EXPLORING CONTRACT UPGRADE PATTERNS

@mseijas

Matias Seijas

@mseijas

Tanooki Labs

tanookilabs.com

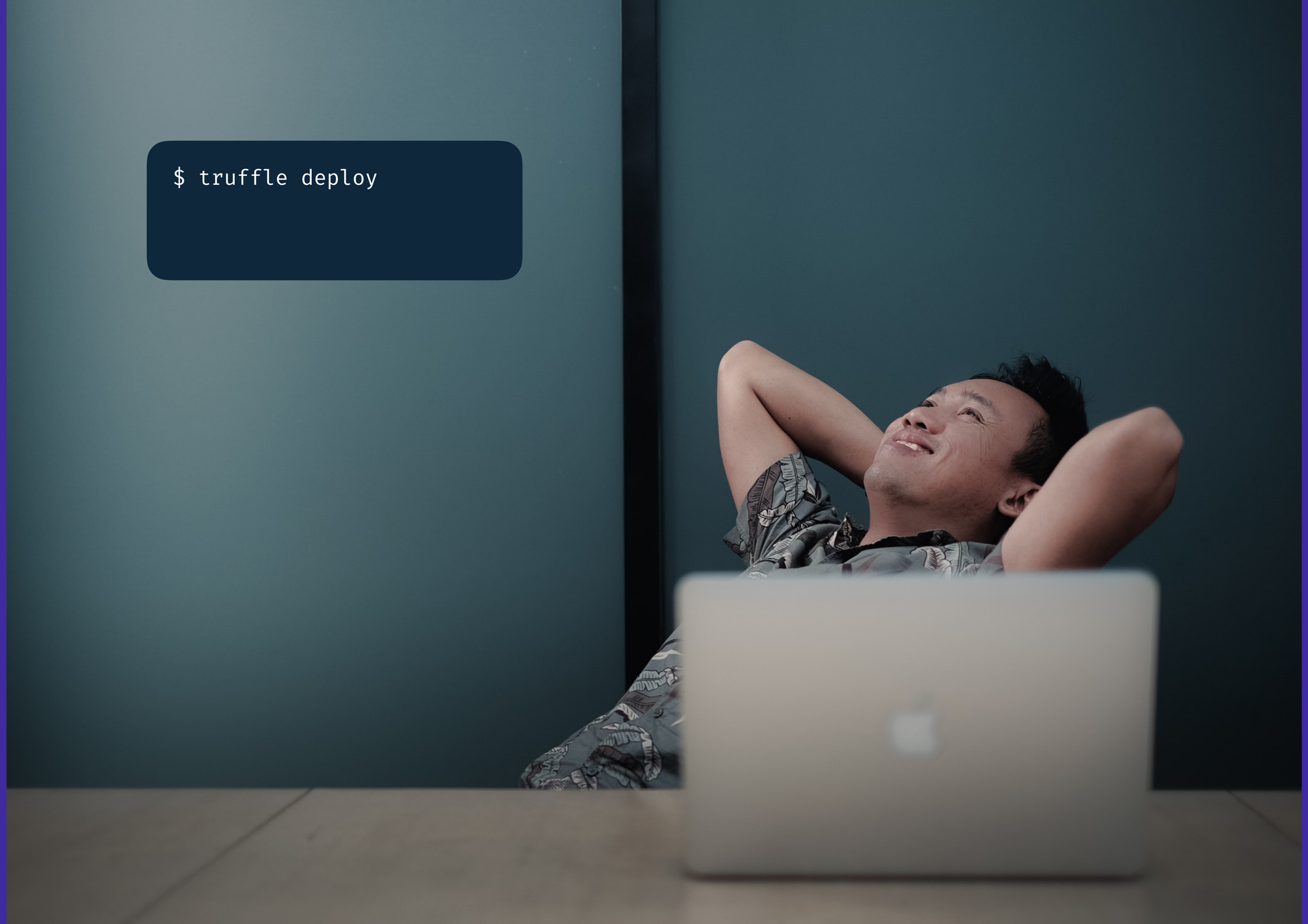
zeppelin

ARAGON

zeppelin

ARAGON

A DAY IN THE LIFE OF AN ETH DEVELOPER...


$ truffle deploy

A FEW DAYS LATER...


TRUE STORY

DAO hack

$60 million in ETH

Parity hack

$162 million in ETH

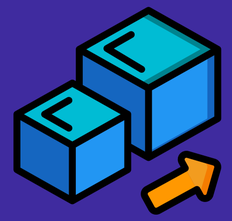
WE NEED A WAY TO

FIX

MISTAKES

CONTRACT UPGRADES ARE NECESSARY

Bugs

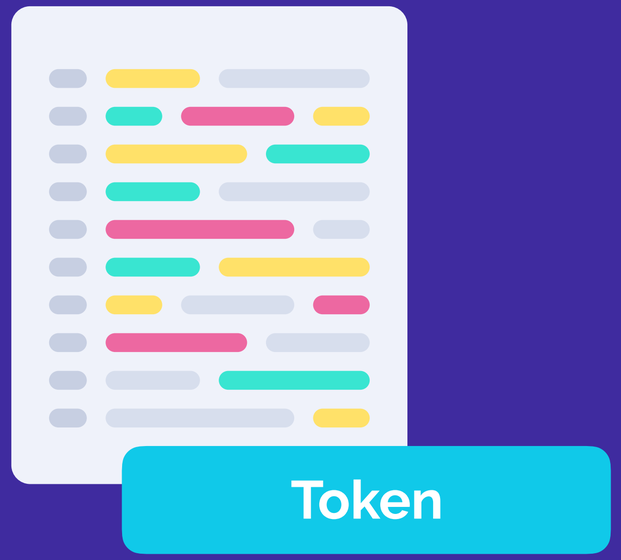


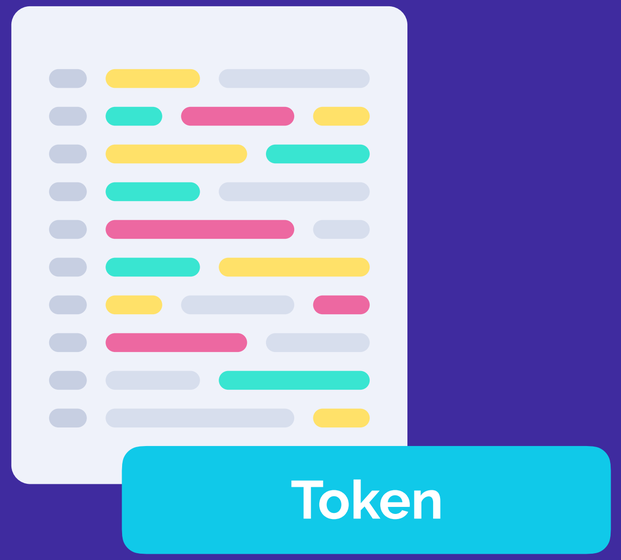
Expand Functionality

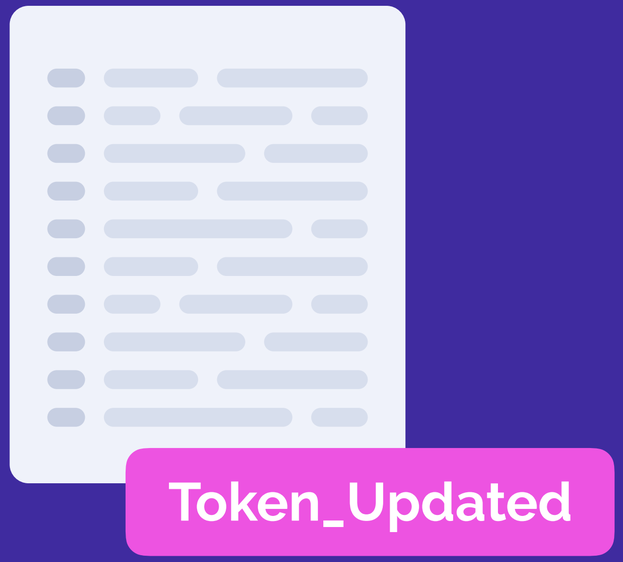
Compiler Vulnerabilities

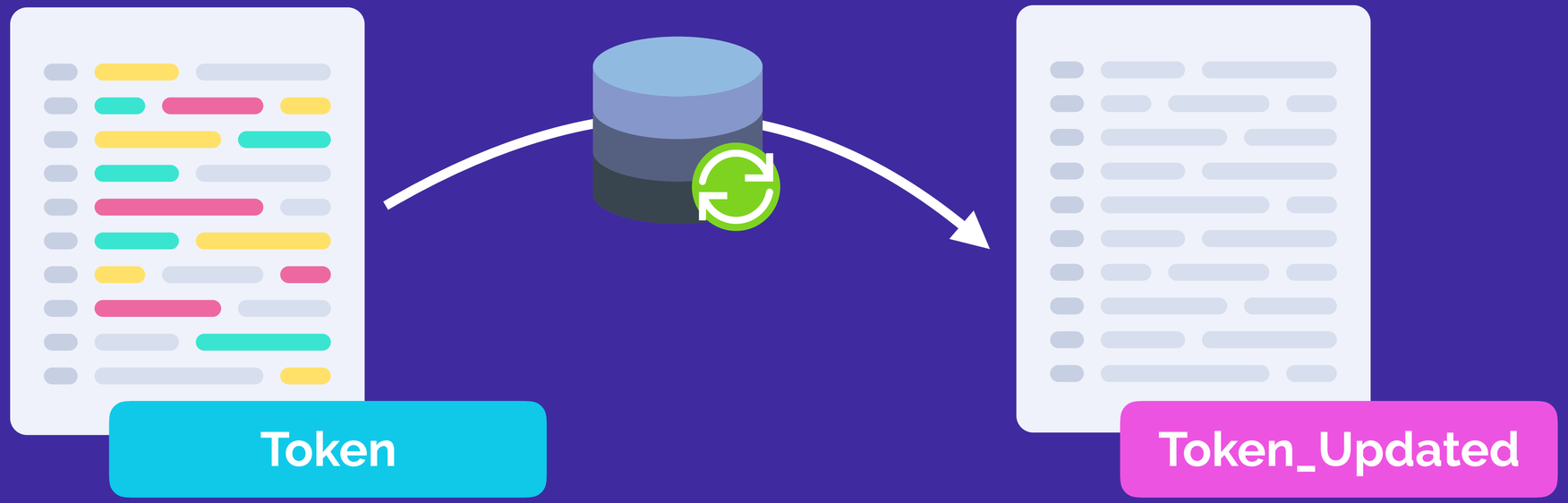
UPGRADES ARE NEEDED

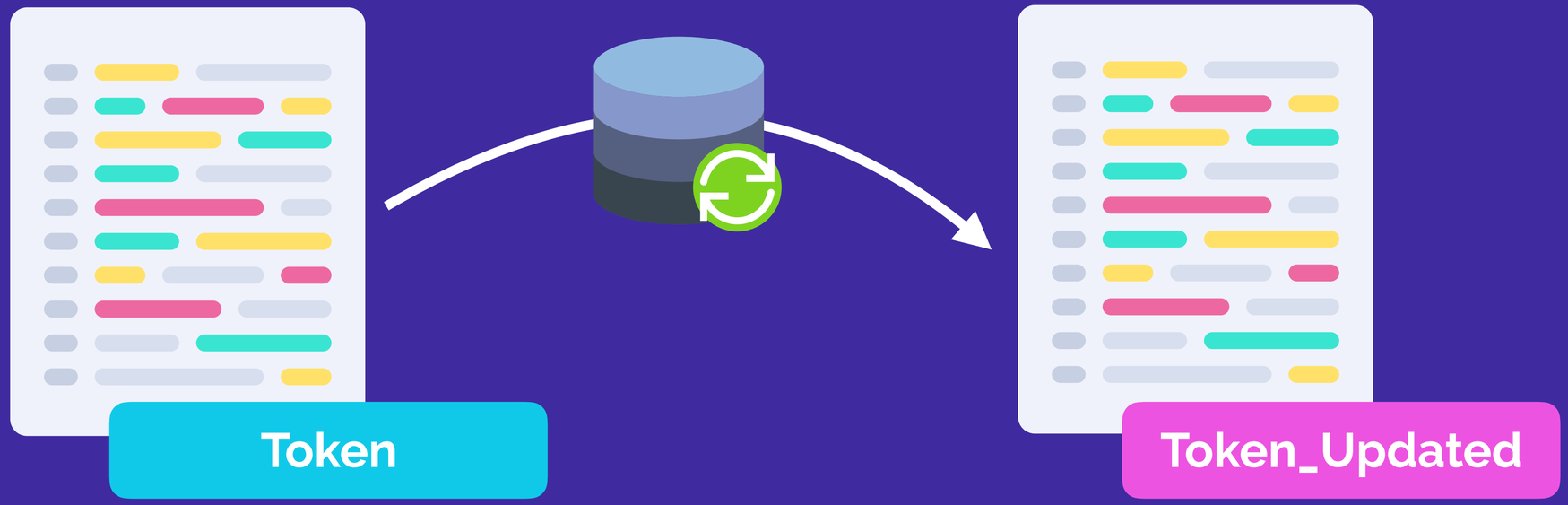
LET'S UPGRADE A CONTRACT


Token


Token

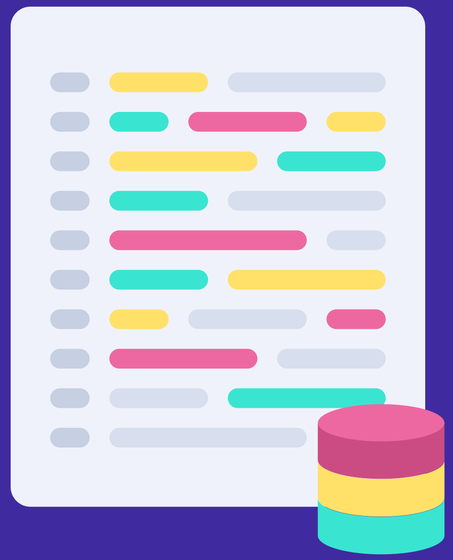

Token_Updated


Token
Token_Updated


Token
Token_Updated


Token_Updated

HOW DO I UPGRADE MY CONTRACT BUT PRESERVE ITS DATA ?



