

Web3: Smart Wallets in React

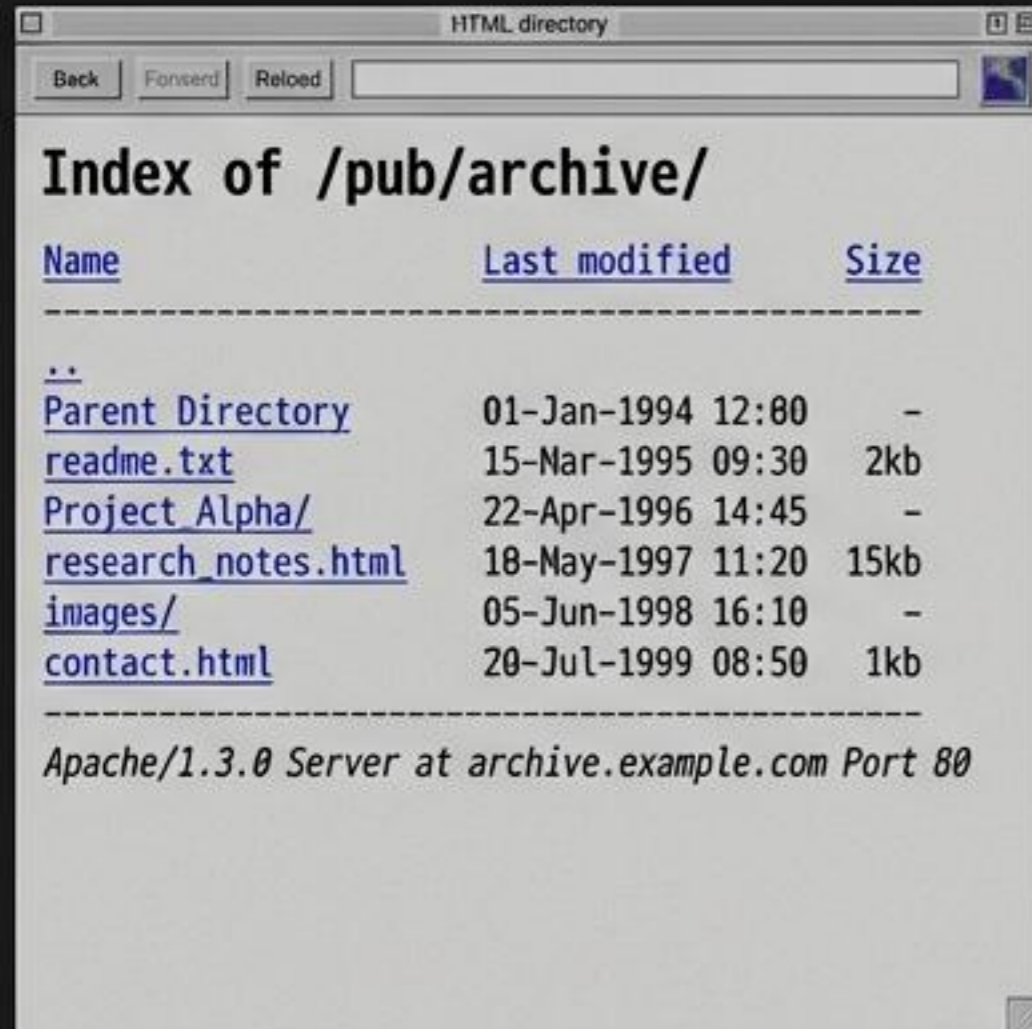
The evolution of ownership, the conflict of wallet standards, and the code that connects them.

