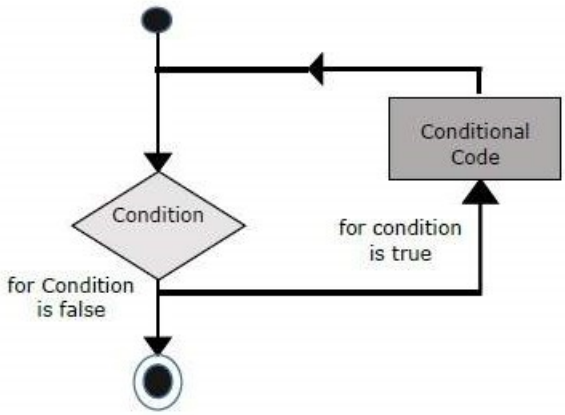
```
for (initialization; test condition; iteration statement){
   Statement(s) to be executed if test condition is true
}
```

در ادامه مثال قبلی را با کمک دستور for بازنویسی می‌کنیم:

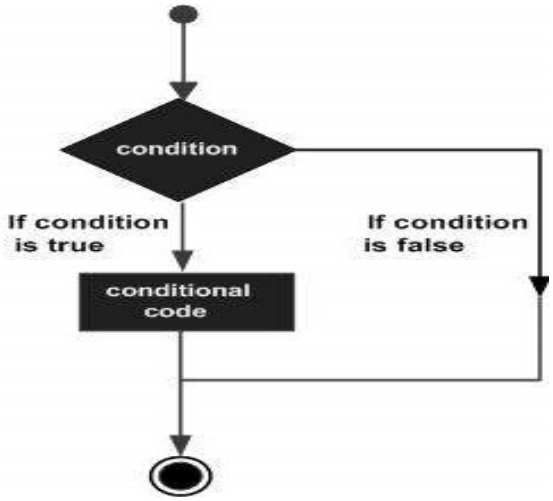
```
pragma Solidity  ^0.5.0;
```

```
contract Solidity Test {
   uint storedData;
   constructor() public{
      storedData = 10;
   }

   function getResult() public view returns(string memory){
      uint a = 10;
      uint b = 2;
      uint result = a + b;
      return integerToString(result);
   }

   function integerToString(uint _i) internal pure
      returns (string memory) {
      if (_i == 0) {
         return "0";
      }
      uint j=0;
      uint len;
      for (j = _i; j != 0; j /= 10) {  //for loop example
         len++;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;
      while (_i != 0) {
         bstr[k--] = byte(uint8(48 + _i % 10));
         _i /= 10;
      }
      return string(bstr);//access local variable}
   }
```

فلوچارت حلقه‌ی for در شکل زیر نشان داده شده است:



عبارت‌های شرطی



در جدول ٤ نحوه‌ی استفاده از عبارت‌های شرطی بررسی شده است.

جدول ٤: عبارت شرطی

`if (expression) {`
`  Statement(s) to be executed if expression is true`
`}`
	if

`if (expression) {`
`  Statement(s) to be executed if expression is true`
`} else {`
`  Statement(s) to be executed if expression is false`
`}`
	if…else

`if (expression 1) {`
`  Statement(s) to be executed if expression 1 is true`
`} else if (expression 2) {`
`  Statement(s) to be executed if expression 2 is true`
`} else if (expression 3) {`
`  Statement(s) to be executed if expression 3 is true`
`} else {`
`  Statement(s) to be executed if no expression is true`
`}`
	if...else if

در ادامه به ترتیب سه برنامه‌ی زیر را بررسی کنید تا نحوه‌ی استفاده از این عبارات شرطی را به‌طور کامل در زبان برنامه‌نویسی Solidity بیاموزید.

- مثال اول: دستور `if` (خروجی برنامه عدد ۳ است).

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   uint storedData;
   constructor() public {
      storedData = 10;
   }
   function getResult() public view returns(string memory){
      uint a = 1;
      uint b = 2;
      uint result = a + b;
      return integerToString(result);
   }
   function integerToString(uint _i) internal pure
      returns (string memory) {
      if (_i == 0) {   // if statement
         return "0";
      }
      uint j = _i;
      uint len;
            while (j != 0) {
         len++;
         j /= 10;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;

      while (_i != 0) {
         bstr[k--] = byte(uint8(48 + _i % 10));
         _i /= 10;
      }
      return string(bstr);//access local variable
   }
}
```

- مثال دوم دستور `else...if` (خروجی برنامه عدد ۲ است).

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   uint storedData;
   constructor() public{
      storedData = 10;
   }
   function getResult() public view returns(string memory){
      uint a = 1;
      uint b = 2;
      uint result
      if( a > b) {   // if else statement
         result = a;
      } else {
         result = b;
      }
      return integerToString(result);
   }
   function integerToString(uint _i) internal pure
      returns (string memory) {
      if (_i == 0) {
         return "0";
```

```
      }
      uint j = _i;
      uint len;

      while (j != 0) {
         len++;
         j /= 10;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;

      while (_i != 0) {
         bstr[k--] = byte(uint8(48 + _i % 10));
         _i /= 10;
      }
      return string(bstr);//access local variable
   }}
```

- مثال سوم: if...else if (خروج برنامه عدد ۳ است).