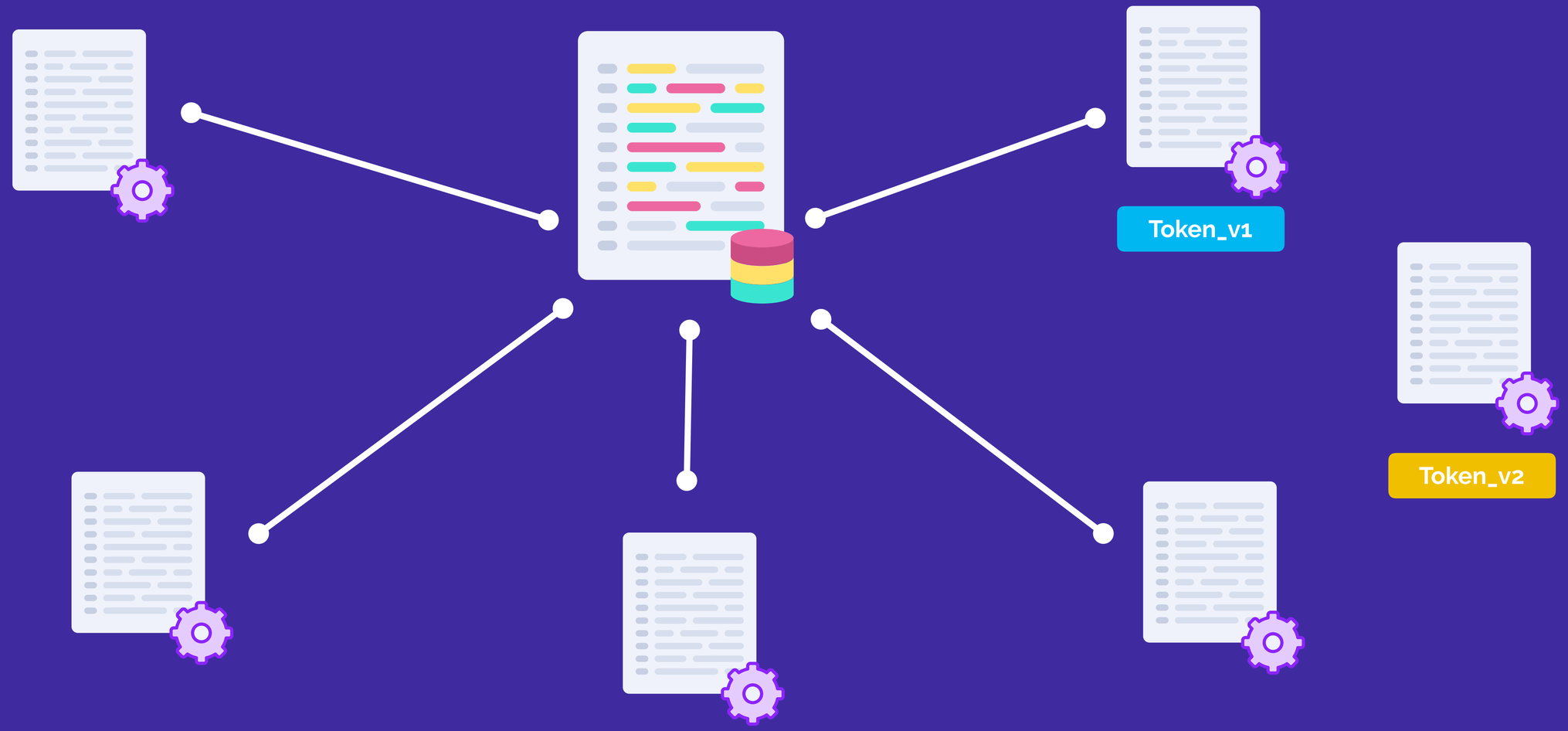
Data

Logic

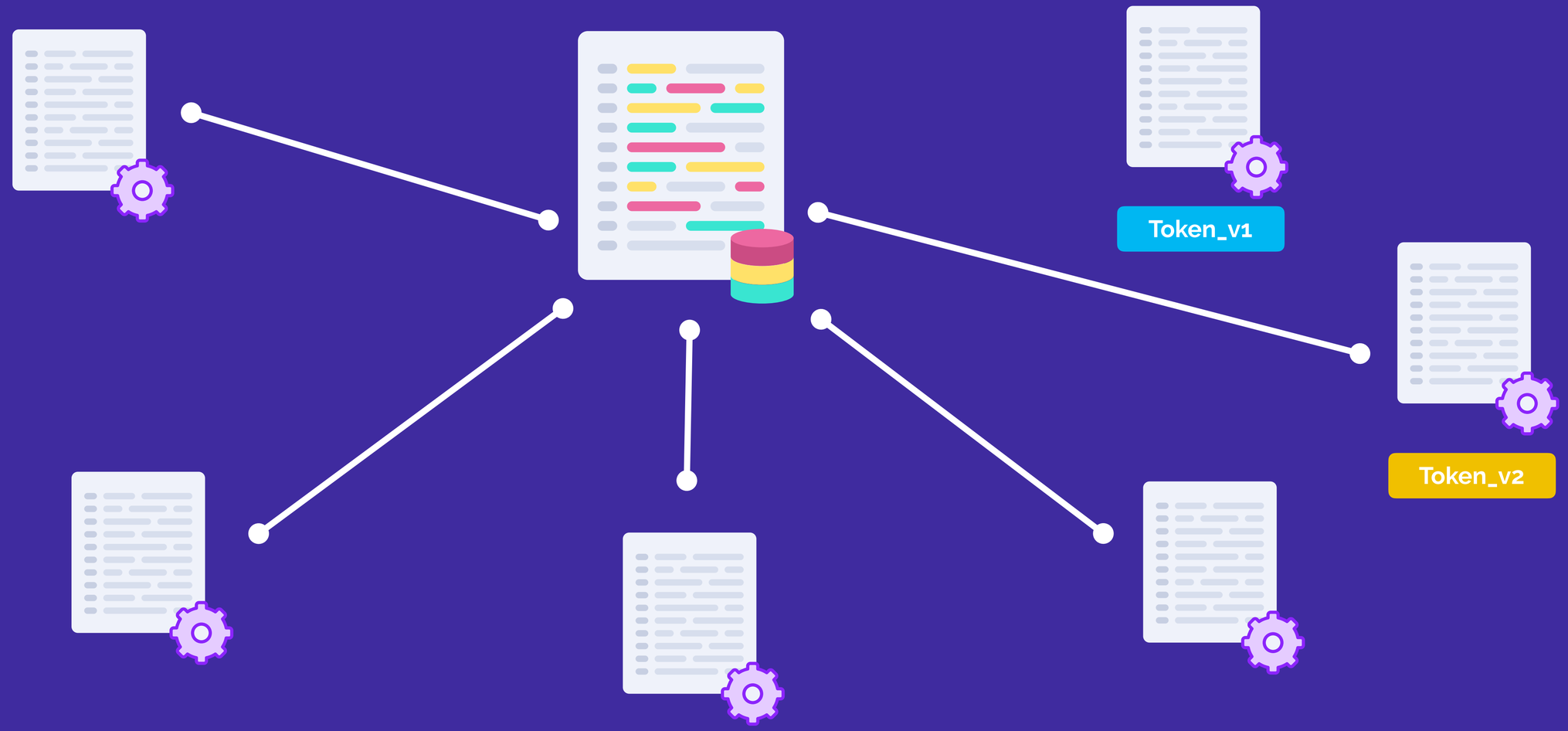
HUB & SPOKE

HUB & SPOKE

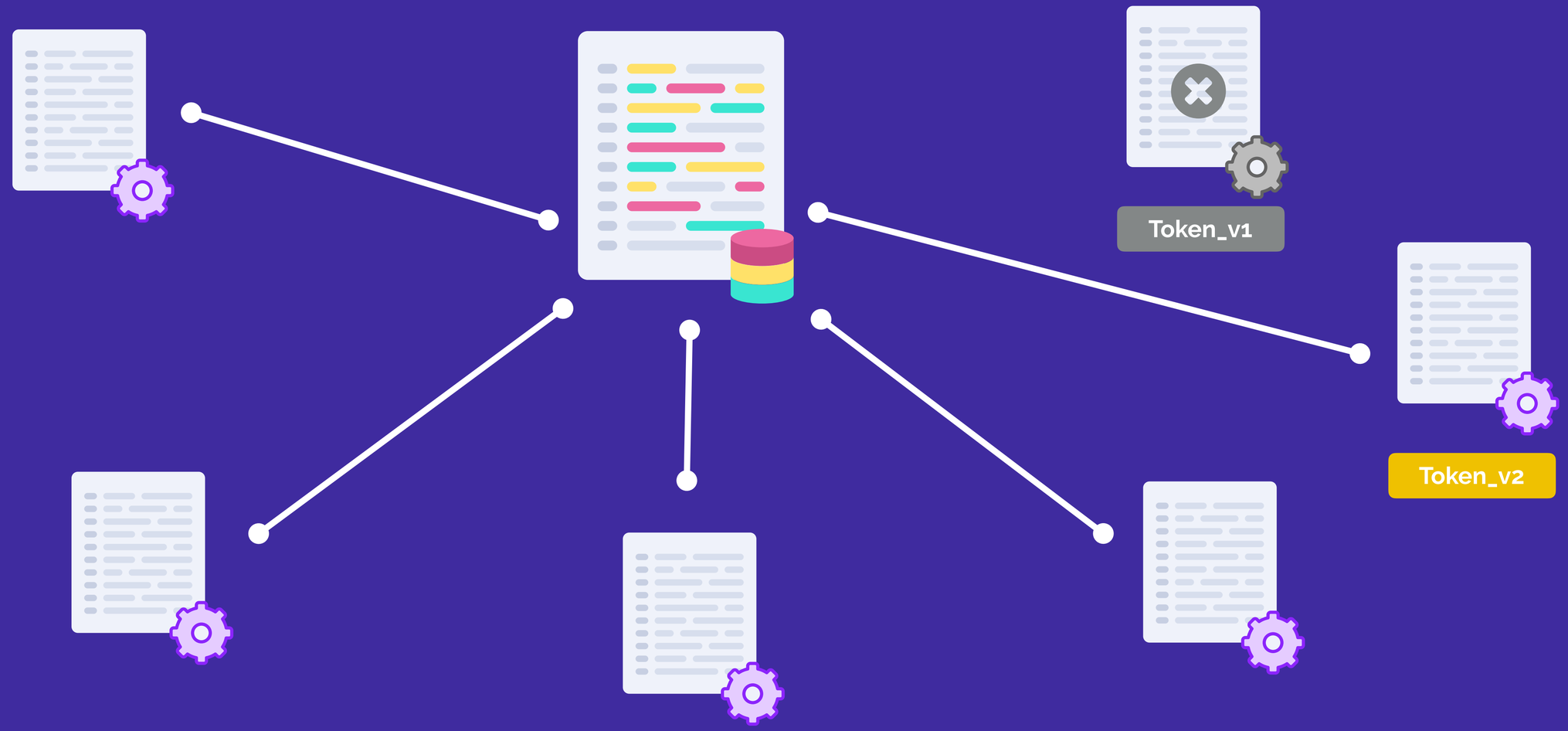
HUB & SPOKE

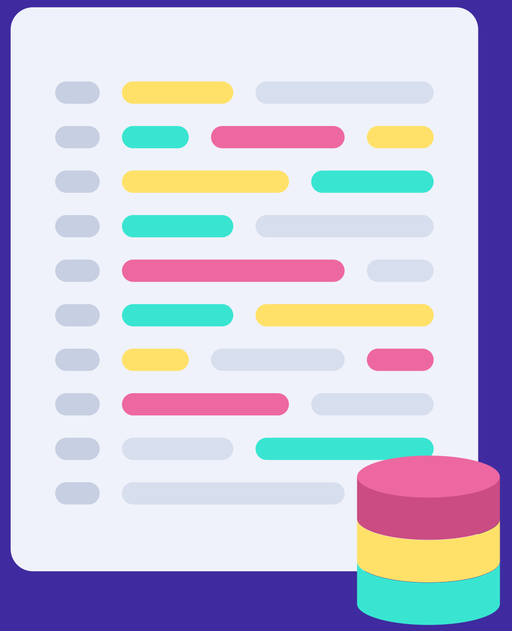


HUB & SPOKE



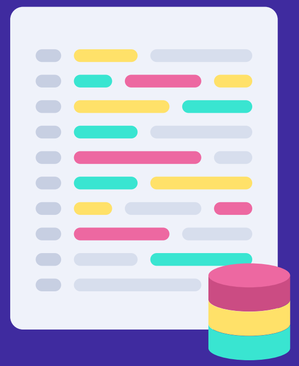
HUB & SPOKE



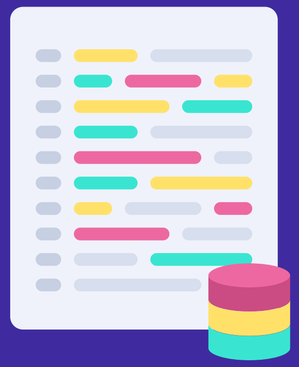


Flexible

Adaptable



BUILDING A FLEXIBLE DATA STORAGE



ETERNAL STORAGE


STORE ALL THE THINGS!

```
contract EternalStorage {









}
```

ETERNAL STORAGE

```
contract EternalStorage {


mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;









}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}




}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

ETERNAL STORAGE