George Michoulis

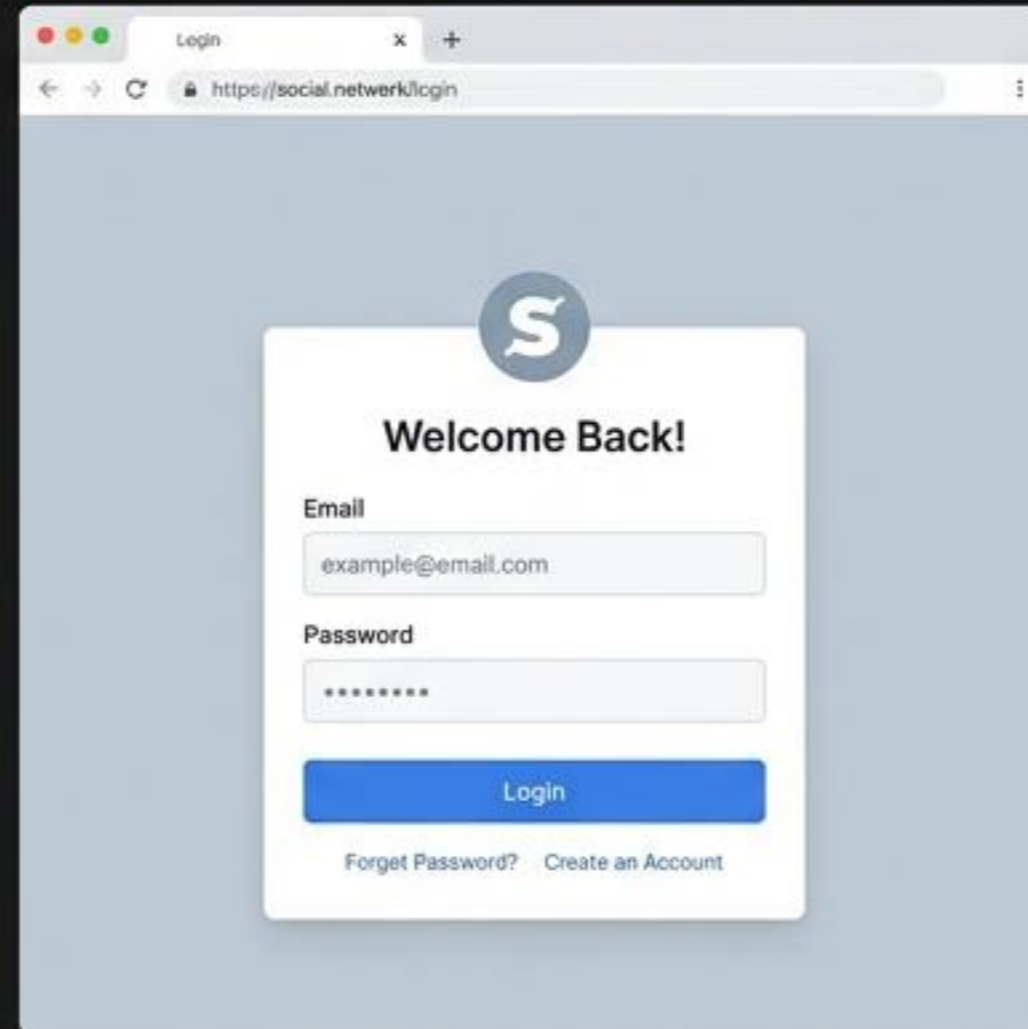
MSc Data & Web Science



EVOLUTION OF USER INTERFACE



Web1: Read



Web2: Read-Write

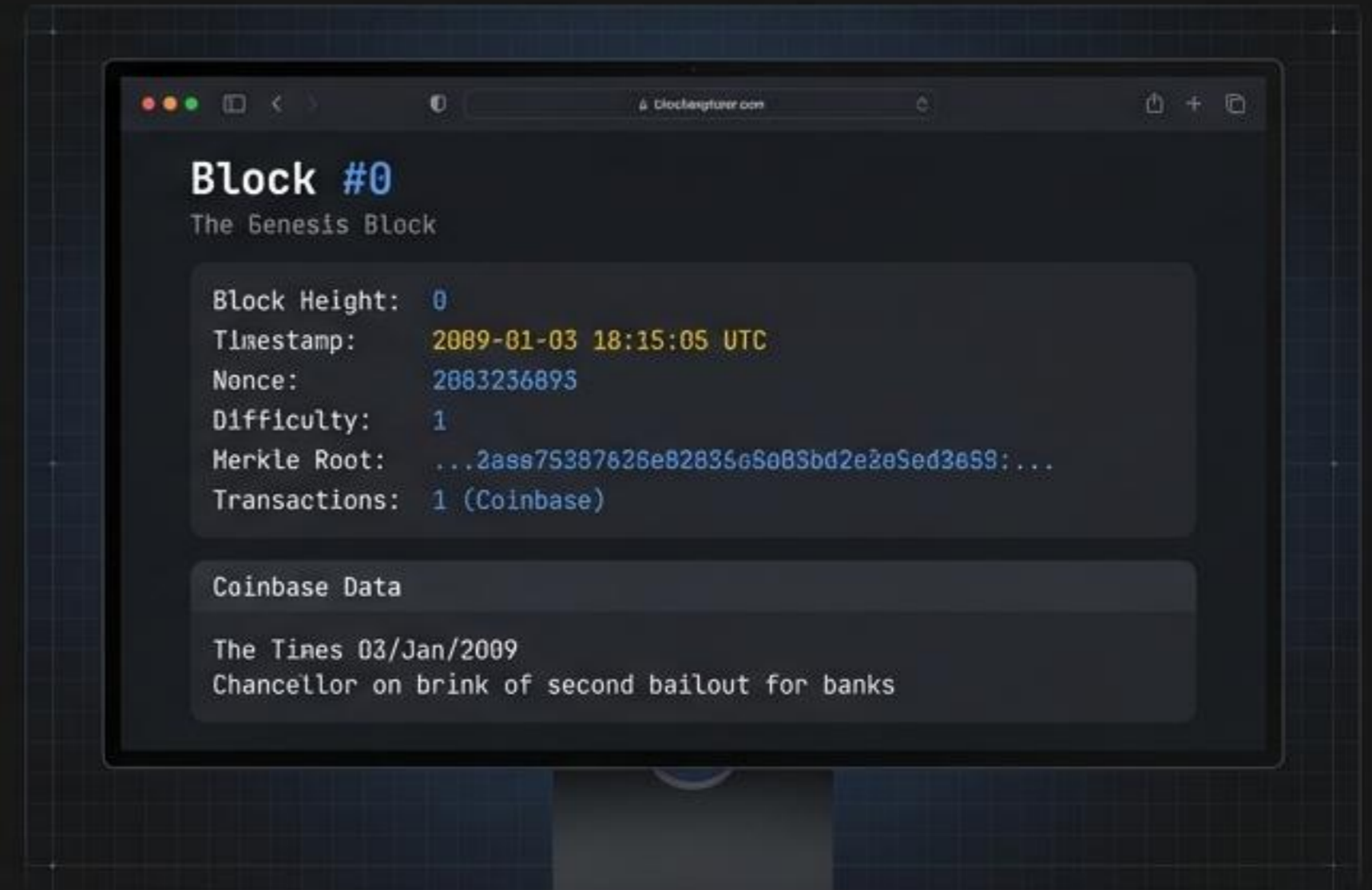


Web3: Read-Write-Own

From static protocols to centralized silos, to user sovereignty.

The Spark: 2009