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
   uint storedData; // State variable
   constructor() public {
      storedData = 10;
   }
   function getResult() public view returns(string memory) {
      uint a = 1;
      uint b = 2;
      uint c = 3;
      uint result

      if( a > b && a > c) {    // if else statement
         result = a;
      } else if( b > a && b > c ){
         result = b;
      } else {
         result = c;
      }
      return integerToString(result);
   }
   function integerToString(uint _i) internal pure
      returns (string memory) {

      if (_i == 0) {
         return "0";
      }
      uint j = _i;
      uint len;

      while (j != 0) {
         len++;
         j /= 10;
      }
      bytes memory bstr = new bytes(len);
      uint k = len - 1;
```

```
        while (_i != 0) {
            bstr[k--] = byte(uint8(48 + _i % 10));
            _i /= 10;
        }
        return string(bstr);//access local variable}
```

توابع در زبان Solidity

```
function function-name(parameter-list) scope returns() {
    //statements
}
```

تابع getResult که دارای پارامتر ورودی نیست و خروجی آن یک مقدار صحیح بی‌علامت است در برنامه‌ی زیر تعریف شده است:

```
pragma Solidity  ^0.5.0;
contract Test {
    function getResult() public view returns(uint){
        uint a = 1; // local variable
        uint b = 2;
        uint result = a + b;
        return result;
    }
}
```

فراخوانی یک تابع

برای فراخوانی یک تابع در هر نقطه از قرارداد، به‌سادگی باید نام آن تابع را همان‌طور که در کد زیر نشان داده شده است، بنویسید و فراخوانی کنید. کد زیر را اجرا کنید تا نحوه‌ی کار رشته در Solidity را درک کنید.

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
    constructor() public{
    }
    function getResult() public view returns(string memory){
        uint a = 1;
        uint b = 2;
        uint result = a + b;
        return integerToString(result);
    }
    function integerToString(uint _i) internal pure
        returns (string memory) {

        if (_i == 0) {
            return "0";
        }
        uint j = _i;
```

```
        uint len;

        while (j != 0) {
            len++;
            j /= 10;
        }
        bytes memory bstr = new bytes(len);
        uint k = len - 1;

        while (_i != 0) {
            bstr[k--] = byte(uint8(48 + _i % 10));
            _i /= 10;
        }
        return string(bstr);//access local variable
    }
}
```

آرگومان‌های ورودی تابع

```
pragma Solidity  ^0.5.0;
contract Solidity Test {
    constructor() public{
    }
    function getResult() public view returns(string memory){
        uint a = 1;
        uint b = 2;
        uint result = a + b;
        return integerToString(result);
    }
    function integerToString(uint _i) internal pure
        returns (string memory) {

        if (_i == 0) {
            return "0";
        }
        uint j = _i;
        uint len;

        while (j != 0) {
            len++;
            j /= 10;
        }
        bytes memory bstr = new bytes(len);
        uint k = len - 1;

        while (_i != 0) {
            bstr[k--] = byte(uint8(48 + _i % 10));
            _i /= 10;
        }
        return string(bstr);//access local variable
    }
}
```

آرگومان خروجی

```
function (<parameter types>) {internal|external} [pure|constan
t|view|payable] [returns (<return types>)]
pragma Solidity  ^0.5.0;
contract Test {
   function getResult() public view returns(uint product, uint
 sum){
      uint a = 1; // local variable
      uint b = 2;
      product = a * b;
      sum = a + b;
   }
}
```

خروجی برنامه زیر به‌صورت زیر است:

```
0: uint256: product 2
1: uint256: sum 3
```

اصلاح‌کننده‌ی توابع[1]

```
contract Owner {
   modifier onlyOwner {
      require(msg.sender == owner);
      _;
   }
   modifier costs(uint price) {
      if (msg.value >= price) {
         _;
      }
   }
}
```

بدنه‌ی تابع درجایی که نماد ویژه‌ی «_;» قرار گرفته است، وارد می‌شود. بنابراین، اگر در هنگام فراخوانی این تابع شرط اصلاح‌کننده برآورده شود، تابع اجرا می‌شود و درغیراین صورت، یک استثنا throw می‌شود. برای درک بهتر، کد زیر را بررسی کنید:

```
pragma Solidity  ^0.5.0;
contract Owner {
   address owner;
   constructor() public {
      owner = msg.sender;
   }
   modifier onlyOwner {
      require(msg.sender == owner);
      _;
   }
   modifier costs(uint price) {
      if (msg.value >= price) {
         _;
      }
```