```
contract EternalStorage {


mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;


function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {









}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {

address store;









}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {

address store;

}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

}
```

ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) private _bool;
mapping(bytes32⇒uint) private _uint;
mapping(bytes32⇒int) private _int;
mapping(bytes32⇒address) private _address;
mapping(bytes32⇒string) private _string;
mapping(bytes32⇒bytes) private _bytes;

function setBool(string _key, bool _value) {
  _bool[keccak256(_key)] = _value;
}

function getBool(string _key) returns(bool) {
  return _bool[keccak256(_key)];
}

}
```

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

}
```

ETERNAL STORAGE

BUT HOW DO I UPGRADE IT ?

Eternal Storage

`0×111 …`

Eternal Storage

0×111 …

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

}
```

Token_v1

0×732 …

Eternal Storage

0×111 …

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

}
```

Token_v1

0×732 …

Eternal Storage

0×111 …

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

}
```

Token_v1

0×732 …

```
contract Token_v2 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

function toggleActive() {
  bool isActive = store.getBool(“token.is_active”);
  store.setBool(“token.is_active”) = !isActive;
}

}
```

Token_v2

0×469 …

Eternal Storage

0×111 …

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

}
```

Token_v1

0×732 …

```
contract Token_v2 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

function toggleActive() {
  bool isActive = store.getBool(“token.is_active”);
  store.setBool(“token.is_active”) = !isActive;
}

}
```

Token_v2

0×469 …

Eternal Storage

0×111 …

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool)
  return store.getBool(“token.is_active”);
}

}
```

Token_v1

0×732 …

```
contract Token_v2 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

function toggleActive() {
  bool isActive = store.getBool(“token.is_active”);
  store.setBool(“token.is_active”) = !isActive;
}

}
```

Token_v2

0×469 …

Eternal Storage

0×111 …

```
contract Token_v1 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool)
  return store.getBool(“token.is_active”);
}

}
```

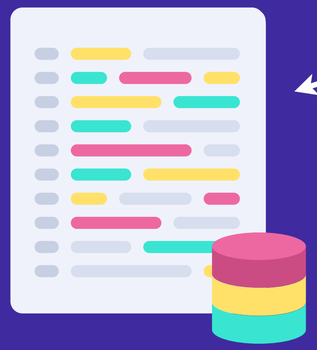
Token_v1

0×732 …

```
contract Token_v2 {

address store;

function setIsActive(bool _value) {
  store.setBool(“token.is_active”) = _value;
}

function getIsActive() returns(bool) {
  return store.getBool(“token.is_active”);
}

function toggleActive() {
  bool isActive = store.getBool(“token.is_active”);
  store.setBool(“token.is_active”) = !isActive;
}

}
```

Token_v2

0×469 …

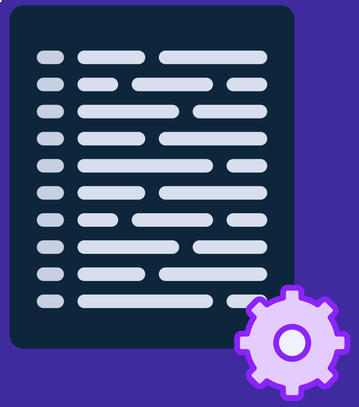


Eternal Storage

0×111 …

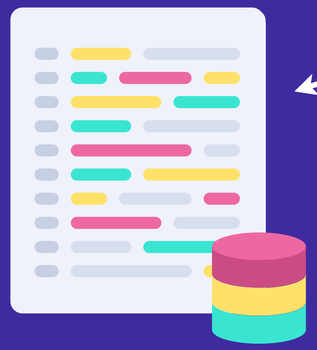
Token_v1

0×732 …



Token_v2

0×469 …



Eternal Storage

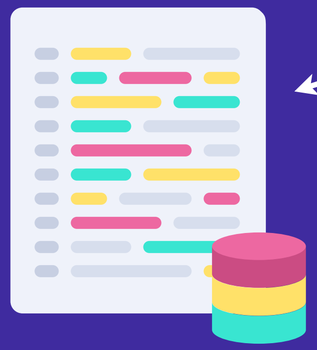
0×111 …

Token_v1

0×732 …

Token_v2

0×469 …

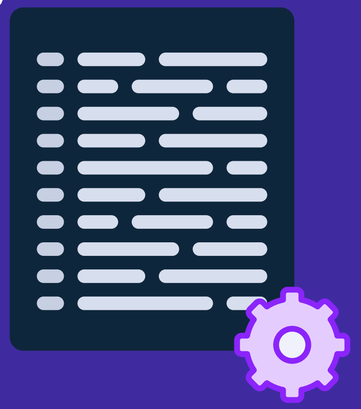


Eternal Storage

0×111 …

Token_v1

0×732 …



Token_v2

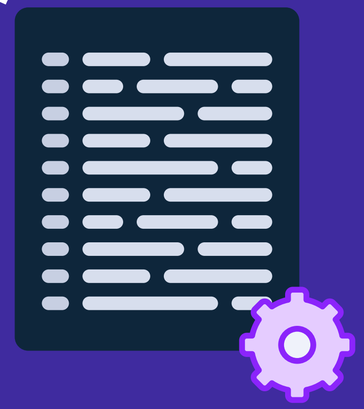
0×469 …

Eternal Storage

0×111 …

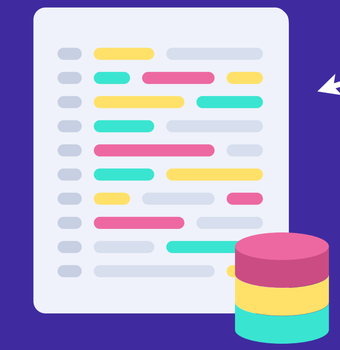
Token_v1

0×732 …



Token_v2

0×469 …



Eternal Storage

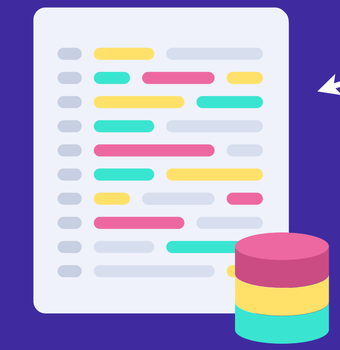
0×111 …

Token_v1

0×732 …

Token_v2

0×469 …

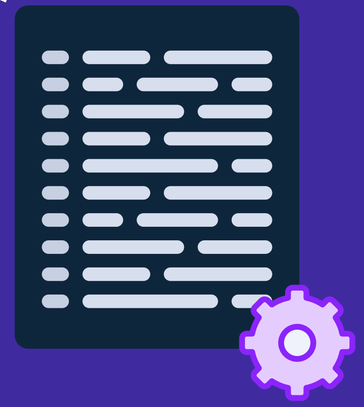


Eternal Storage

0×111 …

Token_v1

0×732 …



Token_v2

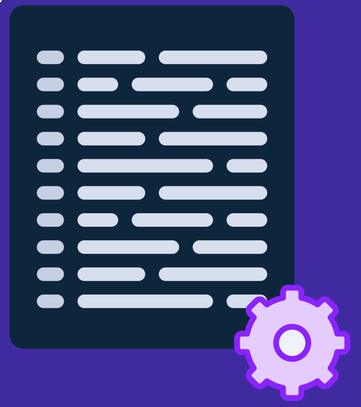
0×469 …

Eternal Storage

0×111 …

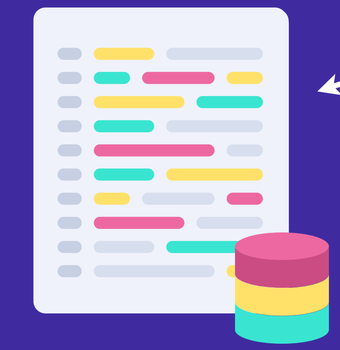
Token_v1

0×732 …



Token_v2

0×469 …

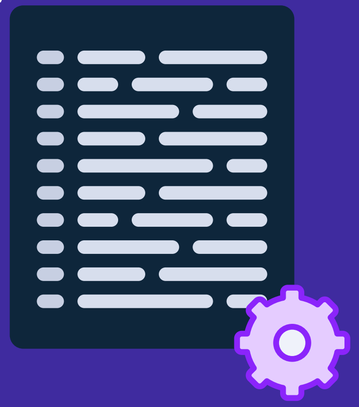


Eternal Storage

0×111 …

Token_v1

0×732 …



Token_v2

0×469 …

👍 Easy redeployment

👍 No data migration needed

👎 Increased complexity

👎 External calls

👎 Careful about breaking integrations

ETERNAL STORAGE

IS THERE A WAY TO UPGRADE WITHOUT *BREAKING* INTEGRATIONS ?