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution.



Trustless Architecture

- Digital Scarcity
- Cryptographic Proof

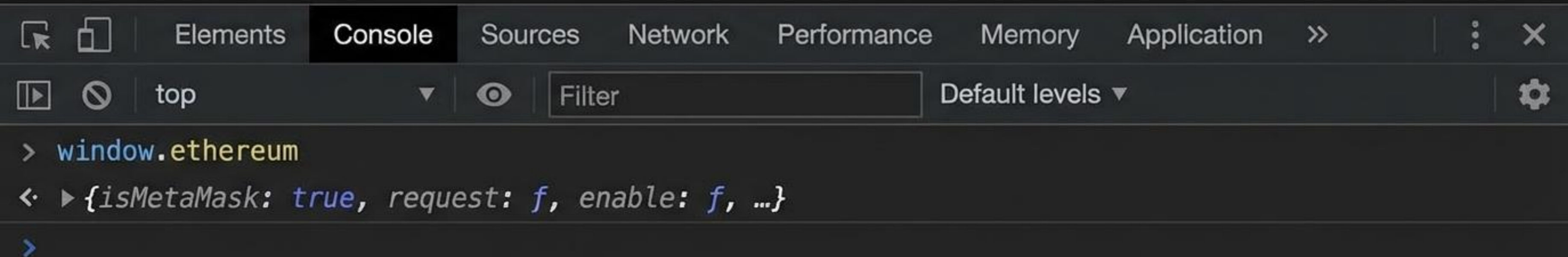
The Passport, Not the Vault



Misconception: Wallets store assets.
Reality: Wallets store Private Keys.
• Function: The Signer.