[1] Function modifiers

```
    }
}
contract Register is Owner {
    mapping (address => bool) registeredAddresses;
    uint price;
    constructor(uint initialPrice) public { price = initialPric
e; }

    function register() public payable costs(price) {
        registeredAddresses[msg.sender] = true;
    }
    function changePrice(uint _price) public onlyOwner {
        price = _price;
    }
}
```

کد داخل اصلاح‌کننده‌ی توابع معمولا قبل از متن اصلی تابع اجرا می‌شود، بنابراین هر تغییر وضعیت یا فراخوانی *خارجی* باعث اختلال و نقض الگوی تعاملات بررسی‌ها و تاثیرات خواهد شد. به علاوه، ممکن است توسعه‌دهنده متوجه این گزاره‌ها نشود، زیرا کد اصلاح‌کننده‌ی تابع می‌تواند فاصله‌ی زیادی با فرض تابع داشته باشد.

توابع view

```
pragma Solidity >=0.4.0 <0.6.0;
contract SimpleStorage {
    uint storedData;
    function set(uint x) public {
        storedData = x;
    }
    function get () public view returns (uint) {
        return storedData;
    }
   function get () public view returns (string memory)}
     return value;
    }
}
```

در ادامه مثالی از یک تابع `view` مشاهده می‌شود:

```
pragma Solidity  ^0.5.0;

contract Test {
    function getResult() public view returns(uint product, uint
      sum){
       uint a = 1; // local variable
       uint b = 2;
       product = a * b;
       sum = a + b;
    }
}
```

توابع pure

```
pragma Solidity ^0.5.0;
contract Test {
   function getResult() public pure returns(uint product, uint
 sum){
      uint a = 1;
      uint b = 2;
      product = a * b;
      sum = a + b;
   }
}
```

خروجی برنامه برابر است با:

```
0: uint256: product 2
1: uint256: sum 3
```

تابع Fallback

.

```
pragma Solidity  ^0.5.0;
contract Test {
   uint public x ;
   function() external { x = 1; }
}
contract Sink {
   function() external payable { }
}
contract Caller {
   function callTest(Test test) public returns (bool) {
      (bool success,) = address(test).call(abi.encodeWithSigna
ture("nonExistingFunction()"));
      require(success);
      // test.x is now 1

      address payable testPayable = address(uint160(address(te
st)));
      // Sending ether to Test contract,
      // the transfer will fail, i.e. this returns false here.
      return (testPayable.send(2 ether));
   }
   function callSink(Sink sink) public returns (bool) {
      address payable sinkPayable = address(sink);
      return (sinkPayable.send(2 ether));
   }
}
```

تابع overloading

```
pragma Solidity  ^0.5.0;

contract Test {
   function getSum(uint a, uint b) public pure returns(uint){
      return a + b;
   }
   function getSum(uint a, uint b, uint c) public pure returns
(uint){
      return a + b + c;
   }
   function callSumWithTwoArguments() public pure returns(uint
){
      return getSum(1,2);
   }
   function callSumWithThreeArguments() public pure returns(ui
nt){
      return getSum(1,2,3);
   }
}
```

خروجی برنامه به‌صورت زیر نمایش داده خواهد شد:

```
0: uint256: 3
0: uint256: 6
```

توابع ریاضی

```
pragma Solidity  ^0.5.0;
contract Test {
   function callAddMod() public pure returns(uint){
      return addmod(4, 5, 3);
   }
   function callMulMod() public pure returns(uint){
      return mulmod(4, 5, 3);
   }
}
```

خروجی:

```
0: uint256: 0
0: uint256: 2
```

توابع رمزنگاری

در ادامه برخی از مهم‌ترین توابع پیش‌فرض و تعریف شده در کتابخانه‌ی Solidity نام برده شده است. در تمام این توابع هدف، به‌دست‌آوردن هشِ ورودی به کمک توابع هش مختلفی است.

- `keccak256(bytes memory) returns(bytes32)`
- `sha256(bytes memory) returns(bytes32)`
- `ripemd160(bytes memory) returns(bytes20)`
- `sha256(bytes memory) returns(bytes32)`

- `ecrecover(bytes32 hash, uint8 v, bytes32 r, bytes32 s) returns (address):`

به‌عنوان مثال، آخرین تابع، آدرس مربوط به کلید عمومی را از امضای منحنی بیضی‌گون بازیابی می‌کند یا صفر را در هنگام خطا بازمی‌گرداند. پارامترهای تابع با مقادیر ECDSA امضا مطابقت دارند. نمونه‌ای از برنامه که از این توابع رمزنگاری استفاده شده است در برنامه‌ی زیر آمده است:

```
pragma Solidity  ^0.5.0;
contract Test {
   function callKeccak256() public pure returns(bytes32 result
){
      return keccak256("ABC");
   }
}
```