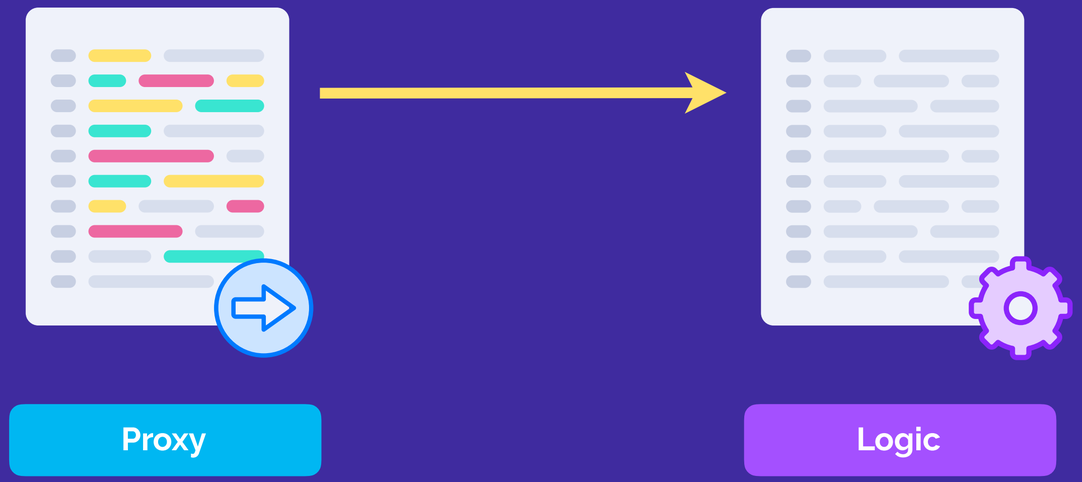
PROXY
CONTRACTS

PROXY PATTERN

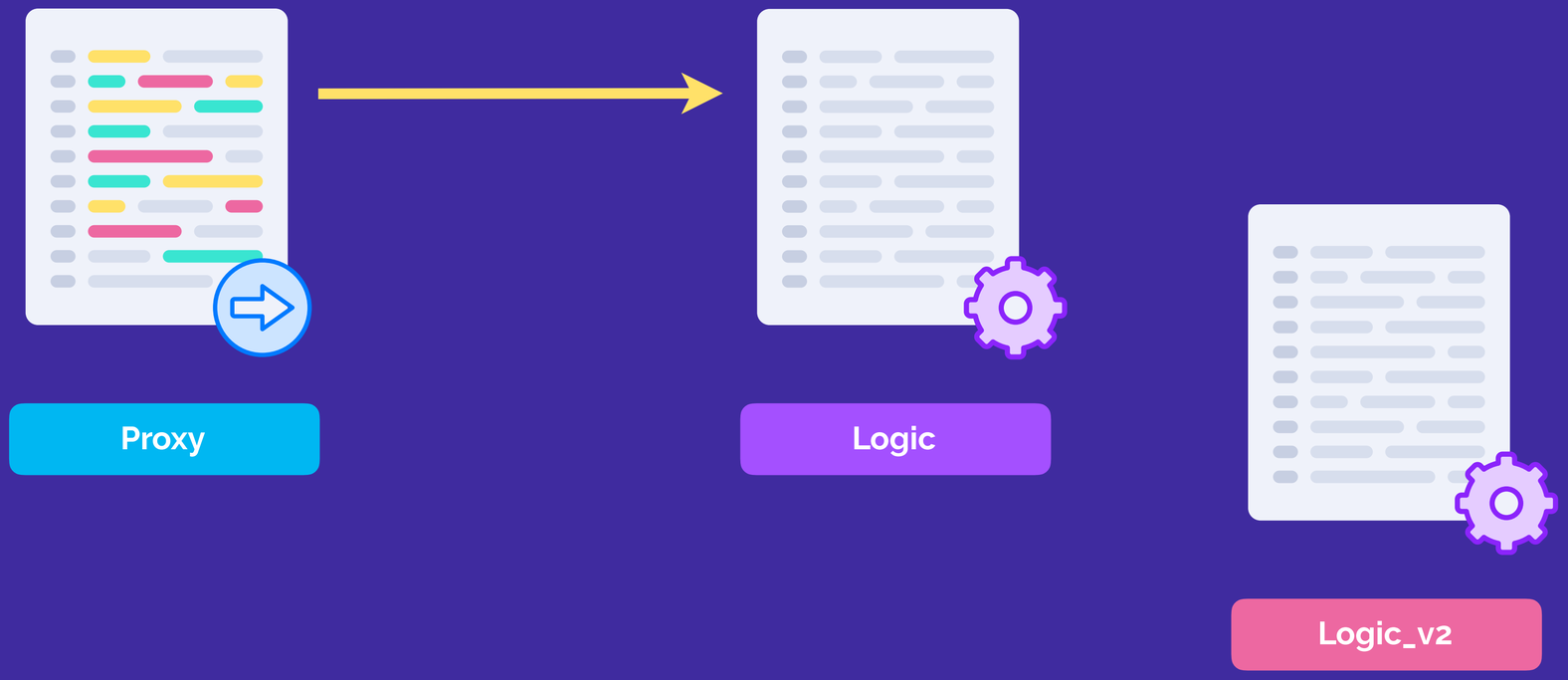
PROXY PATTERN

Proxy

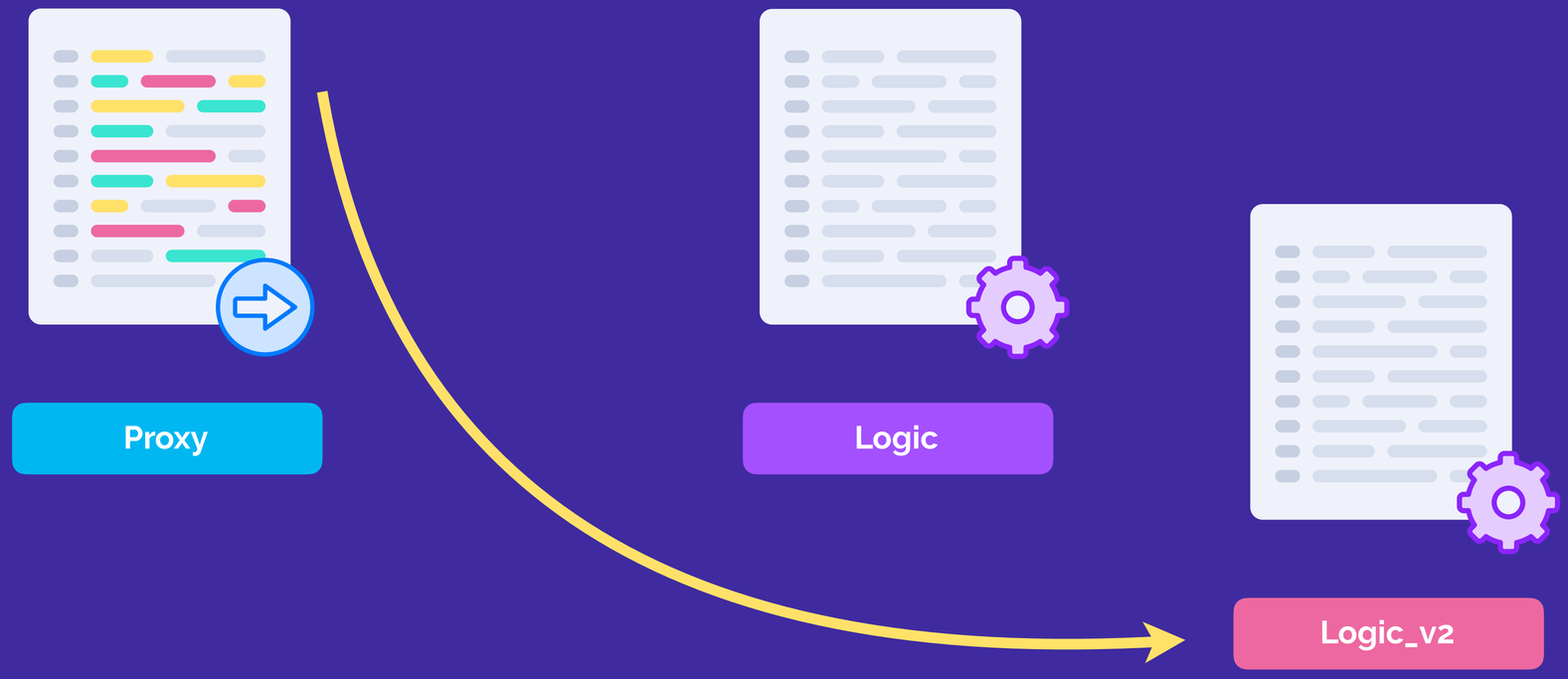
PROXY PATTERN



PROXY PATTERN



PROXY PATTERN



PROXY PATTERN

Proxy

Logic

PROXY PATTERN

Proxy

Logic

PROXY PATTERN

PROXY PATTERN

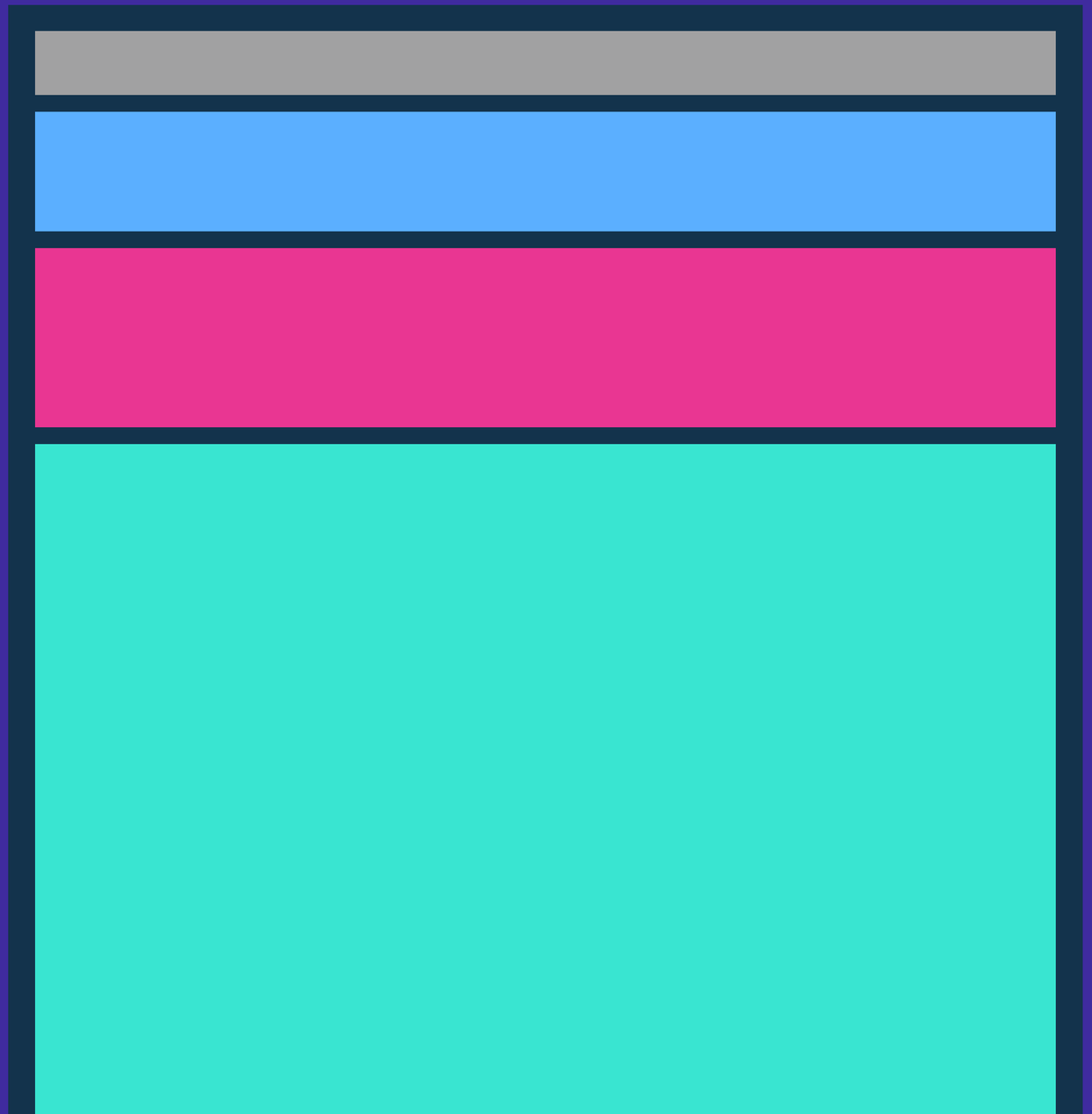
PROXY PATTERN

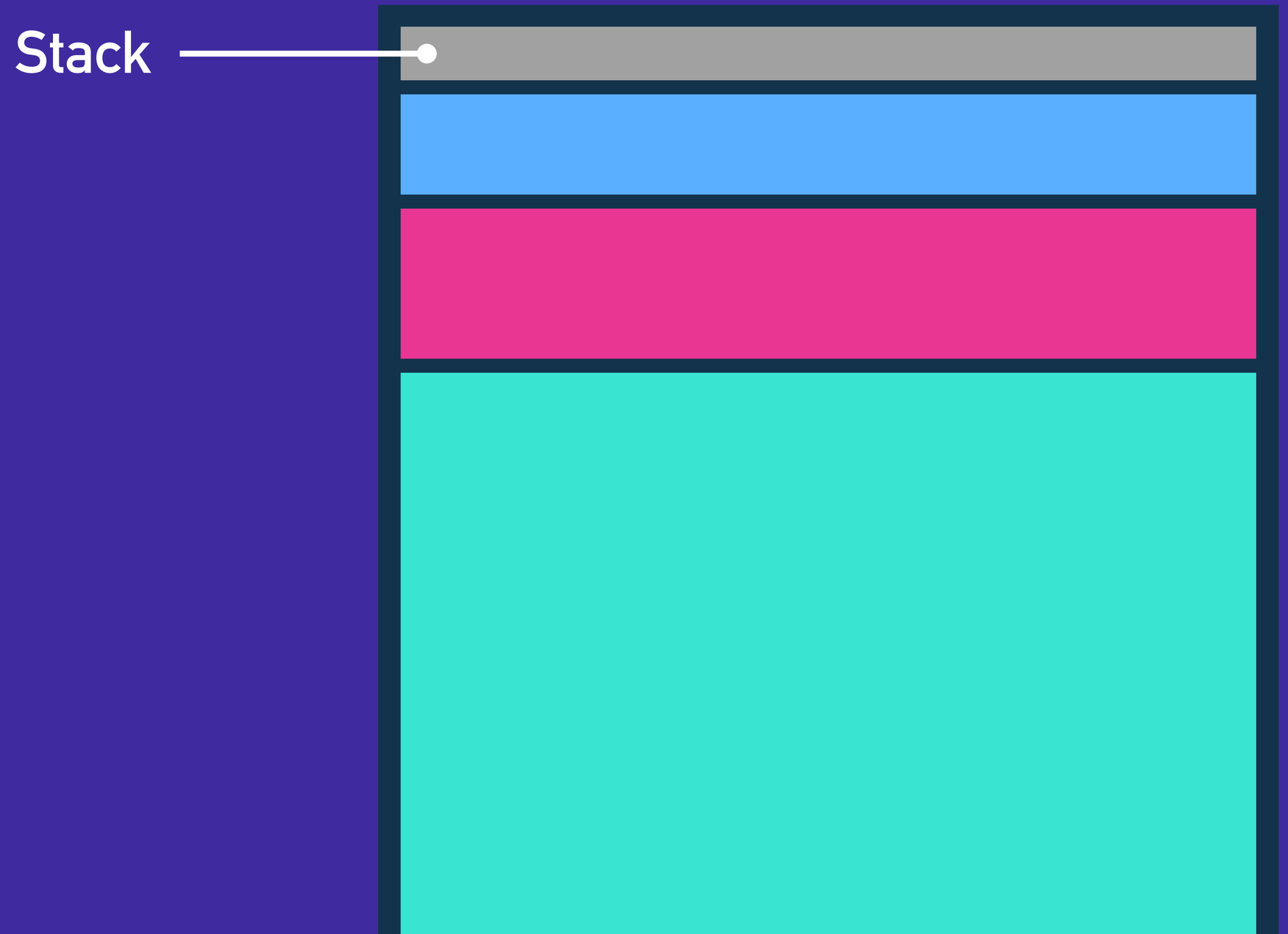
PROXY PATTERN

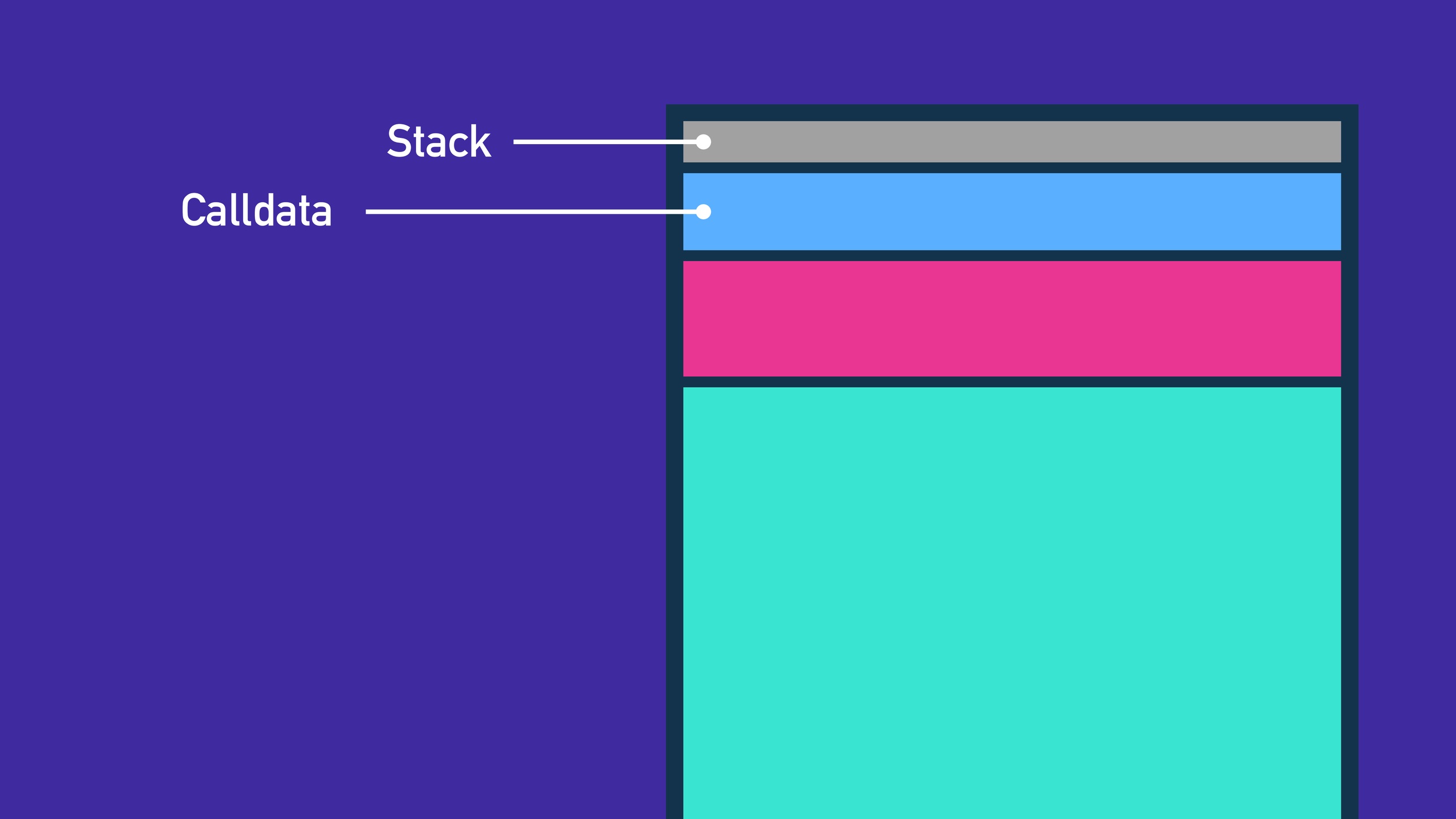
BUT FIRST...