The Injection

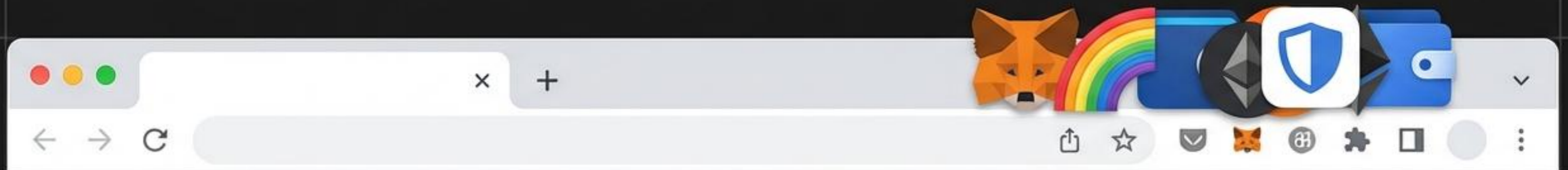
The Bridge: How the browser meets the blockchain via the injected Global Object.



The Wallet Wars: Clobbering

Race Conditions. One global object. Zero choice.

The system broke by its own success.



```
window.ethereum.request({ method: 'eth_requestAccounts' });  
const provider = window.ethereum;  
console.error(provider);  
// Output: Uncaught Error: Multiple providers detected.
```

EIP-6963 GitHub proposal

Proposal: JetBrains Mono
Publication: EIP-6963

Description

This Multi injected provider discovery is event event listeners that hom object to are itersiy the basis of EIP-6963 proposal.

The multi injected provider discovery in the non-injected proposal must to convert up this provider; need to return the provider objects and the provider can connect an identifier to the provider. The respect to get the key information from any information and is an event listener. When the network is compared to develop the sacker in the event tag attached or a proposal.

EIP-6963

JetBrains: Provider Discovery

Multi Injected Provider Discovery

EIP-6963 proposes a multi injected provider discovery mechanism that allows the provider to connect an identifier to the provider enabling to event listeners.

The content proposed is:

Key Insight:

The Peace Treaty: Moving from global objects to event listeners.

The Peace Treaty: Moving from global objects to event listeners.

The Peace Treaty is: Moving from global objects to event listeners. The main idea is the event provider discovery it event there is a way as a state. The main idea in the proposal is convert to the event listener from global provider to the proposal is a result.

The Handshake

Bidirectional communication. Every installed wallet identifies itself with a UUID, Name, and Icon.

dApp Request

```
window.dispatchEvent(new Event('eip6963:requestProvider'));
```



Wallet Announce

```
window.addEventListener('eip6963:announceProvider', (event) => {  
  ...  
});
```

True User Choice

No conflicts. Just options.
The dApp displays exactly
what the user has installed.