خروجی:

```
0: bytes32: result
0xe1629b9dda060bb30c7908346f6af189c16773fa148d3366701fbaa35d54
f3c8
```

قرارداد

```
pragma Solidity ^0.5.0;
contract Base {
   uint data;
   constructor(uint _data) public {
      data = _data;
   }
}
contract Derived is Base (5) {
   constructor() public {}
}
```

سازنده‌ی پایه می‌تواند به روش غیرمستقیم زیر نیز راه‌اندازی شود:

```
pragma Solidity  ^0.5.0;
contract Base {
   uint data;
   constructor(uint _data) public {
      data = _data;
   }
}
contract Derived is Base {
   constructor(uint _info) Base(_info * _info) public {}
}
```

قراردادهای انتزاعی یا abstract

.

```
pragma Solidity  ^0.5.0;
contract Calculator {
   function getResult() public view returns(uint);
}
contract Test is Calculator {
   function getResult() public view returns(uint) {
      uint a = 1;
      uint b = 2;
      uint result = a + b;
      return result;
   }
}
```

میدان دید یا پدیداری کمیت‌ها

```
pragma Solidity  ^0.5.0;
contract C {
   //private state variable
   uint private data;
     //public state variable
   uint public info;

   //constructor
   constructor() public {
      info = 10;
   }
   //private function
   function increment(uint a) private pure returns(uint) { ret
urn a + 1; }
      //public function
   function updateData(uint a) public { data = a; }
   function getData() public view returns(uint) { return data;
 }
   function compute(uint a, uint b) internal pure returns (uin
t) { return a + b; }
}
//External Contract
contract D {
   function readData() public returns(uint) {
      C c = new C();
      c.updateData(7);
      return c.getData();
   }
}
//Derived Contract
contract E is C {
   uint private result;
   C private c;
      constructor() public {
```

```
      c = new C();
   }
   function getComputedResult() public {
      result = compute(3, 5);
   }
   function getResult() public view returns(uint) { return res
ult; }
   function getData() public view returns(uint) { return c.inf
o(); }
}
```

وراثت

```
pragma Solidity  ^0.5.0;
contract C {
   //private state variable
   uint private data;
     //public state variable
   uint public info;
   //constructor
   constructor() public {
      info = 10;
   }
   //private function
   function increment(uint a) private pure returns(uint) { ret
urn a + 1; }
      //public function
   function updateData(uint a) public { data = a; }
   function getData() public view returns(uint) { return data;
 }
   function compute(uint a, uint b) internal pure returns (uin
t) { return a + b; }
}
//Derived Contract
contract E is C {
   uint private result;
   C private c;
   constructor() public {
      c = new C();
   }
   function getComputedResult() public {
      result = compute(3, 5);
   }
   function getResult() public view returns(uint) { return res
ult; }
   function getData() public view returns(uint) { return c.inf
o(); }
}
```

رابط‌ها یا Interface

```
pragma Solidity  ^0.5.0;
interface Calculator {
```

```
   function getResult() external view returns(uint);
}
contract Test is Calculator {
   constructor() public {}
   function getResult() external view returns(uint){
      uint a = 1;
      uint b = 2;
      uint result = a + b;
      return result;
   }
}
```

رویدادها یا Events

```
//Declare an Event
event Deposit(address indexed _from, bytes32 indexed _id, uint
 _value);
//Emit an event
emit Deposit(msg.sender, _id, msg.value);
```

برای آن‌که نحوه‌ی عملکرد *رویداد* را متوجه شویم ابتدا قراردادی را ایجاد و یک *رویداد* را منتشر می‌کنیم.

```
pragma Solidity  ^0.5.0;
contract Test {
   event Deposit(address indexed _from, bytes32 indexed _id, u
int _value);
   function deposit(bytes32 _id) public payable {
      emit Deposit(msg.sender, _id, msg.value);
   }
}
```

در زبان برنامه‌نویسی جاوا اسکریپت یک *رویداد* قرارداد به روش زیر قابل دسترسی است.

```
var abi = /* abi as generated using compiler */;
var ClientReceipt = web3.eth.contract(abi);
var clientReceiptContract = ClientReceipt.at("0x1234...ab67" /
* address */);

var event = clientReceiptContract.Deposit(function(error, resu
lt) {
   if (!error)console.log(result);
})
```

که خروجی زیر را به ما نشان خواهد داد.

```
{
   "returnValues": {
      "_from": "0x1111...FFFFCCCC",
      "_id": "0x50...sd5adb20",
      "_value": "0x420042"
```

```
    },
    "raw": {
        "data": "0x7f...91385",
        "topics": ["0xfd4...b4ead7", "0x7f...1a91385"]
    }
}
```