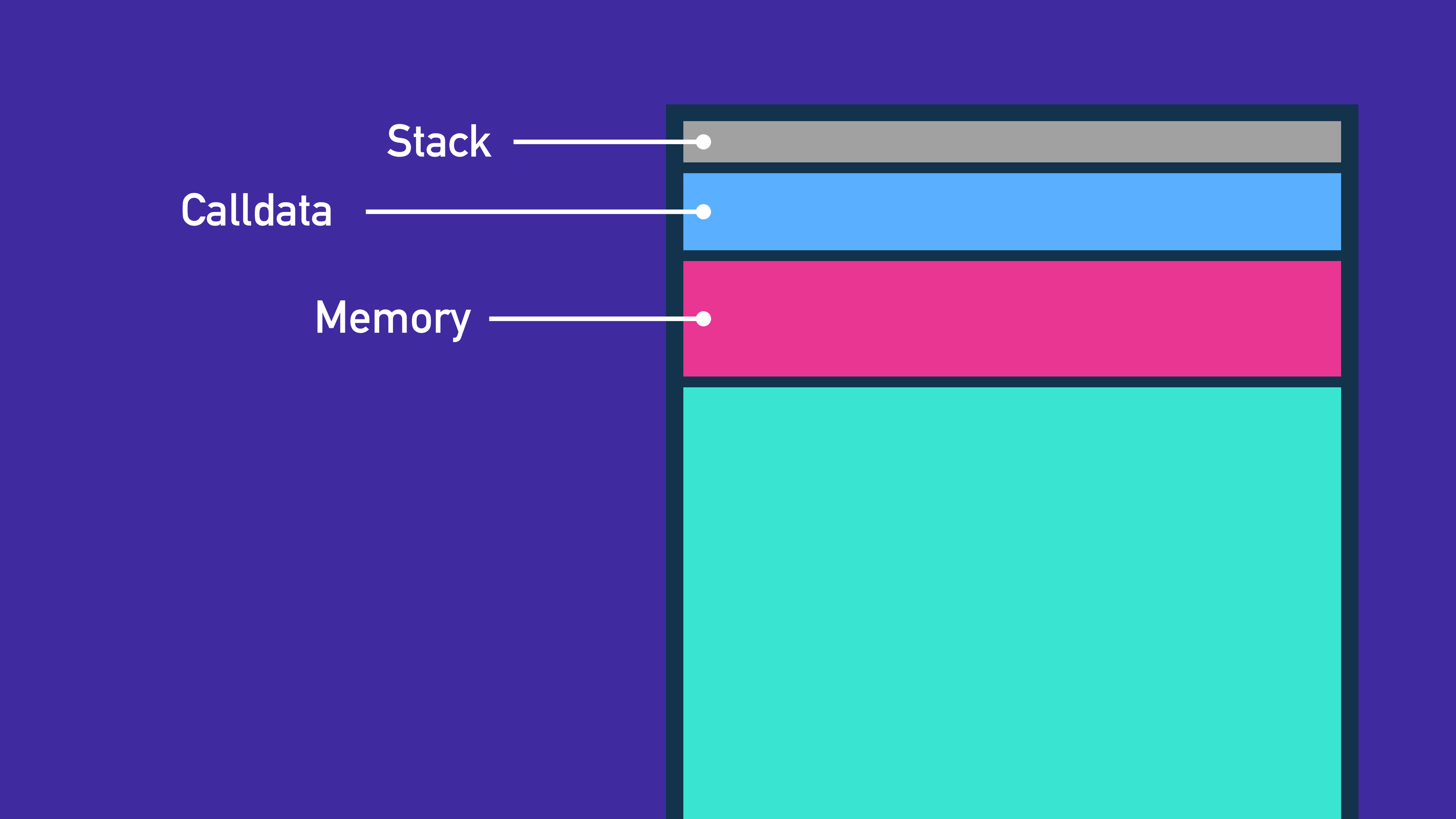
INSIDE THE EVM

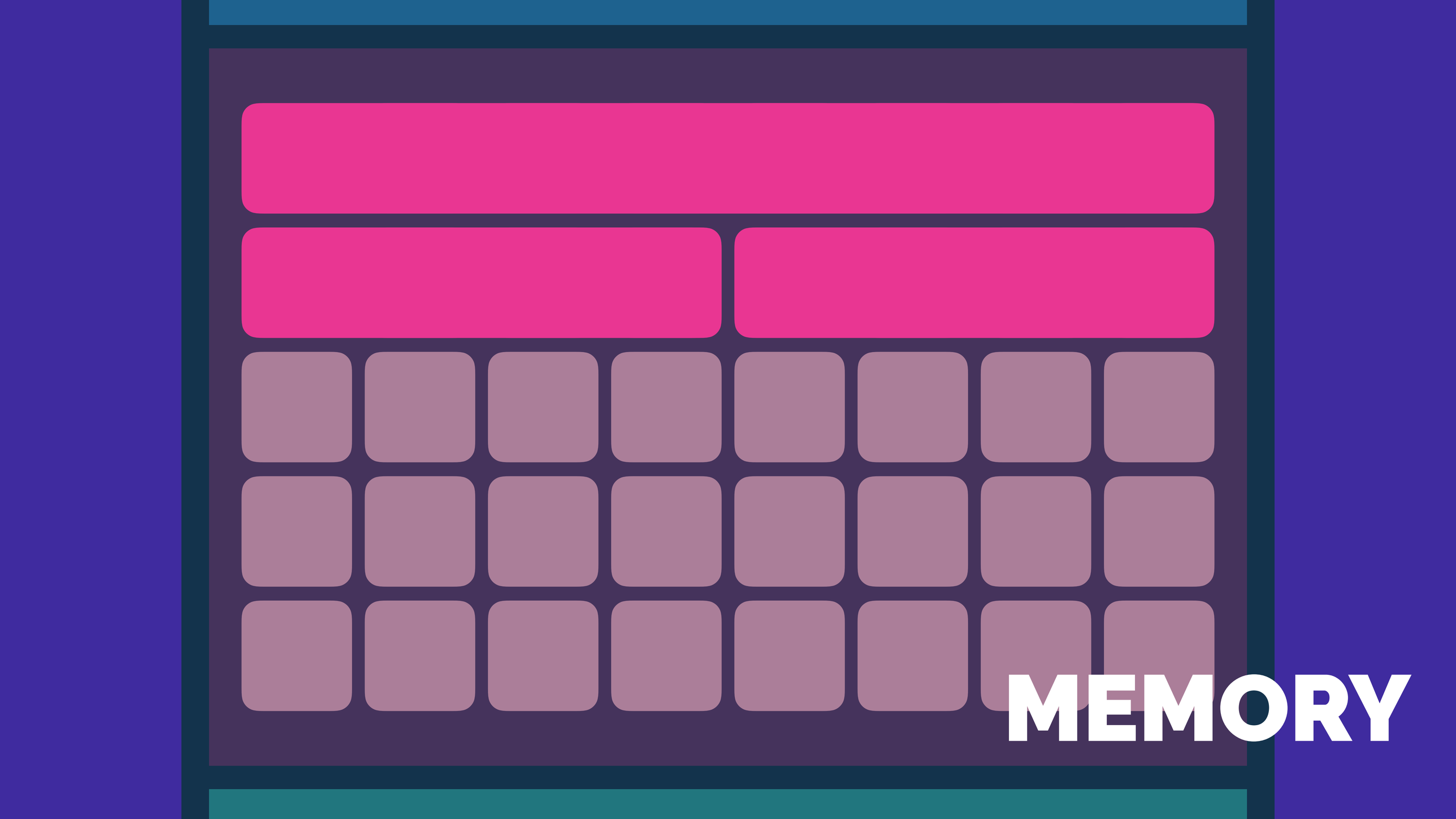
DATA LAYOUT

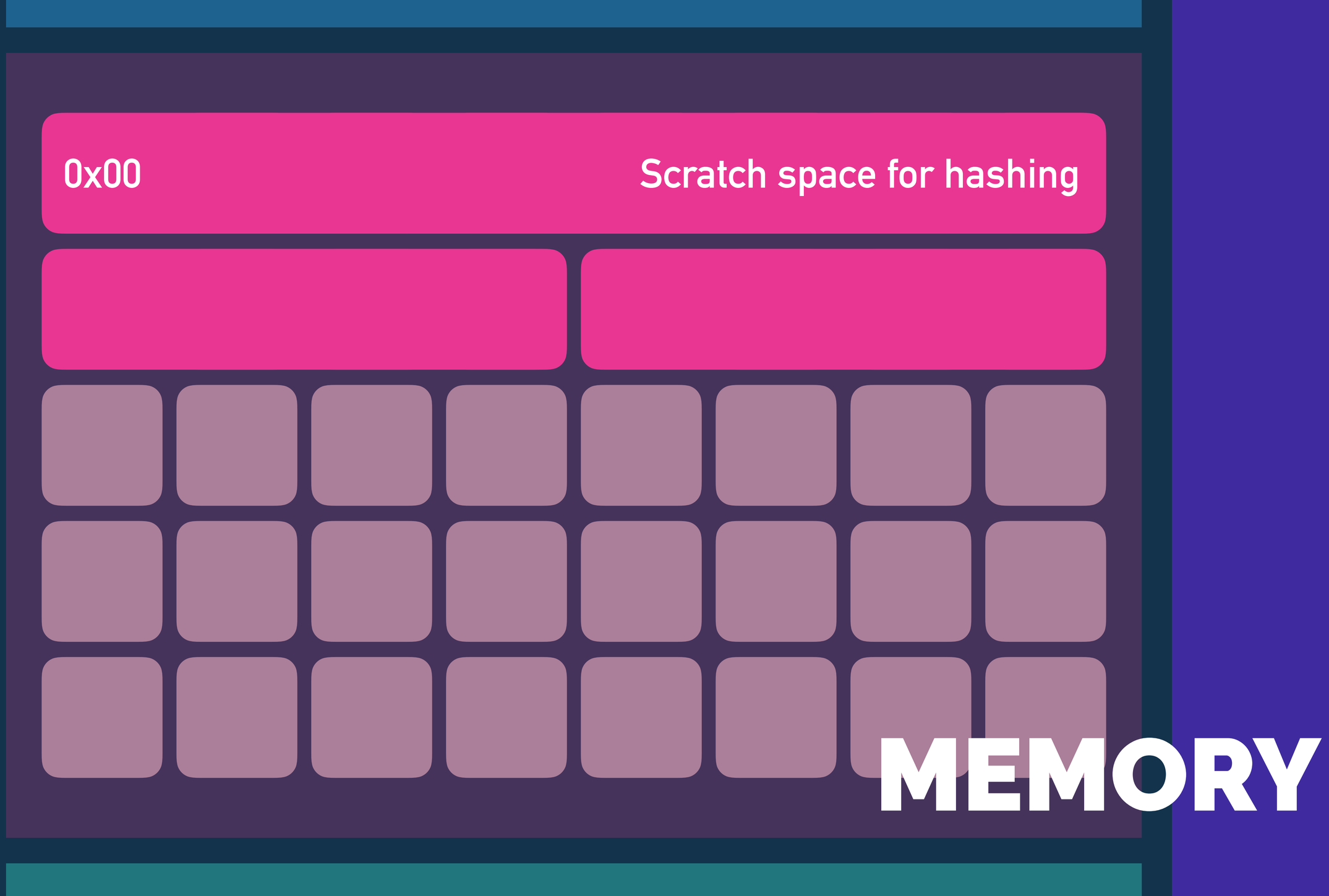


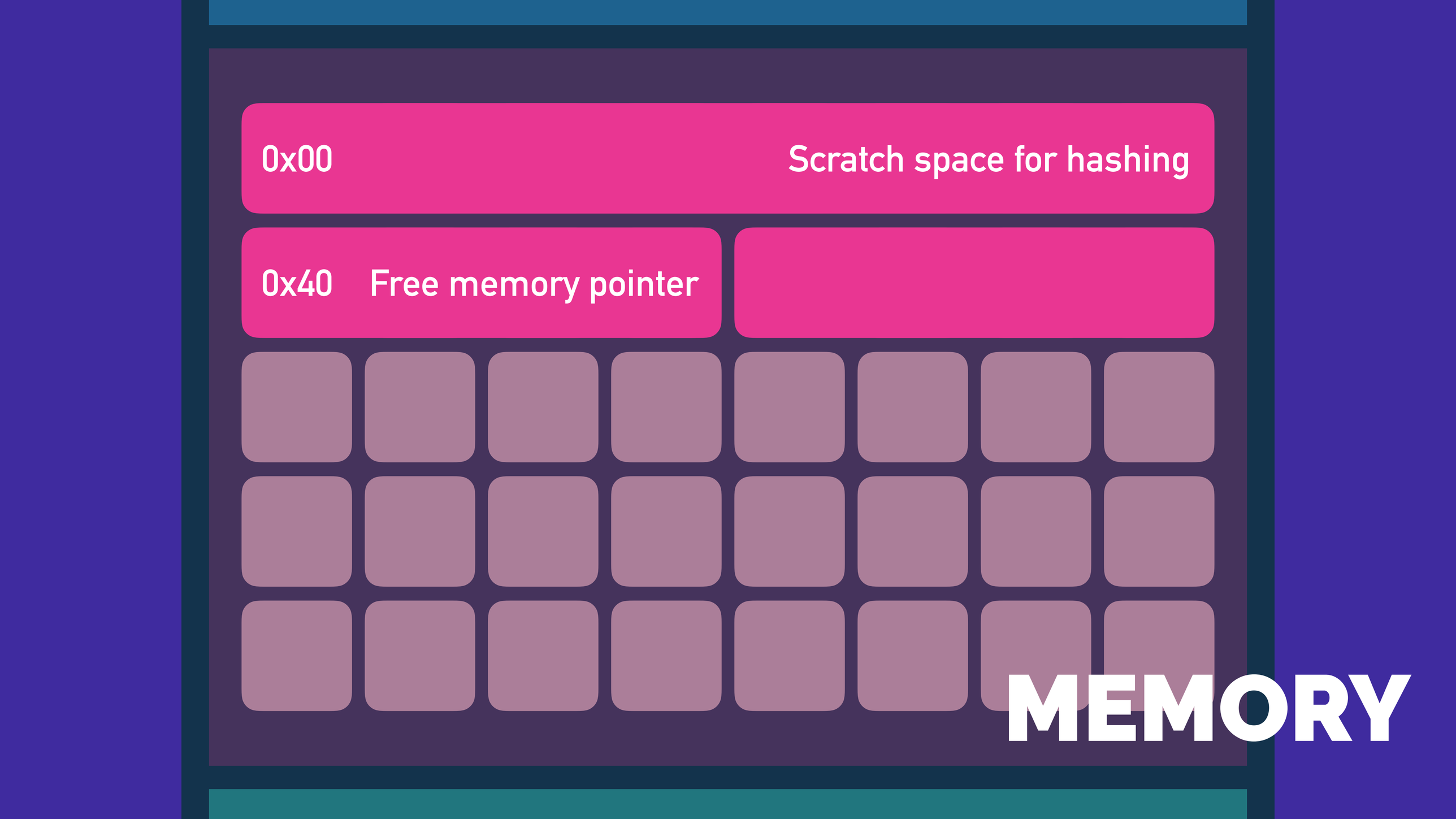

Stack

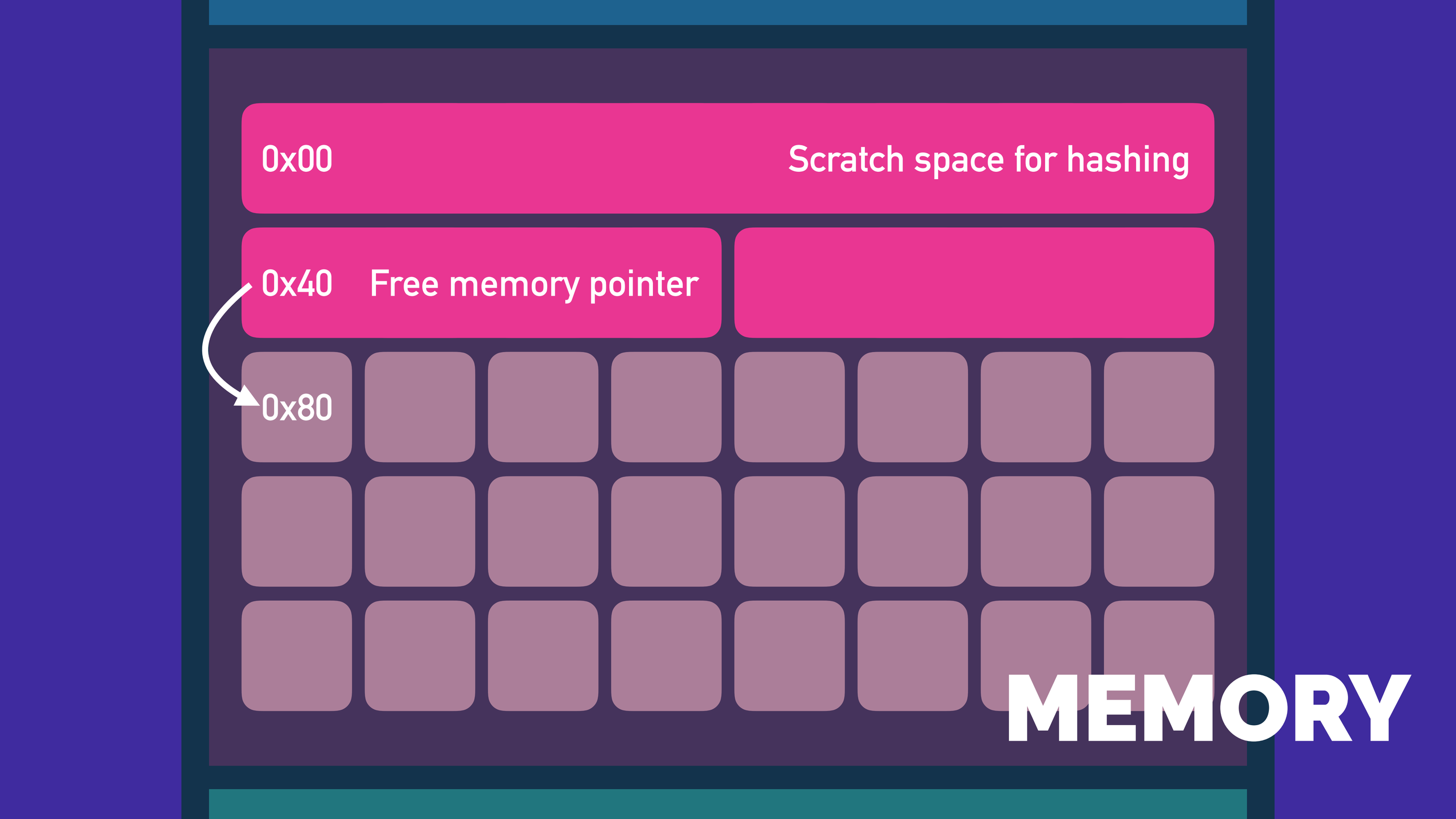

Stack
Calldata

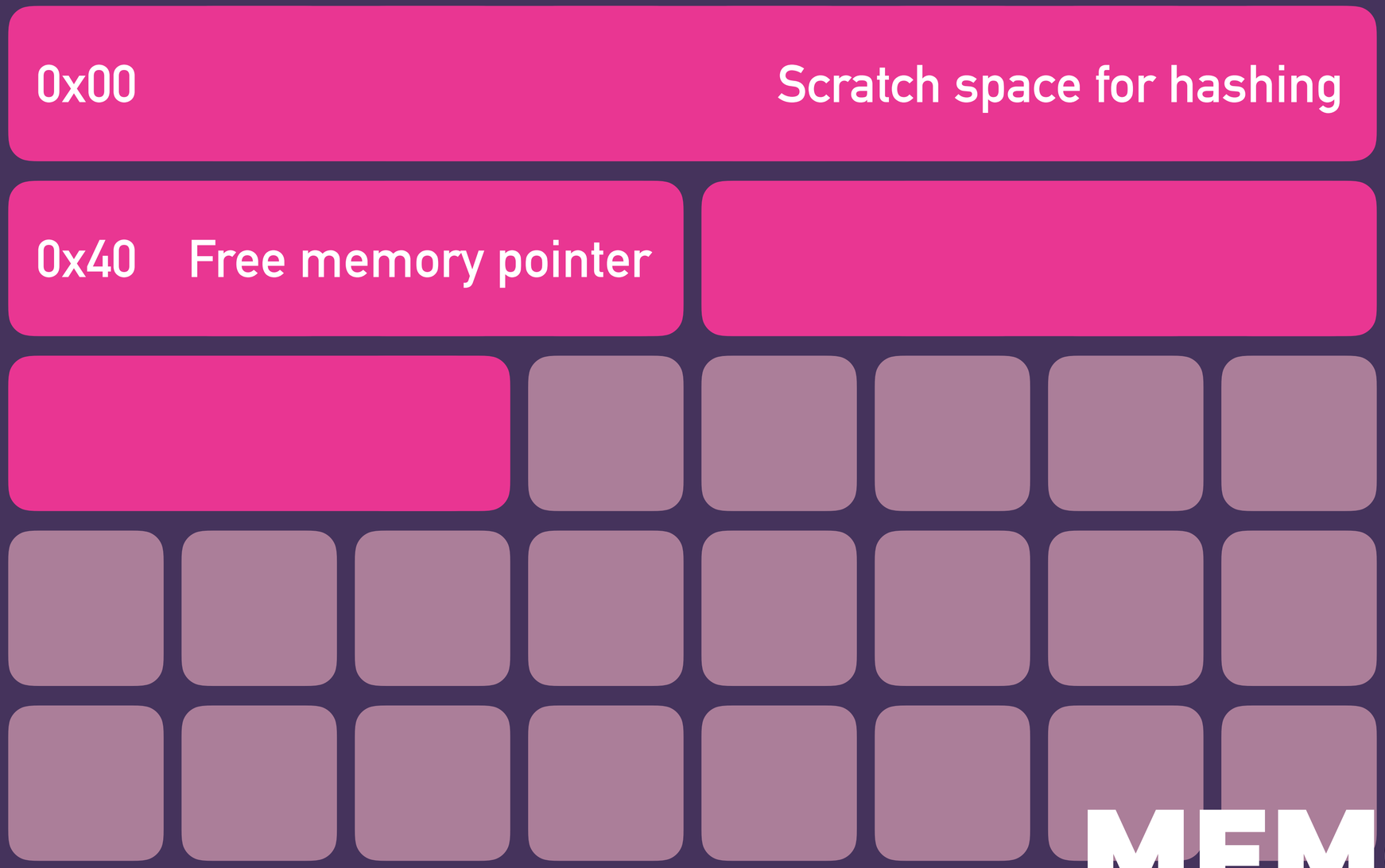