Connect Wallet



MetaMask



Rainbow



Trust Wallet



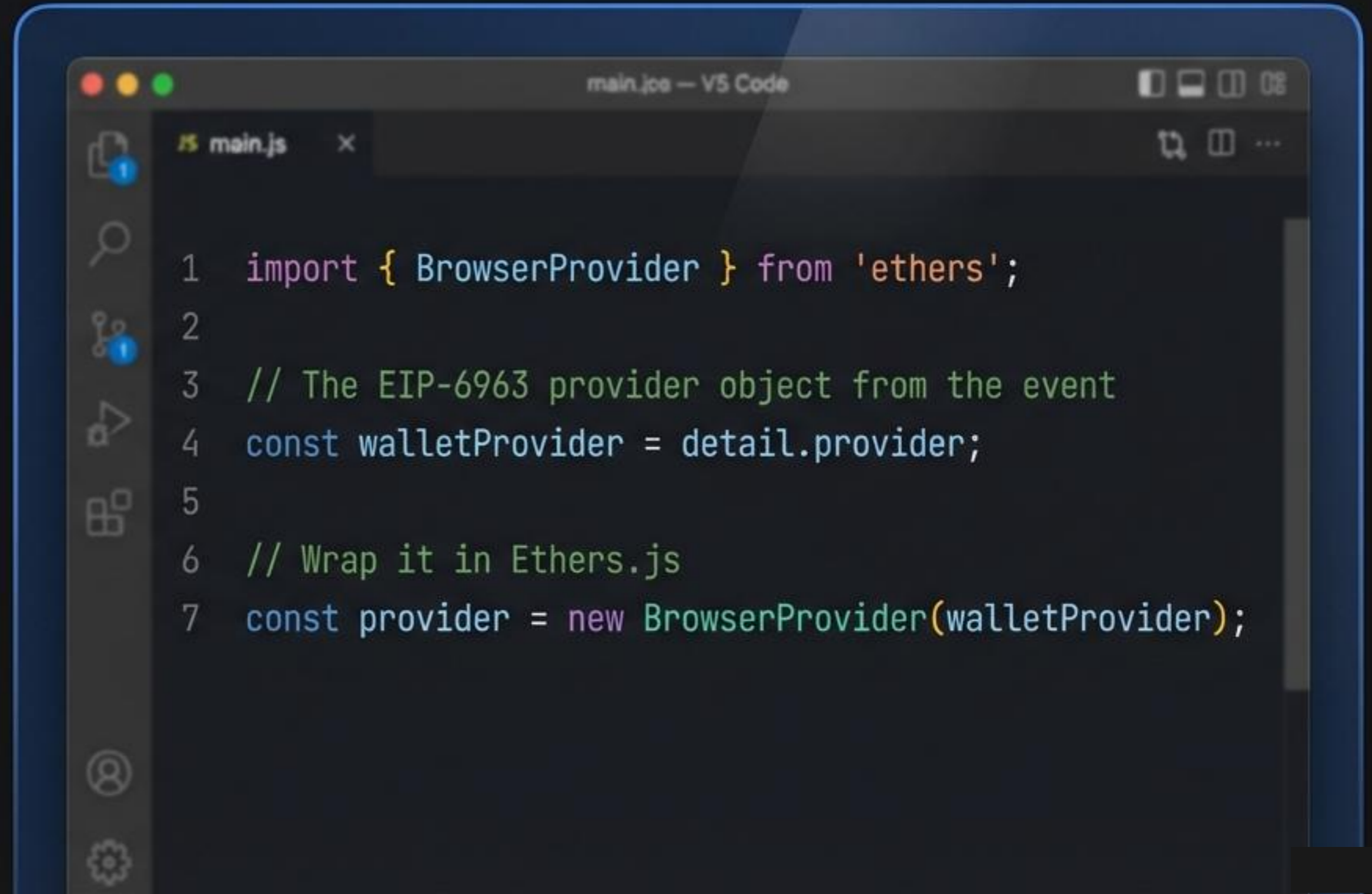
Coinbase Wallet

```
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
```

```
35 wallet.js x
arc > wallets > event > Connect Wallet
11 return rovent;
12
13
14 export default {binckPceHekhetConnectors} {
15   const wallet = new Wallet.conneet();
16   wallet.connection() => {
17     watb.hooveet and
18     sateott => {
19       retsvatStaconectHextos!);
20     }
21   }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
```

The Tool: Ethers.js

Abstraction. Wrapping complex RPC calls into clean JavaScript methods.

A screenshot of a Visual Studio Code editor window. The window title is "main.js - VS Code". The editor shows a JavaScript file named "main.js" with the following code:

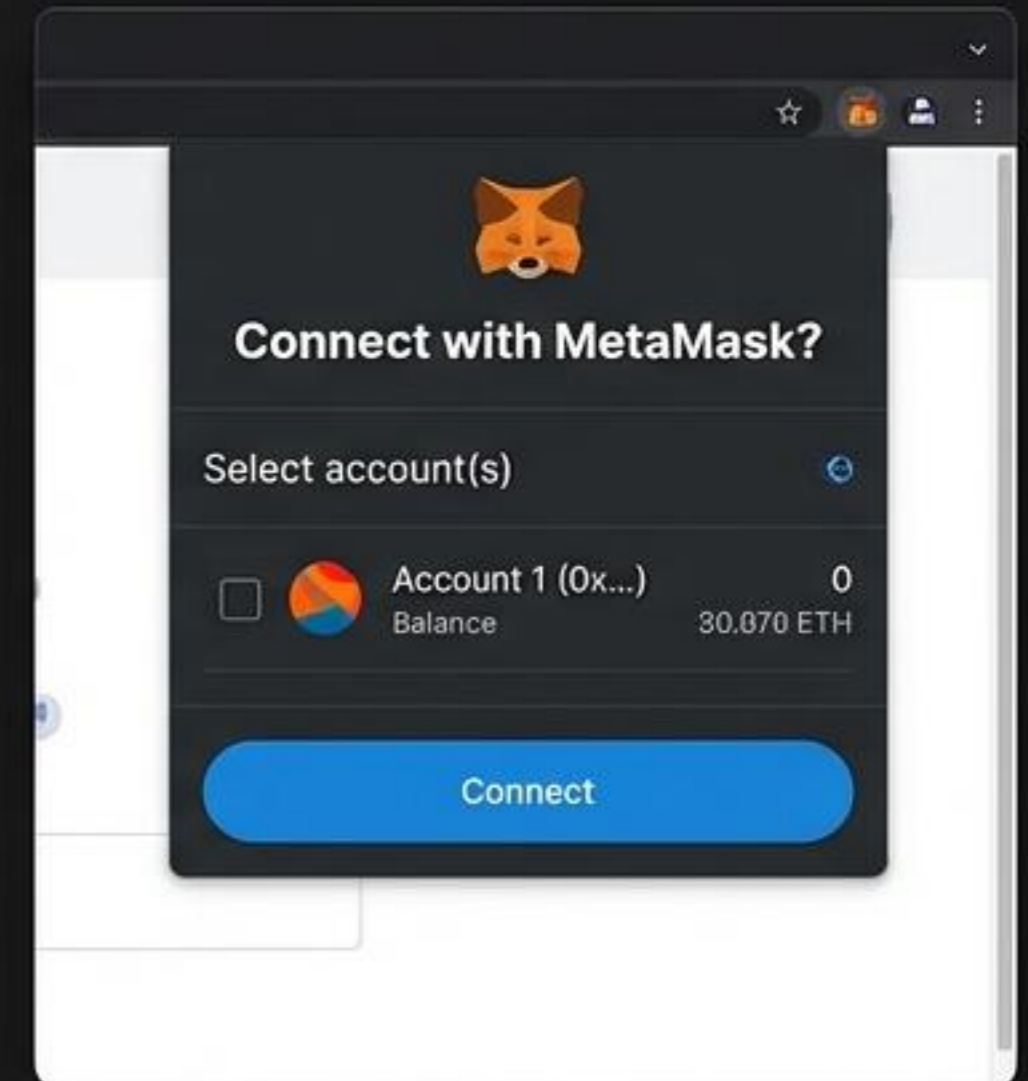
```
1 import { BrowserProvider } from 'ethers';
2
3 // The EIP-6963 provider object from the event
4 const walletProvider = detail.provider;
5
6 // Wrap it in Ethers.js
7 const provider = new BrowserProvider(walletProvider);
```

The code is displayed in a dark theme with syntax highlighting. The left sidebar shows the Explorer, Search, Source Control, Run and Debug, and Extensions views. The right sidebar shows the Output and Debug Console views.

The Connection

The Handshake. Requesting permission to view the address and initiate the Signer.

```
meto.11852 — VS Code
JS main.js x
JS main.js
1 // Request access
2 await provider.send('eth_requestAccounts', []);
3 // Get the signer
4 const signer = await provider.getSigner();
```

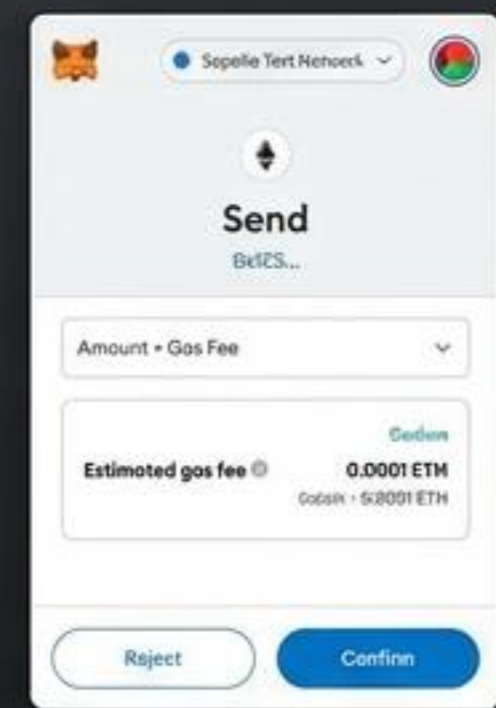


The Transaction: Sepolia Testnet

State Change. Moving value, paying gas, and recording truth on the ledger.