مدیریت خطا

```
pragma Solidity  ^0.5.0;
contract Vendor {
    address public seller;
    modifier onlySeller() {
        require(
            msg.sender == seller,
            "Only seller can call this."
        );
        _;
    }
    function sell(uint amount) public payable onlySeller {
        if (amount > msg.value / 2 ether)
            revert("Not enough Ether provided.");
        // Perform the sell operation.
    }
}
```

بررسی چند مثال عملی

```
pragma Solidity >=0.5.0 <0.6.0;
contract ZombieFactory {

  event NewZombie (uint zombieId, string name, uint dna);
  uint dnaDigits = 16;
  uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }
    Zombie[] public zombies;

function _createZombie(string memory _name, uint _dna) private
 {
        uint id = zombies.push(Zombie(_name, _dna)) - 1;
        emit NewZombie(id, _name, _dna);
    }

function _generateRandomDna(string memory _str) private view r
eturns    (uint) {


        uint rand = uint(keccak256(abi.encodePacked(_str)));
        return rand % dnaModulus;
```

1 2 3 4 5 6 7

```
    }

function createRandomZombie(string memory _name) public {
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }
}
```

مثال ۲

```
pragma Solidity  >=0.5.0 <0.6.0;

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }

    Zombie[] public zombies;                                  1

    mapping (uint => address) public zombieToOwner;
    mapping (address => uint) ownerZombieCount;

    function _createZombie(string memory _name, uint _dna) pri
vate {
        uint id = zombies.push(Zombie(_name, _dna)) - 1;


        zombieToOwner[id] = msg.sender;      2
        ownerZombieCount[msg.sender]++;
        emit NewZombie(id, _name, _dna);
    }

    function _generateRandomDna(string memory _str) private vi
ew returns (uint) {
        uint rand = uint(keccak256(abi.encodePacked(_str)));
        return rand % dnaModulus;
    }

    function createRandomZombie(string memory _name) public {

                                                      3
require(ownerZombieCount[msg.sender] == 0);
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }
                                          4
}
contract ZombieFeeding is ZombieFactory {
}
```

۱. یکی از انواع داده‌ای که در Solidity از آن پشتیبانی می‌شود آدرس و نگاشت است که امکان ایجاد یک زوج *کلید-مقدار* را می‌دهد. برای ذخیره‌ی مالکیت زامبی ها، از دو نگاشت استفاده می‌شود: یکی از نوع آدرس، که آدرس مالک آن را ثبت کند و دیگری که تعداد زامبی‌های یک مالک را ثبت کند.

۲. اکنون که به کمک نوع داده‌ی نگاشت توانسته‌ایم مالک زامبی‌ها را تعیین کنیم. زمان آن رسیده است با تغییر تابع `_createZombie` از آن‌ها استفاده کنیم. همانطور که بررسی شد در زبان برنامه‌نویسی Solidity *متغیرسراسری* به نام `msg.sender` وجود دارد که قابل دسترسی برای همه‌ی توابع هست. این متغیر به آدرس *قرارداد هوشمندی* که تابع فعلی را فراخوانی می‌کند اشاره دارد.

توجه: در Solidity، اجرای تابع همیشه باید با فراخوانی *خارجی* شروع شود. یک قرارداد فقط روی زنجیره‌ی بلوکی قرار می‌گیرد و هیچ‌کاری انجام نمی‌دهد تا این‌که قراردادی یکی از تابع‌های آن را فراخوانی کند. بنابراین، همیشه یک `msg.sender` وجود خواهد داشت. در اینجا هدف اختصاص مالکیت زامبی به کسی است که تابع `_createZombie` را فراخوانی کرده است. ابتدا `msg.sender` را در `id` ذخیره کرده و سپس باید تعداد زامبی-های مالک را برای این `msg.sender` افزایش بدهیم.

۳. در مثال۱، کاربران قادر بودند تا با فراخوانی `createRandomZombie` و وارد کردن یک رشته، زامبی‌های جدید ایجاد کنند. در این بخش می‌خواهیم کاری کنیم که هرکسی فقط یک بار بتواند این تابع را فراخوانی کند. برای این کار باید از دستور `require` استفاده کنیم. تا در صورت عدم صحت شرط، تابع خطایی را ایجاد کرده و اجرا را متوقف کند.

