TOOPLOOX

How your JavaScript skills apply in blockchain space?

Michał Załęcki

Agenda

- Czym jest blockchain?
- Jak działa funkcja hashująca?
- Zbuduj własny blockchain?
- Czym jest Ethereum?
- Jak budujemy zdecentralizowane aplikacje?

Blockchain

- Niemutowalna struktura bloków
- Otwarty i zdecentralizowany rejestr
- Utrzymywany przez sieć peer-to-peer
- Finansowo wynagradza użytkowników

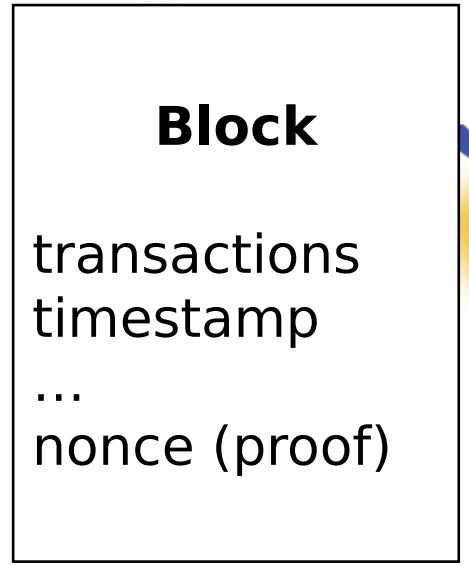
```
> sha256("GrillJS")
"c285ef8ad6a998fb47e432ac6a9e685c74ba5f3aef69ed979716184195d9b532"
```

```
> sha256("GrillJs")
"f848bd6afb301d9751c068d8931a0481f5085a30c2e1a1bac4e0122dcb498a51"
```

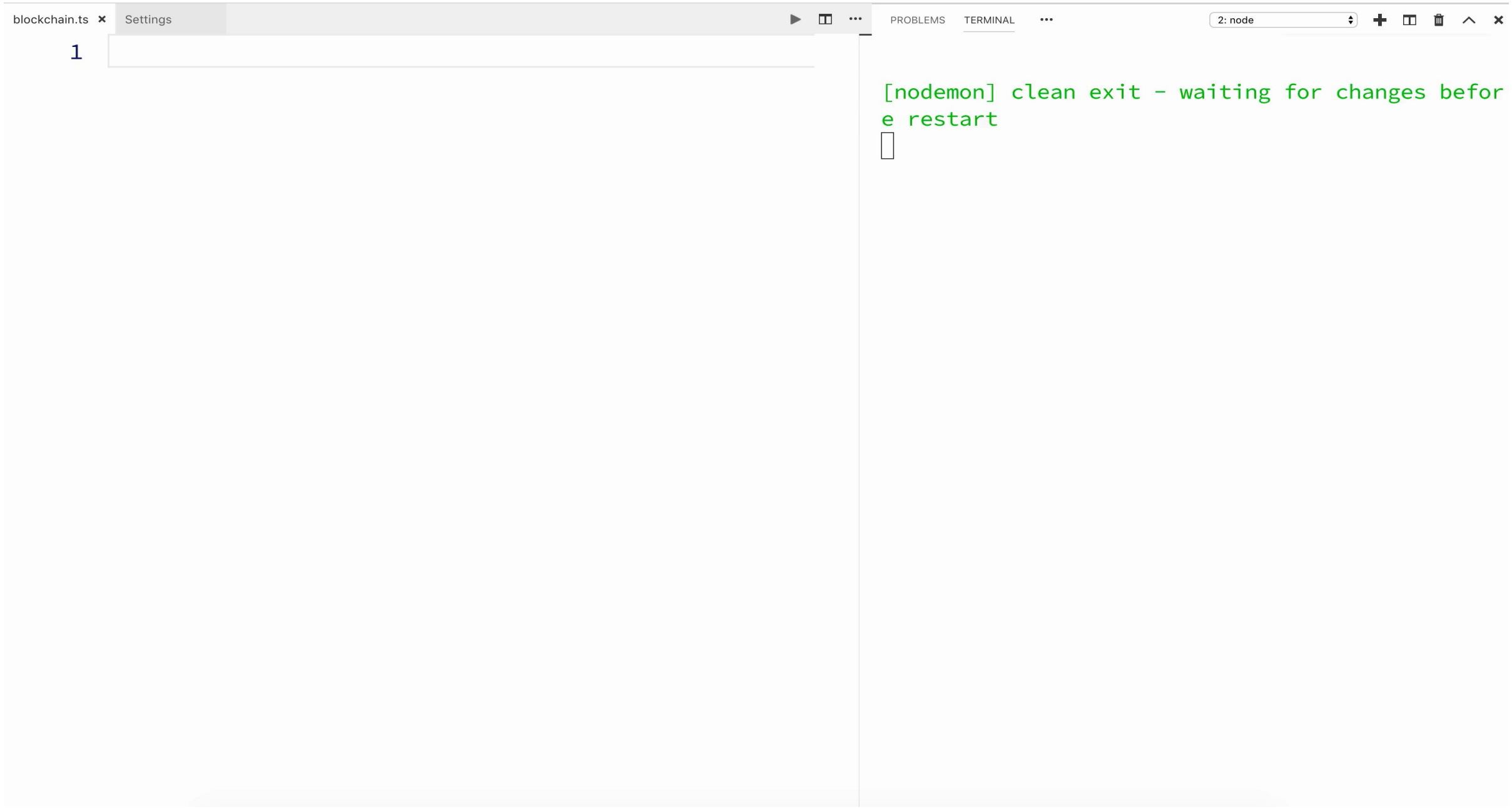
```
> sha256([...Array(10000)].map(() => "GrillJS").join(""))
"6ece48e3969d47f0a7246389610591 88e82ef3f48e5f34658e875211b5bebdf1"
```