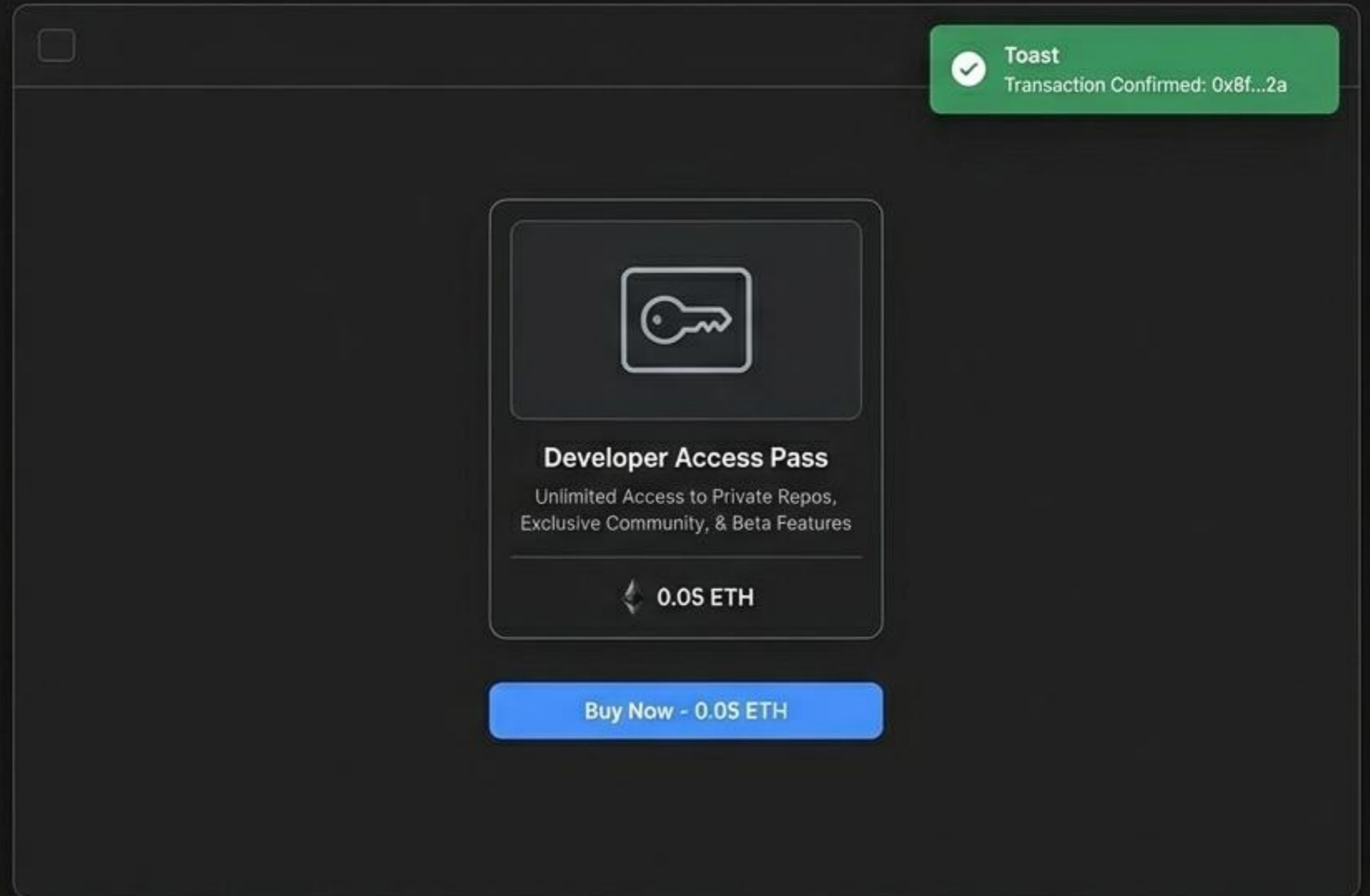
JetBrains Mono

```
JS main.js x
JS main.js > --
1  const tx = await signer.sendTransaction({
2    to: '0x123...',
3    value: parseEther('0.01')
4  });
5
```