٤. وراثت: برای جلوگیری از طولانی شدن *قراردادهای هوشمند* گاهی منطقی است که منطق کد خود را به چندین قرارداد تقسیم کنیم تا بهتر بتوان *قرارداد هوشمند* را سازماندهی کرد. برای این منظور باید از وراثت یا `inheritance` استفاده کرد. که برای این کار باید قرارداد مشتق شده را به

شکل contract ZombieFeeding is ZombieFactory {} فراخوانی کرد که بیانگر این است قرارداد ZombieFeeding از قرارداد ZombieFactory مشتق شده است، به این معناست که با اجرای قرارداد ZombieFeeding به کلیه‌ی توابع در قرارداد ZombieFactory دسترسی دارد.

```
pragma Solidity  >=0.5.0 <0.6.0;
import "./zombiefactory.sol";

contract ZombieFeeding is ZombieFactory {

function feedAndMultiply(uint _zombieId, uint _targetDna) publ
ic {

require(msg.sender == zombieToOwner[_zombieId]);
    Zombie storage myZombie = zombies[_zombieId];
    _targetDna = _targetDna % dnaModulus;
    uint newDna = (myZombie.dna + _targetDna) / 2;
    _createZombie("NoName", newDna);
  }
}
```

۱. وارد کردن یا import: اگر یک کد بسیارطولانی شود می‌توان آن را به چندین فایل تقسیم کرد و سپس آن را فراخوانی کرد. برای این منظور باید از دستور import استفاده کرد.

۲. محل ذخیره‌سازی داده (storage در مقابل memory). در Solidity، می‌توانید متغیرها را در Storage و memory ذخیره کنید. Storage برای ذخیره‌سازی دائمی متغیرها در زنجیره‌ی بلوکی استفاده می‌شود. این در حالی است که memory حافظه‌ی موقتی برای ذخیره‌سازی متغیرها است و بین فراخوانی‌های *خارجی* قرارداد هوشمند پاک می‌شوند در واقع عملکرد آن مشابه RAM رایانه است. بیشتر اوقات نیازی به استفاده از این کلمات کلیدی نیست زیرا Solidity به‌طور پیش‌فرض آن‌ها را کنترل می‌کند. متغیرهای وضعیت (متغیرهایی که خارج از توابع اعلام می‌شوند) به‌صورت پیش‌فرض

در Storage ذخیره می‌شوند و برای همیشه در زنجیره‌ی بلوکی نوشته می‌شوند، در حالی که متغیرهای اعلام شده در داخل توابع در memory هستند و با پایان دادن به فراخوانی تابع از بین می‌روند. با این حال، مواردی وجود دارد که شما مجبور به استفاده از این کلمات کلیدی هستید، به‌عنوان مثال در هنگام استفاده از struct و آرایه‌ها در داخل توابع.

۳. تابع جدیدی را ایجاد می‌کنیم که با ترکیب DNA میزبان و زامبی، یک زامبی جدید ایجاد شود. این تابع عمومی دارای دو آرگومان ورودی است. ابتدا به کمک require باید چک شودکه فرد دیگری زامبی ما را تغذیه نکند. اکنون می‌خواهیم که DNA زامبی را دریافت کنیم. ابتدا یک متغیر محلی به نام myZombie از ساختمان داده Zombie ایجاد کرده و آن را با آرایه‌ای به نام zombies که نمایه‌ی آن _zombieId است مساوی قرار داده‌ایم. در اینجا چون از ساختمان داده در تابع استفاده شده است باید از کلمه‌ی کلیدی storage نیز استفاده شود.

٤. محاسبه DNA زامبی جدید ساده است، تنها کافی است که متوسط DNAهای میزبان و زامبی را محاسبه کرد. برای دسترسی به dna و name مربوط به myZombie کافی است آن را با *myZombie.name* و *myZombie.dna* فراخوانی کنید. سپس تابع _createZombie را فراخوانی می‌کنیم که زامبی جدید ایجاد شود.

```
pragma Solidity  >=0.5.0 <0.6.0;

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;

    struct Zombie {
        string name;
        uint dna;
    }

    Zombie[] public zombies;

    mapping (uint => address) public zombieToOwner;
    mapping (address => uint) ownerZombieCount;
```

1

```
function _createZombie(string memory _name, uint _dna) internl
 {

 uint id = zombies.push(Zombie(_name, _dna)) - 1;
        zombieToOwner[id] = msg.sender;
        ownerZombieCount[msg.sender]++;
        emit NewZombie(id, _name, _dna);
    }

    function _generateRandomDna(string memory _str) private vi
ew returns (uint) {
        uint rand = uint(keccak256(abi.encodePacked(_str)));
        return rand % dnaModulus;
    }

    function createRandomZombie(string memory _name) public {
        require(ownerZombieCount[msg.sender] == 0);
        uint randDna = _generateRandomDna(_name);
        _createZombie(_name, randDna);
    }

}
```