Proof of Work

- zabezpiecza przed spamem i atakami DoS
- zagadka kryptograficzna
- ciężko znaleźć rozwiązanie problemu
- łatwo sprawdzić poprawność rozwiązania



sha256(block) < target


blockchain.ts
Settings
1
PROBLEMS
TERMINAL
2: node
[nodemon] clean exit - waiting for changes befor
e restart

blockchain.ts

```ts
import { SHA256 as sha256 } from "crypto-js";

class Block {
  public hash: string;

  constructor(
    public index: number,
    public prevHash: string,
    public timestamp: number,
    public data: any,
  ) {
    this.hash = this.calculateHash();
  }

  calculateHash() {
    return sha256(
      this.index +
      this.prevHash +
      this.timestamp +
      JSON.stringify(this.data)
    ).toString();
  }
}
```

PROBLEMS TERMINAL 2: node

```
Block {
  index: 0,
  prevHash: '',
  timestamp: 1547025531635,
  data: 'Genesis block',
  hash:
   '96b4b03b947a80974922eb03039d406fff4d97624ff4
57972c3940cff30f68cf' }
[nodemon] clean exit - waiting for changes befor
e restart
```

blockchain.ts

```
26      this.hash = this.calculateHash();
27    }
28
29    calculateHash() {
30      return sha256(
31        this.index +
32        this.prevHash +
33        this.timestamp +
34        JSON.stringify(this.data)
35      ).toString();
36    }
37  }
38
39  const target = calculateTarget(60000);
40  console.log(
41    padLeft(target.toString(16), 64, "0")
42  );
43
```

PROBLEMS TERMINAL 2: node

```
0001179ec9cbd821e0000000000000000000000000000000000000000000000
[nodemon] clean exit - waiting for changes before restart
```

blockchain.ts ●

```
43      while (
44        parseInt(sha256(this.hash + this.nonce).toString(),
45        >= target
46      ) {
47        this.nonce++;
48      }
49    }
50  }
51
52
53  const block = new Block(0, "", Date.now(), "Genesis block
54  block.mine();
55  console.log(block);
56
```

TERMINAL

2: node

```
Block {
  index: 0,
  prevHash: '',
  timestamp: 1547028356119,
  data: 'Genesis block!',
  difficulty: 60000,
  nonce: 150133,
  hash:
   '8e5d90ea3bff1896bba07d1a129d1e4145
73dc090b2bad7102d138dffe5e3c6a' }
[nodemon] clean exit - waiting for cha
nges before restart
```

blockchain.ts

```
68
69      return true;
70    }
71  }
72
73  const blockchain = new Blockchain();
74  const prev = blockchain.blocks[0];
75  const block = new Block(1, prev.hash, Date.now(), { amoun
76  block.mine();
77  console.log(blockchain.isBlockValid(block));
78
```

TERMINAL

2: node

```
true
[nodemon] clean exit - waiting for cha
nges before restart
```

Ethereum

- dwa typy kont: zwykłe (zewnętrzne) i smart kontrakty
- zwykłe konto jest zewnętrznie kontrolowane poprzez klucz prywatny
- smart kontrakt jest kontrolowany przez swój (niezmienialny) kod
- smart contract tworzymy poprzez wysłanie transakcji z kodem na adress 0
- funkcje smart kontraktu wywołujemy wysyłając inną transakcje