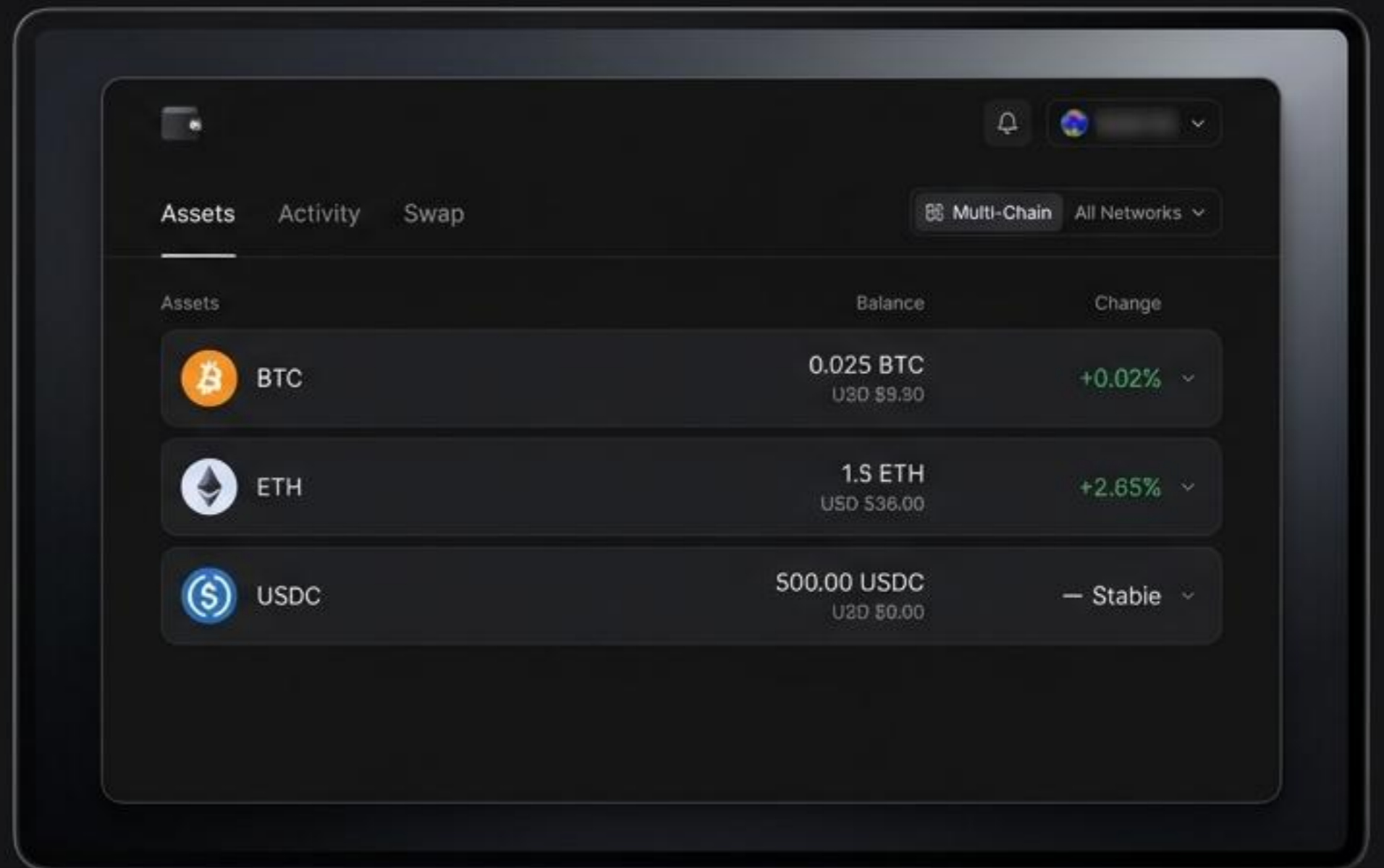
Value Exchange: The Marketplace

The code empowers the commerce. Connect -> Sign -> Own.



Beyond Ethereum: Multi-Coin Support

Interoperability.
Managing a diverse
portfolio of Bitcoin
and ERC-20s in a
single view.



Thank you

George Michoulis



linkedin.com/in/george-michoulis



github.com/gmixoulis/cryptobloom