مثال ۳- رابط یا Interface و دستورات شرطی

```
pragma Solidity  >=0.5.0 <0.6.0;
import "./zombiefactory.sol";

contract KittyInterface {
  function getKitty(uint256 _id) external view returns (
    bool isGestating,
    bool isReady,
    uint256 cooldownIndex,
    uint256 nextActionAt,
    uint256 siringWithId,
    uint256 birthTime,
    uint256 matronId,
    uint256 sireId,
    uint256 generation,
    uint256 genes
  );
}
contract ZombieFeeding is ZombieFactory {

  address ckAddress = 0x06012c8cf97BEaD5deAe237070F9587f8E7A26
6d;
  KittyInterface kittyContract = KittyInterface(ckAddress);

  function feedAndMultiply(uint _zombieId, uint _targetDna, st
ring memory _species) public {
    require(msg.sender == zombieToOwner[_zombieId]);
    Zombie storage myZombie = zombies[_zombieId];
    _targetDna = _targetDna % dnaModulus;
    uint newDna = (myZombie.dna + _targetDna) / 2;
```

```
     if (keccak256(abi.encodePacked(_species)) ==  keccak256(a
bi.encodePacked("kitty"))) {
       newDna = newDna - newDna % 100 + 99;
     }
     _createZombie("NoName", newDna);
   }

   function feedOnKitty(uint _zombieId, uint _kittyId) public {
     uint kittyDna;
     (,,,,,,,,,kittyDna) = kittyContract.getKitty(_kittyId);

     feedAndMultiply(_zombieId, kittyDna, "kitty");
   }
```

4

5

مثال ٤

```
pragma Solidity  >=0.5.0 <0.6.0;

import "./ownable.sol";

contract ZombieFactory is Ownable {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;
    uint cooldownTime = 1 days;

    struct Zombie {
      string name;
      uint dna;
      uint32 level;
      uint32 readyTime;
    }

    Zombie[] public zombies;

    mapping (uint => address) public zombieToOwner;
    mapping (address => uint) ownerZombieCount;

    function _createZombie(string memory _name, uint _dna) int
ernal {

    uint id = zombies.push(Zombie(_name, _dna, 1, uint32(now +
 cooldownTime))) - 1;

        zombieToOwner[id] = msg.sender;
        ownerZombieCount[msg.sender]++;
        emit NewZombie(id, _name, _dna);
    }

    function _generateRandomDna(string memory _str) private vi
ew returns (uint) {
        uint rand = uint(keccak256(abi.encodePacked(_str)));
        return rand % dnaModulus;
```

1

2

3

4

```
    }

    function createRandomZombie(string memory _name) public {
        require(ownerZombieCount[msg.sender] == 0);
        uint randDna = _generateRandomDna(_name);
        randDna = randDna - randDna % 100;
        _createZombie(_name, randDna);
    }

}
```

۱.فایل owner.sol را import می‌کنیم تا از ویژگی‌ها آن در قرارداد خود استفاده کنیم (ارث بری).

۲.Ownable Contract: در مرحله‌ی قبل با تعریف اعلان تابع setKittyContractAddress به‌صورت اعلان external، این امکان فراهم می‌شود که هر فردی قادر به فراخوانی تابع و تغییر آدرس قرارداد CryptoKitties باشد که خود یک نقطه‌ی ضعف برای تامین امنیت *قرارداد هوشمند* است. به همین علت، نه تنها هدف استفاده از توانایی به‌روزرسانی این آدرس است، بلکه، نمی‌خواهیم همه‌ی افراد این دسترسی را داشته باشند. یکی از راه حل‌ها و روش‌های معمول برای اغنای این شرایط استفاده از owanable contract ها است که امتیازات ویژه‌ای را فراهم می‌کند.

۳.Gas: در Solidity ، کاربران هر بار که تابعی را در DApp شما اجرا می‌کنند، باید ارزی به نام gas را پرداخت کنند. کاربران gas را با Ether (واحد پولی *اتریوم*) خریداری می‌کنند، بنابراین کاربران برای اجرای توابع در DApp شما مجبور به خرج کردن اتر هستند. اینکه چه مقدار gas برای اجرای یک تابع مورد نیاز است، بستگی به پیچیده بودن منطق آن تابع دارد. هر عملیات، جداگانه هزینه gas مبتنی بر مقدار منابع محاسباتی مورد نیاز برای انجام آن عملیات، محاسبه می‌کند (به‌عنوان مثال نوشتن در storage بسیار گران‌تر از افزودن دو عدد صحیح است). از آنجا که توابع در حال اجرا برای کاربران هزینه‌ی واقعی دارد، بهینه‌سازی کد در *اتریوم* بسیار مهم‌تر از سایر زبان‌های برنامه‌نویسی است. اگر کد شما بهینه نباشد، کاربران مجبورند برای اجرای عملکردهای شما حق بیمه بپردازند و این می‌تواند به میلیون‌ها دلار هزینه‌ی