Solidity

- zorientowany obiektowo
- statycznie typowany
- kompiluje się do EVM bytecode
- składnia to mieszanina JavaScript, Python i C++ (typy)
- w dużym stopniu polega na ekoststemie JSa (npm, tooling)

```
import "openzeppelin-solidity/Ownable.sol";

contract Funding is Ownable {
  function donate() public payable {
    balances[msg.sender] += msg.value;
    raised += msg.value;
  }

  function withdraw() public onlyOwner {
    owner.transfer(address(this).balance);
  }

  function refund() public {
    require(!isFunded());
    uint amount = balances[msg.sender];
    require(amount > 0);
    balances[msg.sender] = 0;
    msg.sender.transfer(amount);
  }
}
```

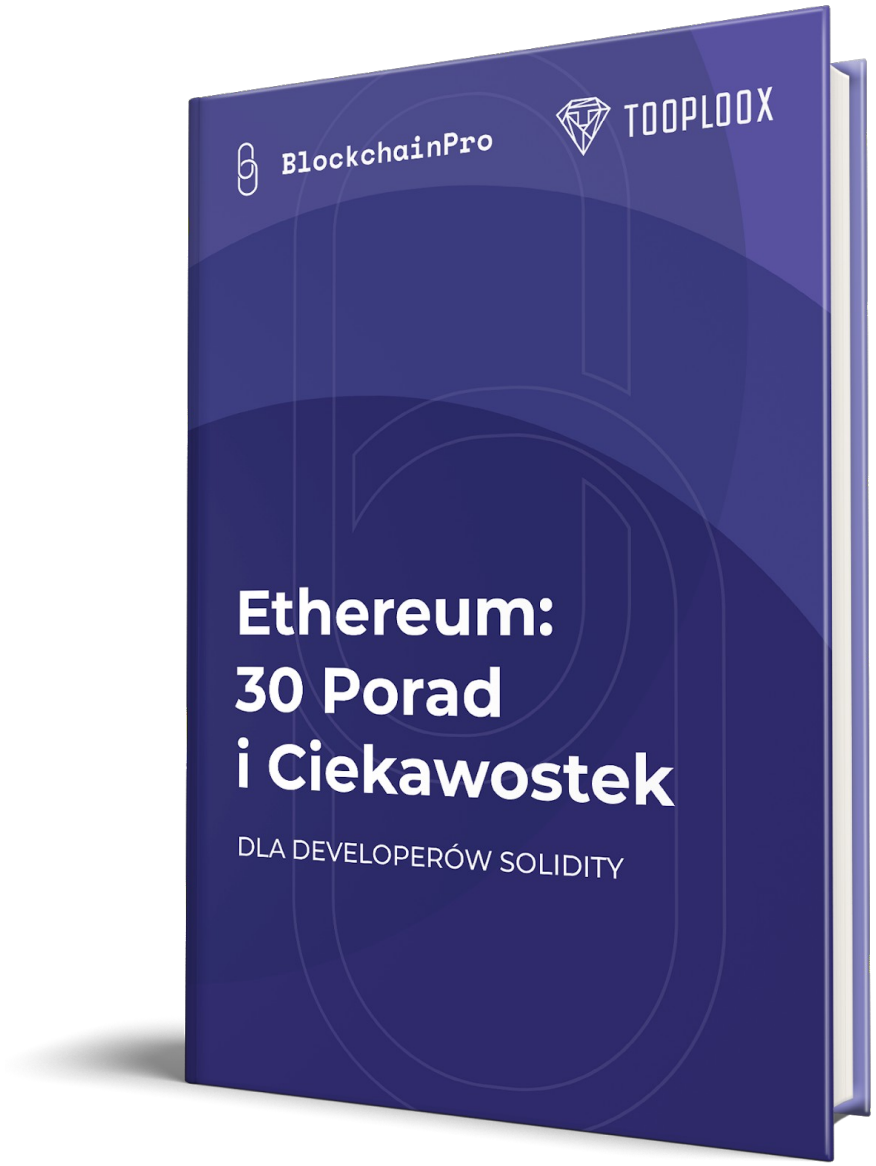
ĐApps

- łączy się do nodea Ethereum
- wywołuje procedury poprzez JSON-RPC
- wykorzystuje blockchain jako swój backend
- wspiera złożony cykl życia transakcji

```
const [account] = web3.eth.getAccounts()

Funding.methods.refund()
  .send({ from: account })
  .on("transactionHash", txhash => { })
  .on("receipt", receipt => { })
  .on("error", error => { })
  .on("confirmation", confs => { });
```

BlockchainPro.pl/ebook



michalzalecki.com/ebooks

-50% with „cross” code

Dziękuję! Pytania?

@MichalZalecki
michal@michalzalecki.com