


Solidity Development

James Young

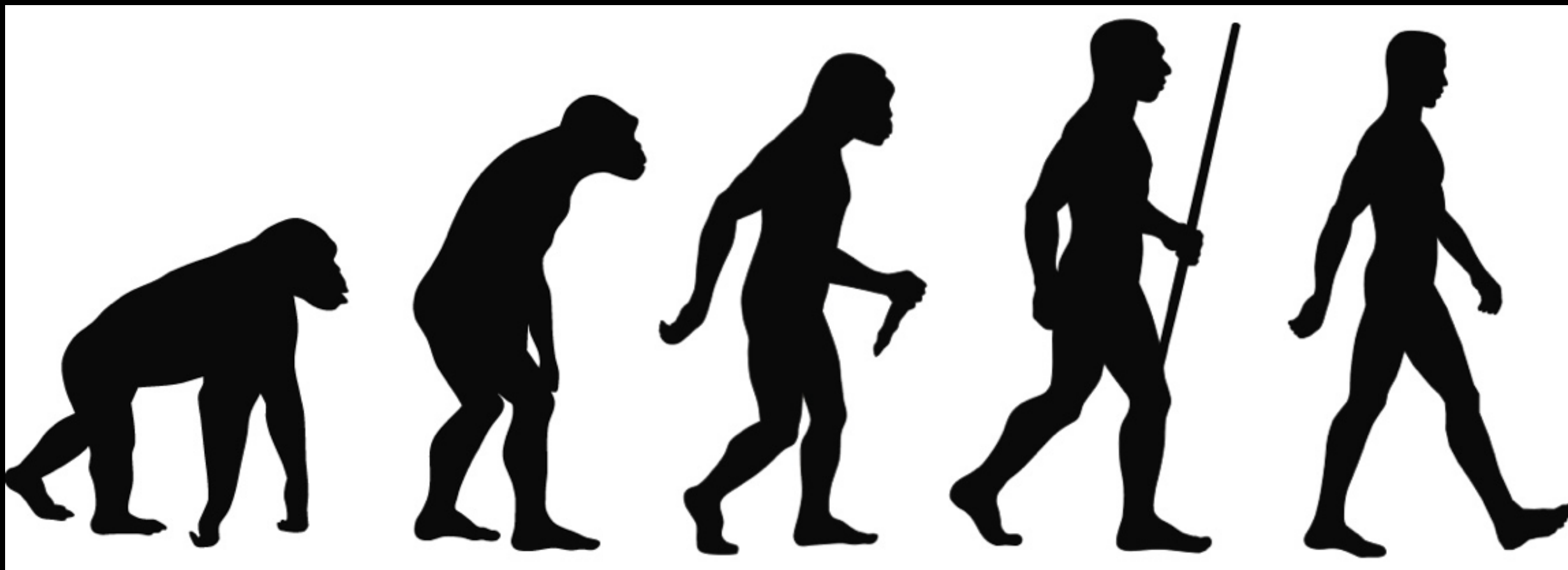
about me

- Video streaming :
 - InterVU acquired by Akamai
 - Click.tv acquired by Cisco
- Social games : FarmVille launch team
- Online advertising : AdChain co-author
- I  state channels