Stack
Calldata
Memory

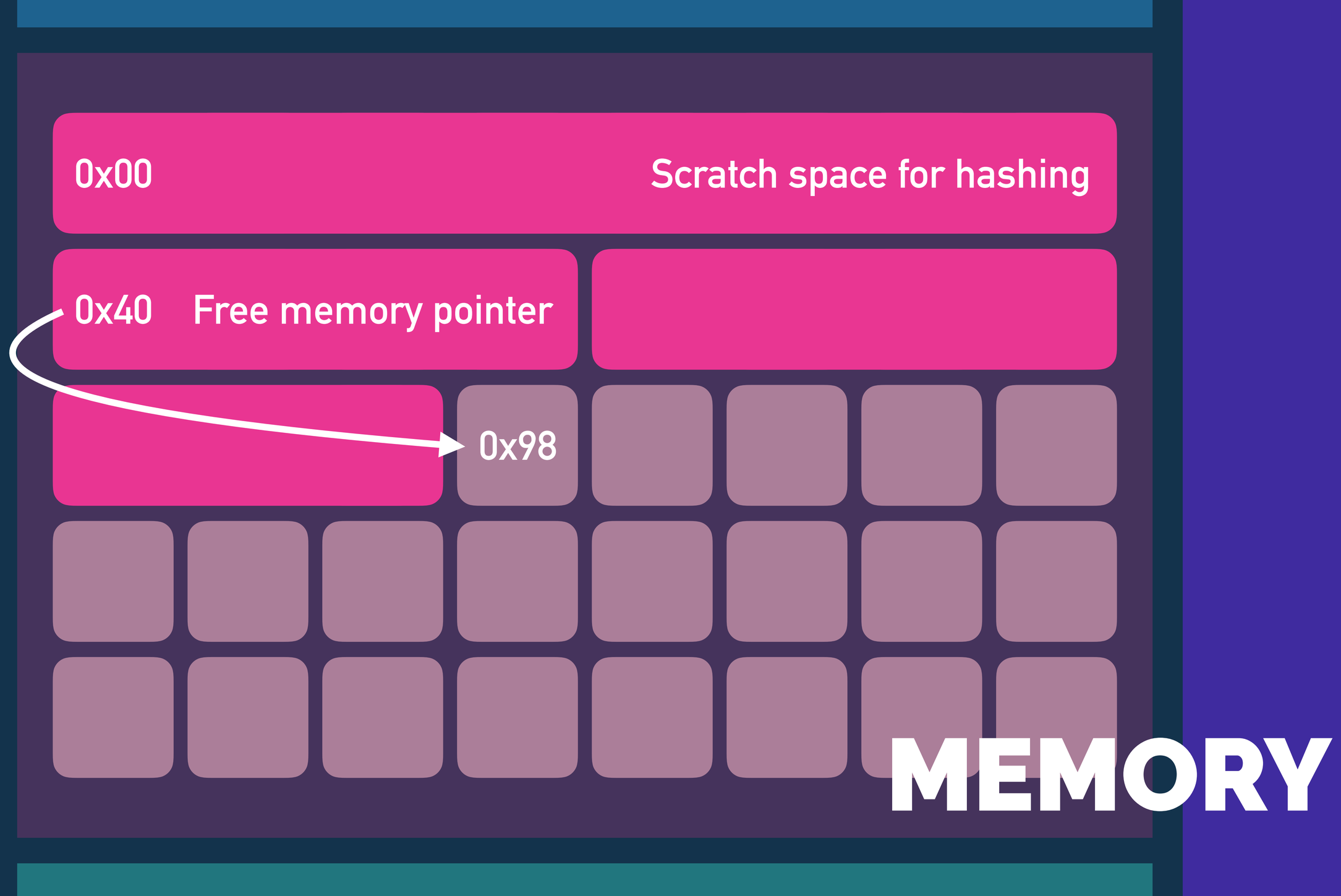

MEMORY

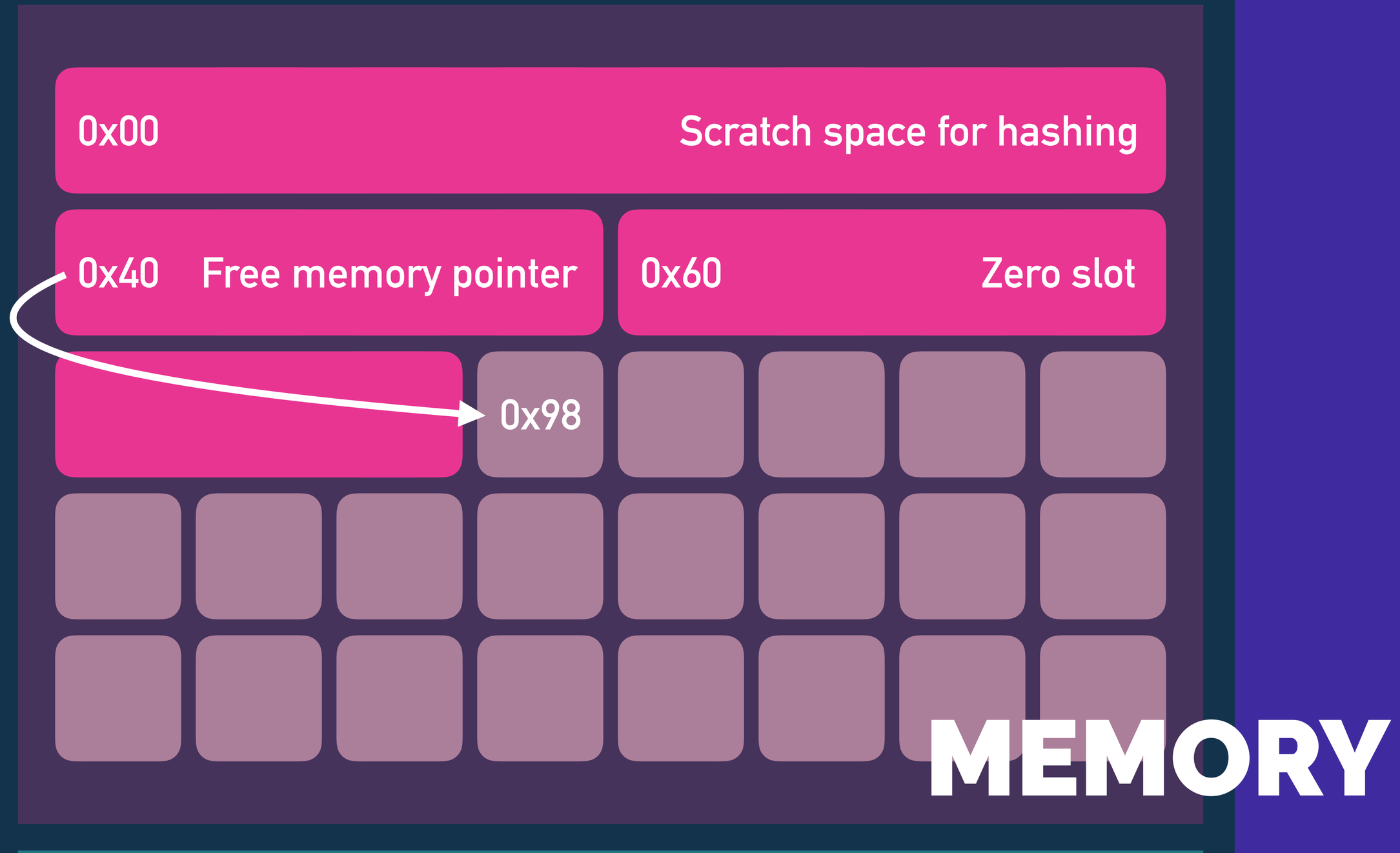

0x00
Scratch space for hashing
MEMORY

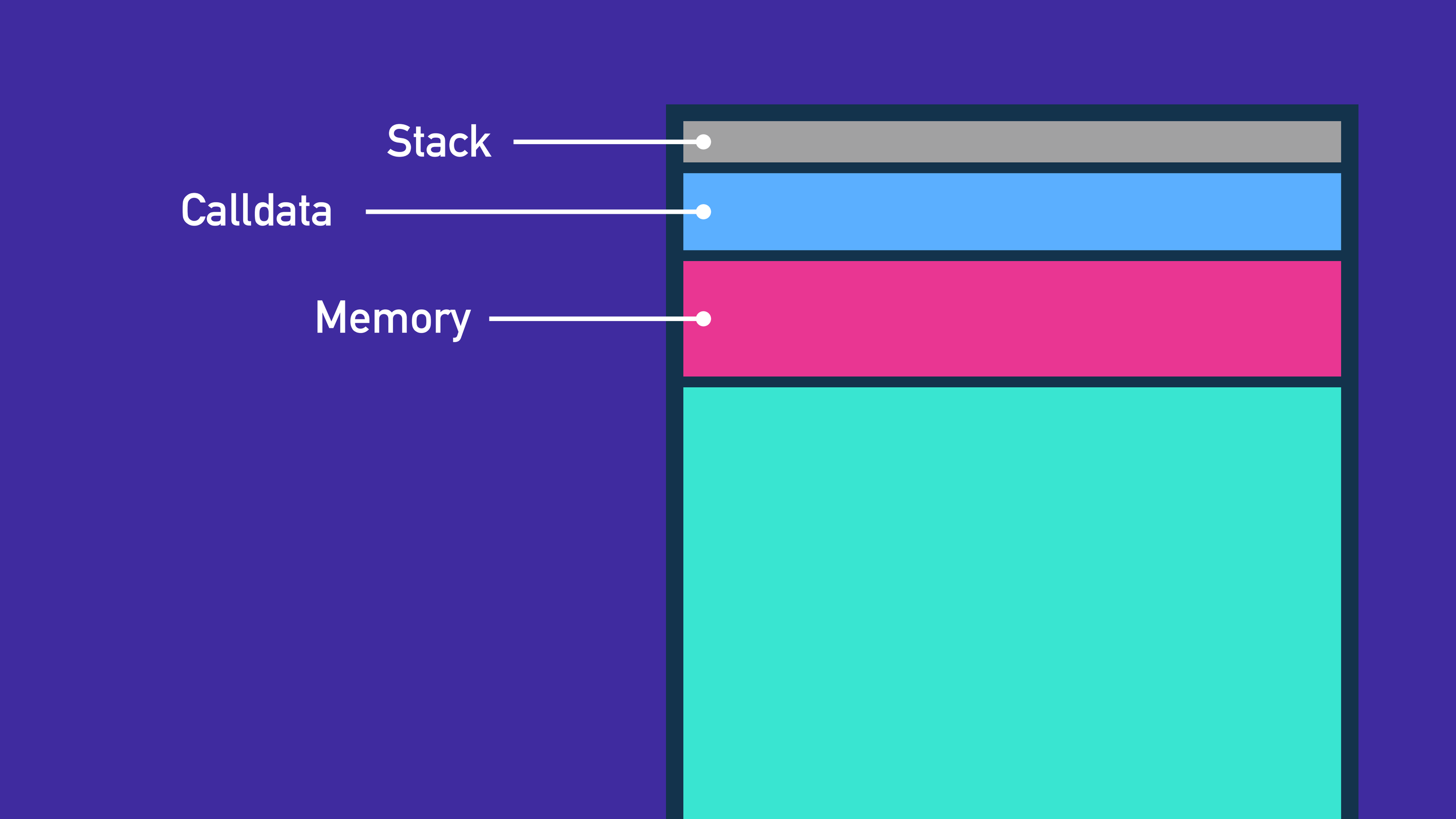

0x00
Scratch space for hashing
0x40
Free memory pointer
MEMORY

