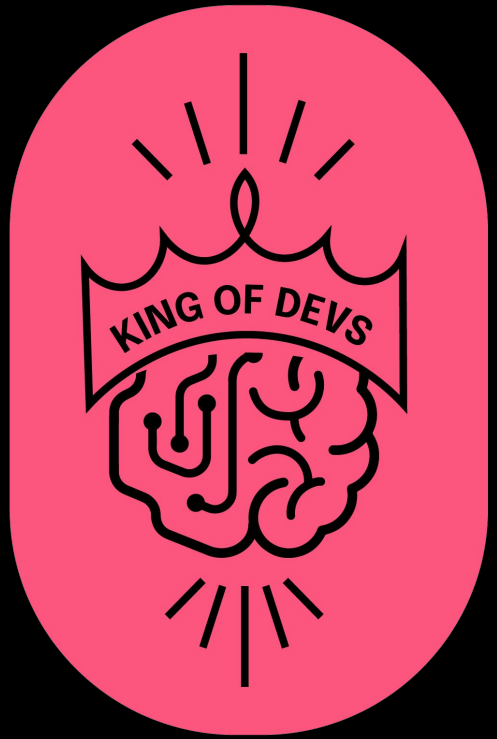
KING OF DEVS

KING OF



WORKSHOP DAY 1

Tips and optimizations

```
2  pragma solidity ^0.8.9;
3
4  contract ReasonStrings {
5      uint256 balance;
6      uint256 amount;
7
8      modifier badReasonString {
9          require(balance >= amount, "To whomsoever it may concern.
10         I am writing this error message to let you know that the amount you are trying to
11         transfer is unfortunately more than your current balance. Perhaps you made a typo or
12         you are just trying to be a hacker boi. In any case, this transaction is going to revert.
13         Please try again with a lower amount. Warm regards, EVM");
14         _;
15     }
16
17     modifier goodReasonString {
18         require(balance >= amount, "Insufficient balance");
19     }
20
21     modifier possibleOptionForLongStrings {
22         require(balance >= amount, "CODE ERROR: 20, please refer to www....");
23     }
24
25 }
```

REQUIRE STRINGS

- Cualquier require-string ocupa al menos 32 bytes
    - Traten de no usar más de eso
- Opciones posibles:
    - Strings cortos lo más descriptivos posibles
    - Strings con ShortUrls a la documentación de dicho error
    - Códigos de error y documentación por cada código

2022

```solidity
//Balances
mapping(address => uint) s_balances;

// Historico de transferencias
mapping(address => uint256[]) s_transfers;

function transfer(address payable receiver, uint256 amount) public payable{
    require(s_balances[address(this)]>= amount, "Not enough balance");

    transferBalance(address(this),receiver, amount);

    //SEND ETH
    (bool sent,) = receiver.call{value: amount}("");
    require(sent, "Failed to send Ether");

    //MAL USO DE STORAGE: GUARDO QUE LE ENVIE ETH
    s_transfers[receiver].push(amount);
}
```

VS

```solidity
//Balances
mapping(address => uint) s_balances;

//Transfer Event
event Transfer(address indexed sender, address indexed receiver, uint256 amount);

function transfer(address payable receiver, uint256 amount) public payable{
    require(s_balances[address(this)]>= amount, "Not enough balance");

    transferBalance(address(this),receiver, amount);

    //SEND ETH
    (bool sent,) = receiver.call{value: amount}("");
    require(sent, "Failed to send Ether");

    emit Transfer(address(this), receiver, amount);
}
```

Ahorro de **gas y** de **storage**: ~40%

```solidity
pragma solidity ^0.8.9;

contract VariablesPacking {

    struct RegistroSinPacking {
        address variableA;
        address variableB;
        uint96  variableC;
        uint96  variableD;
    }
    /**
        variableA = 20/32 bytes --> SLOT 1 = 20bytes
        variableB = 20/32 bytes --> SLOT 2 = 20 bytes
        variableC = 12/32 bytes --> SLOT 2 = 20 + 12 = 32 bytes
        variableD = 12/32 bytes --> SLOT 3 = 12 bytes
    TOTAL = 3 SLOTS
    */

    struct RegistroConPacking {
        address variableA;
        uint96  variableC;
        address variableB;
        uint96  variableD;
    }

    /**
        variableA = 20/32 bytes --> SLOT 1 = 20bytes
        variableC = 12/32 bytes --> SLOT 1 = 20 + 12 = 32 bytes
        variableB = 20/32 bytes --> SLOT 2 = 20 bytes
        variableD = 12/32 bytes --> SLOT 2 = 20 + 12 bytes = 32 bytes
    ¡ TOTAL = 2 SLOTS !
    */

}
```

PACK YOUR VARIABLES

- Cada storage-slot es de 32 bytes
- Se almacenan de forma secuencial
- Cuando una nueva variable declarada no cabe dentro de un slot, se abre uno nuevo

```
for (uint256 i=0; i < array.length; i++) {
   doStuff(array[i]);
}
```

[PASS] test_loop() (gas: 106969)

```
uint256 length = array.length;
for (uint256 i=0; i < length; i++) {
   doStuff(array[i]);
}
```

[PASS] test_loop() (gas: 106682)

```
uint256 length = array.length;
for (uint256 i=0; i < length; ++i) {
   doStuff(array[i]);
}
```

[PASS] test_loop() (gas: 106182)

```
uint256 length = array.length;
for (uint256 i=0; i < length;) {
   doStuff(array[i]);
   unchecked { i++; }
}
```

[PASS] test_loop() (gas: 94382)

```
uint256 length = array.length;
for (uint256 i=0; i < length;) {
   doStuff(array[i]);
   unchecked { ++i; }
}
```

[PASS] test_loop() (gas: 93882)

GAS OPTIMIZATION

- Hay mucho factores que afectan el uso de gas
- Problema de la sábana corta
    - A veces para optimizar gas se sacrifica espacio, y viceversa. Es responsabilidad del dev tratar de encontrar el mejor equilibrio para nuestro contrato

GRACIAS