ERC-20

CryptoKitties





front-end

backend

full-stack

Web3

State Channels

^

**will present
tomorrow**

#rekt

\$117,000,000 bug

9 ■■■■ js/src/contracts/snippets/enhanced-wallet.sol Show comments View

Original Code	Patched Code
<pre>@ -104,7 +104,7 @@ contract WalletLibrary is WalletEvents { 104 105 // constructor is given number of sigs required to do protected "onlymanyowners" transactions 106 // as well as the selection of addresses capable of confirming them. 107 - function initMultiowned(address[] _owners, uint _required) { 108 m_numOwners = _owners.length + 1; 109 m_owners[1] = uint(msg.sender); 110 m_ownerIndex[uint(msg.sender)] = 1; @ -198,7 +198,7 @@ contract WalletLibrary is WalletEvents { 198 } 199 200 // constructor - stores initial daily limit and records the present day's index. 201 - function initDaylimit(uint _limit) { 202 m_dailyLimit = _limit; 203 m_lastDay = today(); 204 } @ -211,9 +211,12 @@ contract WalletLibrary is WalletEvents { 211 m_spentToday = 0; 212 } 213</pre>	<pre>104 105 // constructor is given number of sigs required to do protected "onlymanyowners" transactions 106 // as well as the selection of addresses capable of confirming them. 107 + function initMultiowned(address[] _owners, uint _required) internal { 108 m_numOwners = _owners.length + 1; 109 m_owners[1] = uint(msg.sender); 110 m_ownerIndex[uint(msg.sender)] = 1; 198 199 200 // constructor - stores initial daily limit and records the present day's index. 201 + function initDaylimit(uint _limit) internal { 202 m_dailyLimit = _limit; 203 m_lastDay = today(); 204 } 211 m_spentToday = 0; 212 } 213 214 + // throw unless the contract is not yet initialized. 215 + modifier only_uninitialized { if (m_numOwners > 0) throw; _; }</pre>

**Every contract is a
financial application**

[illegible]

7

♡ 122



ONE DOES NOT SIMPLY

WRITE SOLIDITY

write tests

until you are sure there are no bugs

Once you have ZERO
bugs

write more tests

**Then audit your
contract**

write more tests

Avoid Adversarial Owner

too many “safeguards” in contracts

Local dev environments

- Remix
- Truffle
- Parity
- Geth

Private networks

- Parity
- Puppeth (Geth)

Test networks

- Ropsten
- Rinkeby
- Kovan

some things to note

Addresses

- No difference between a user and a smart contract
- A smart contract does not have a private key
- Convention : deployment address is contract owner

GAS

$$\text{Transaction fee} = \text{Gas Limit} * \text{Gas Price}$$

High gas prices get
picked first

BEWARE : block gas limit



Valid transactions will FAIL

runs out of gas

- or -

exceeds block gas limit

GAS INFO

- bit.ly/gas-opcodes
- ethgasstation.info

UINT256

$$2^{256} - 1 =$$

115792089237316195
42357098500868790785
32699846656405640394
57584007913129639935

floating point

NO ROUNDING ERRORS

IF THERE ARE NO DECIMALS

Common Primitives

- Boolean
- Integers
- Strings
- Addresses

Common Data Structures

- Array
- Struct
- Mapping

Data Locations

- Memory
- Storage

Modifiers

- Code that run inline
- Test explicitly for missing modifiers

Events

- Return values
(example : UI updates)
- Async messaging
- Log storage

Exceptions

- `assert` : uses all gas
- `require` : refunds gas

```
pragma solidity ^0.4.18;

contract Courses {

    address public ownerAddress;

    struct Instructor {
        uint courseNum;
        string name;
        bool active;
    }

    mapping (address => Instructor) instructors;
    address[] public instructorArray;

    modifier ownerOnly {
        require(msg.sender == ownerAddress);
    }

    event NewInstructor(address instructor);

    function Courses() public {
        ownerAddress = msg.sender;
    }

    function setInstructor(address _address, uint _courseNum, string _name, bool _active) public {
        Instructor storage instructor = instructors[_address];
        instructor.courseNum = _courseNum;
        instructor.name = _name;
        instructor.active = _active;

        instructorArray.push(_address);
        emit NewInstructor(_address);
    }

    function getInstructor(address _address) view public ownerOnly returns (uint, string, bool) {
        return (instructors[_address].courseNum, instructors[_address].name, instructors[_address].active);
    }
}
```

What is wrong with this?

```
uint active = 0;

for (uint i=0; i<instructorsArray; i++)
{
    if (instructorsArray[i].active)
    {
        active += instructorsArray[i].active;
    }
    return active;
}
```

Do not loop through dynamic arrays

```
function getInstructors() view public  
returns (address[])  
{  
    return instructorArray;  
}
```

**Do not loop through
dynamic arrays**

If possible, loop through
array elements off-chain and
design on-chain functions to
process array elements.

If you remember ONE thing today



Felix Weis @FelixWeis · 16h

There are 2 things you can't doublespend:

- 1) Bitcoin
- 2) Your reputation after pumping worthless ICOs