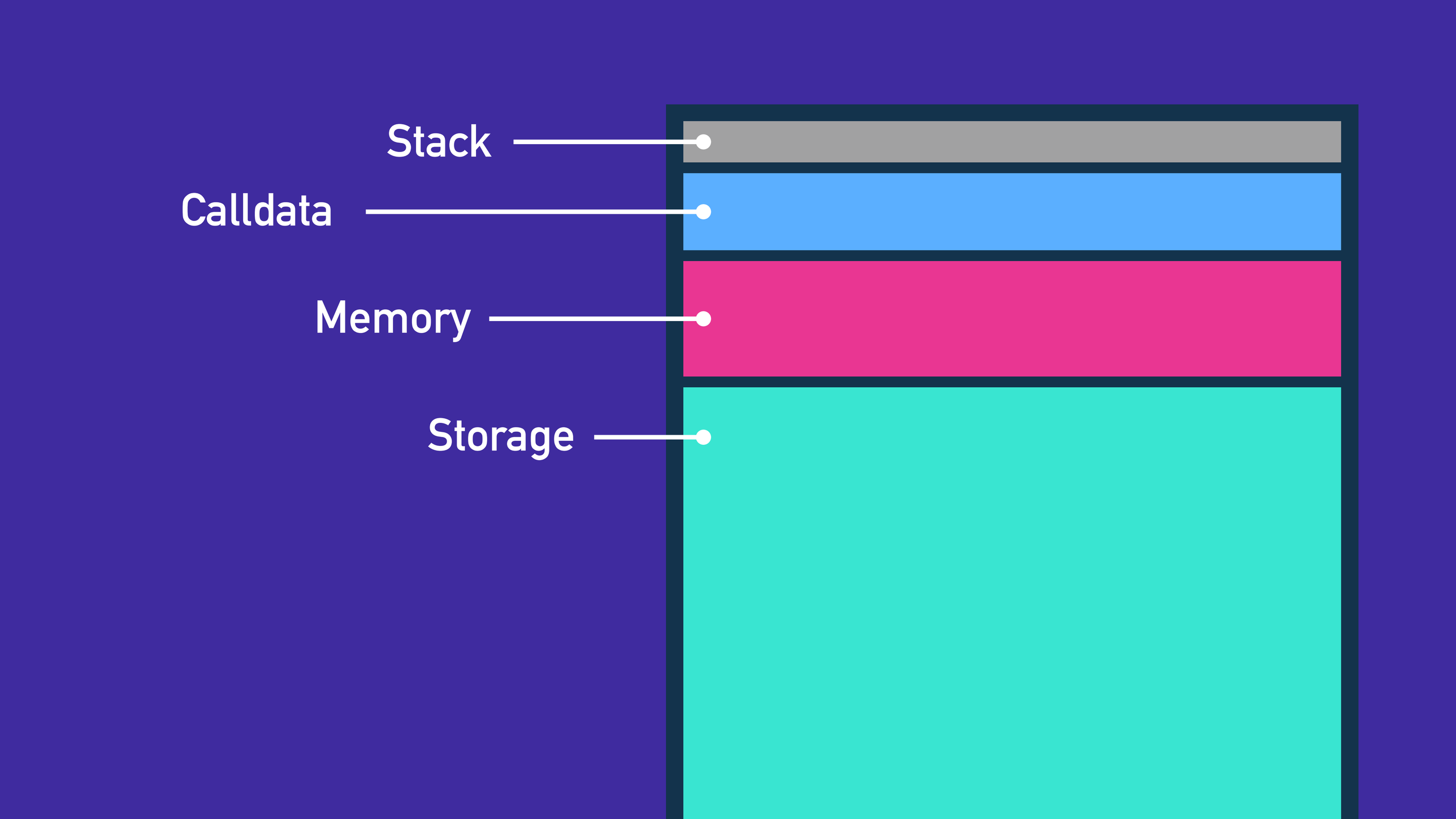

0x00
Scratch space for hashing
0x40
Free memory pointer
0x80
MEMORY

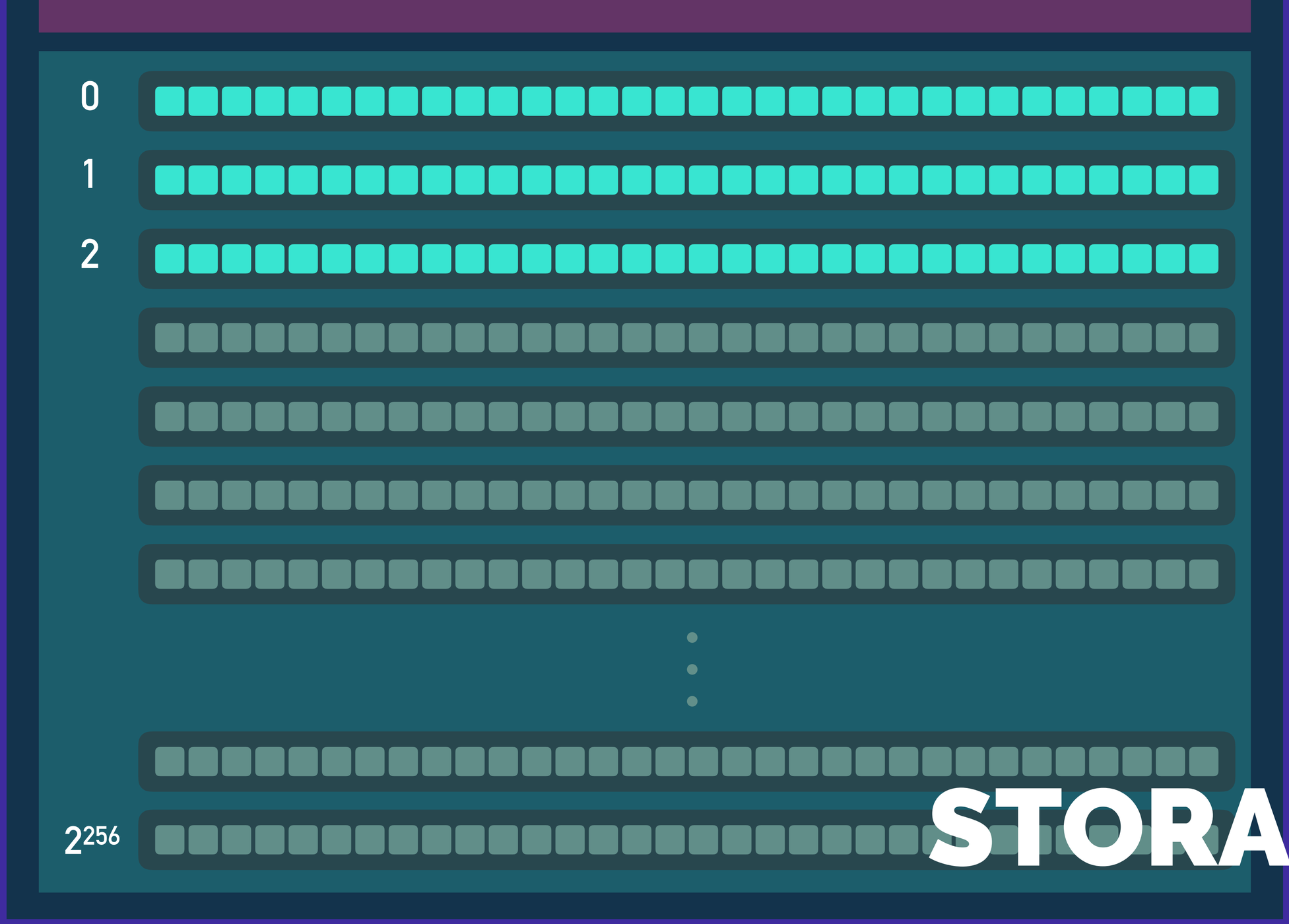

0x00
Scratch space for hashing
0x40
Free memory pointer
MEMORY


0x00
Scratch space for hashing
0x40
Free memory pointer
0x98
MEMORY