غیر ضروری برای هزاران کاربر تبدیل گردد. *اتریوم* مانند یک کامپیوتر بزرگ، کند، اما بسیار امن است. هنگام اجرای یک تابع، تک تک گره‌های شبکه برای تأیید خروجیِ خود، باید همان تابع را اجرا کنند، این دقیقا ویژگی‌ای است که باعث می‌شود *اتریوم* غیرمتمرکز باشد، و داده‌های آن غیرقابل تغییر و در برابر تغییرات مقاوم هستند. هدف سازندگان *اتریوم* آن است که اطمینان حاصل کنند که کسی نمی‌تواند شبکه را با یک حلقه‌ی نامحدود مسدود کند، یا تمام منابع شبکه را با محاسبات واقعاً فشرده مسدود نماید. بنابراین، آن‌ها تراکنش‌ها را شاملِ هزینه کردند تا معاملات رایگان نباشند و کاربران مجبور شوند هزینه-ی زمان محاسبه و همچنین فضای مورد نیاز برای ذخیره‌سازی را پرداخت کنند. برای این منظور به‌جای استفاده از داده‌ای که ۲۵٦ بیت را برای ذخیره‌سازی استفاده می‌کند از ۳۲ بیت برای ذخیره‌سازی مهرزمانی و سطح استفاده شده است.

٤. ویژگی سطح یا `level` قابل توضیح است. زیرا زمانی که یک سیستم نبرد ایجاد شود، زامبی‌هایی که در نبردهای بیشتری پیروز می‌شوند، با گذشت زمان سطح بالاتری پیدا می‌کنند و به توانایی‌های بیشتری دسترسی پیدا می‌کنند. اما بررسی ویژگی `readyTime` اندکی به توضیح بیشتری نیاز دارد. به کمک این ویژگی می‌خواهیم فرصتی را ایجاد کنیم که هر زامبی پس از تغذیه و یا حمله، مدت زمانی را برای نوبت بعدیِ تغذیه و حمله صبر کند. ازاین رو از واحدهای زمانی که در زبان برنامه‌نویسی **Solidity** وجود دارد استفاده می‌شود. متغیر زمانی `now`، مهرزمانی یونیکس آخرین بلوک فعلی را باز می-گرداند. زبان برنامه‌نویسی **Solidity** همچنین شامل واحدهای زمانی ثانیه، دقیقه، ساعت، روز، هفته و سال است. برای ایجاد این فاصله‌ی زمانی از یک `cooldownTime` یک روزه استفاده شده است. از آن‌جا که ساختمان داده‌ی ما به‌روز رسانی شده و دارای دو داده‌ی جدید شده است، باید تابع `_creatZombie` و نیز آرایه‌ای که از ساختمان داده `Zombie` ساخته شده است، به‌روز رسانی شود. همچنین دو آرگومان `level` و `readyTime`

باید مقدار دهی شود که به‌ترتیب با ۱ و uint32(now + cooldownTime) مقدار دهی شده است.

در ادامه باید بخشی از تغییرات لازم را در فایل zombiefeeding.sol ایجاد کنیم.

```
pragma Solidity  >=0.5.0 <0.6.0;

import "./zombiefactory.sol";

contract KittyInterface {
  function getKitty(uint256 _id) external view returns (
    bool isGestating,
    bool isReady,
    uint256 cooldownIndex,
    uint256 nextActionAt,
    uint256 siringWithId,
    uint256 birthTime,
    uint256 matronId,
    uint256 sireId,
    uint256 generation,
    uint256 genes
  );
}

contract ZombieFeeding is ZombieFactory {

  KittyInterface kittyContract;

  function setKittyContractAddress(address _address) external
onlyOwner {
    kittyContract = KittyInterface(_address);
  }

  function _triggerCooldown(Zombie storage _zombie) internal {

    _zombie.readyTime = uint32(now + cooldownTime);
  }


function isReady(Zombie storage _zombie) internal view returns
 (bool) {
      return (_zombie.readyTime <= now);
  }

   function feedAndMultiply(uint _zombieId, uint _targetDna, st
ring memory _species) internal {
    require(msg.sender == zombieToOwner[_zombieId]);
    Zombie storage myZombie = zombies[_zombieId];
    require(_isReady(myZombie));
    _targetDna = _targetDna % dnaModulus;
    uint newDna = (myZombie.dna + _targetDna) / 2;
```

1

2

3

```
    if (keccak256(abi.encodePacked(_species)) == keccak256(abi
.encodePacked("kitty"))) {
      newDna = newDna - newDna % 100 + 99;
    }
    _createZombie("NoName", newDna);
    _triggerCooldown(myZombie);
  }

  function feedOnKitty(uint _zombieId, uint _kittyId) public {
    uint kittyDna;
    (,,,,,,,,,kittyDna) = kittyContract.getKitty(_kittyId);
    feedAndMultiply(_zombieId, kittyDna, "kitty");
  }
}
```