Three new Ganache features to improve your developer experience

ByKingsley Arinze

Since the release of Ganache v7.0 in January, we've mostly focused our attention on bug fixes and UX improvements. Recently, we decided to expand our focus to also include new features that will help improve the developer experience for our users.

In this post, we'll touch on 3 of those new features that we've added to Ganache since the Ganache v7.0 release.

Zero-config mainnet forking now available in the browser¶

Ganache v7.0 made it possible to run Ganache in the browser. Simply add this script to your HTML:

< script src = "https://cdn.jsdelivr.net/npm/ganache@7.0.0/dist/web/ganache.min.js"

</ script

And Ganache is available in your browser for use with:

const

options

=

{}; const

provider

=

Ganache . provider (options); Adding the above lines of code would make the Ganache blockchain simulator available in your browser. However, zero-config mainnet forking wasn't yet available.

Since the release of Ganache v7.3.2, users can now utilize this feature in the browser by specifying additional options during setup:

< script src = "https://cdn.jsdelivr.net/npm/ganache@7.3.2/dist/web/ganache.min.js"

</ script

const

options

=

{

fork :

{

network :

"mainnet"

}

} const

provider

=

Ganache . provider (options); You can verify that this is working by attempting to fetch a mainnet block using theeth_getBlockByNumber method:

```
const
block
=
await
provider . request ({
method :
"eth_getBlockByNumber" ,
params :
[ "0xec4eb0" ]
}); console . log ( block );
// will be mainnet's block 15486640
```

Ability to use console.log from Solidity¶

If you use Vyper'sprint statement or Hardhat'sconsole.sol library contract for writing values to stdout, you can now use these tools together with Ganache, as Ganache now understands Vyper'sprint statement, as well as Hardhat'sconsole.sol library!

Development of nativeconsole.log support in Truffle is currently in progress, but Truffle users don't have to wait to use this feature, as they can install Ganache'sconsole.log library withnpm install @ganache/console.log and use it in their Solidity project:

```
// SPDX-License-Identifier: MIT pragma solidity
^ 0.8.13 ; import
"@ganache/console.log/console.sol" ; contract
GanacheCoin
{
mapping
( address
=>
uint )
balances;
constructor ()
{
balances[ tx.origin ]
=
10000 ;
}
function
sendCoin ( address
receiver ,
uint256
amount )
public
returns ( bool
sufficient )
{
if
( balances[ msg.sender ]
    =
amount)
{
balances[ msg.sender ]
-=
amount;
balances[ receiver]
+=
amount;
console. log( "send: %o coins to %o" ,
amount,
receiver);
return
true ;
}
else
{
return
false ;
}
} }
```

New eth_getProof RPC method¶

Ganache now supports theeth_getProof RPC method, as specified byEIP-1186 . This method returns some useful values, including hashes and proofs associated with the given address and storage keys. This method can be used as shown below:

```
const
result
=
await
provider . request ({
"method" :
"eth_getProof" ,
"params" :
[
// the address to query and generate proof for
"0x7F0d15C7FAae65896648C8273B6d7E43f58Fa842" ,
// optional storage keys to generate proofs for
[ "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421" ],
// the block at which to generate the proof
"latest"
] }); It should return the following output:
{
"accountProof" :
[
"0xf90211a...0701bc80" ,
"0xf90211a...0d832380" ,
"0xf90211a...5fb20c80" ,
"0xf90211a...0675b80" ,
"0xf90151a0...ca08080"
],
"balance" :
"0x0" ,
"codeHash" :
"0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470" ,
"nonce" :
"0x0" ,
"storageHash" :
"0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421" ,
"storageProof" :
[
{
"key" :
"0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421" ,
"proof" :
[
"0xf90211a...0701bc80" ,
"0xf90211a...0d832380"
],
"value" :
```

"0x1"

}

] }

Conclusion¶

At Truffle, we remain committed to improving and simplifying the user experience for dapp developers in the Web3 ecosystem by creating developer tools, resources, and educational materials.

To find out more about our suite of developer tools, visit the officialTruffle website . If you have questions, feel free to start a discussion on ourGithub Discussions channel . We also hold weekly live streamed sessions calledWeb Unleashed , where we build, interview folks, and discuss important developments around the ecosystem. Keep an eye on our Twitter for updates on the next session. You can also find past episodes on theTruffle Youtube channel and theunleashed section of our website if you prefer written materials.