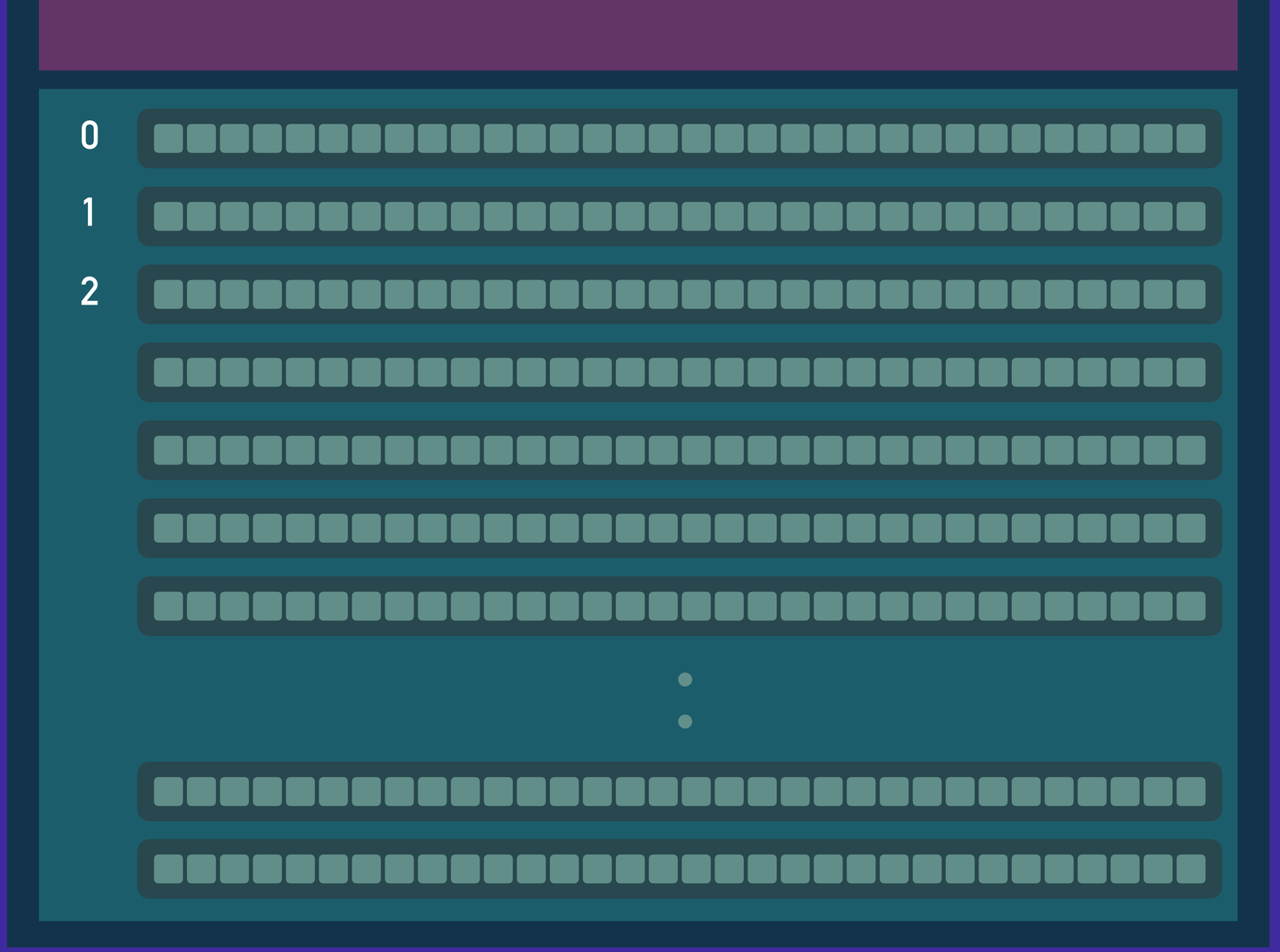

0x00
Scratch space for hashing
0x40
Free memory pointer
0x60
Zero slot
0x98
MEMORY


Stack
Calldata
Memory


Stack
Calldata
Memory
Storage

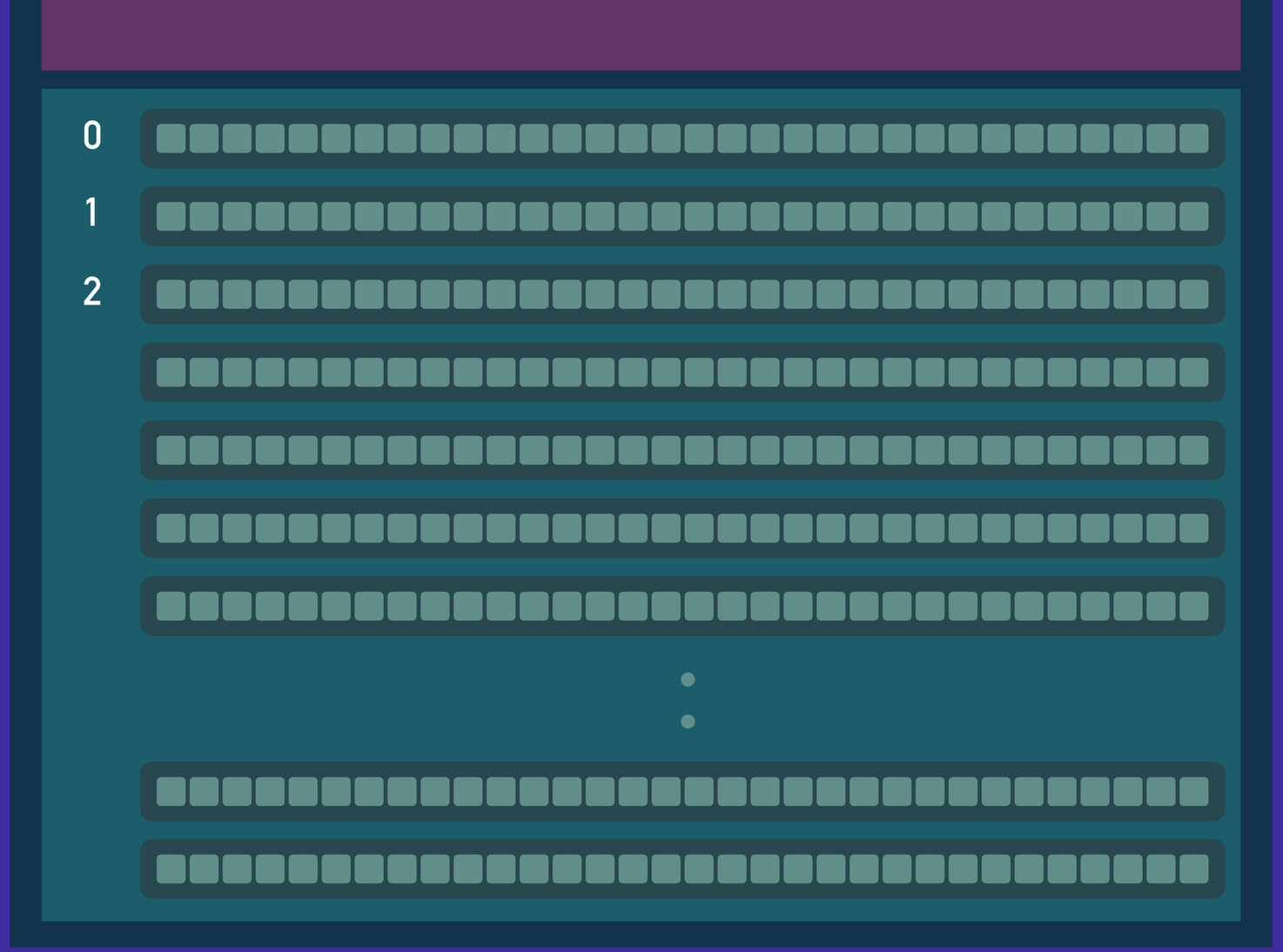

0
1
2
2^{256}
STORAGE

```
contract Token {




}
```



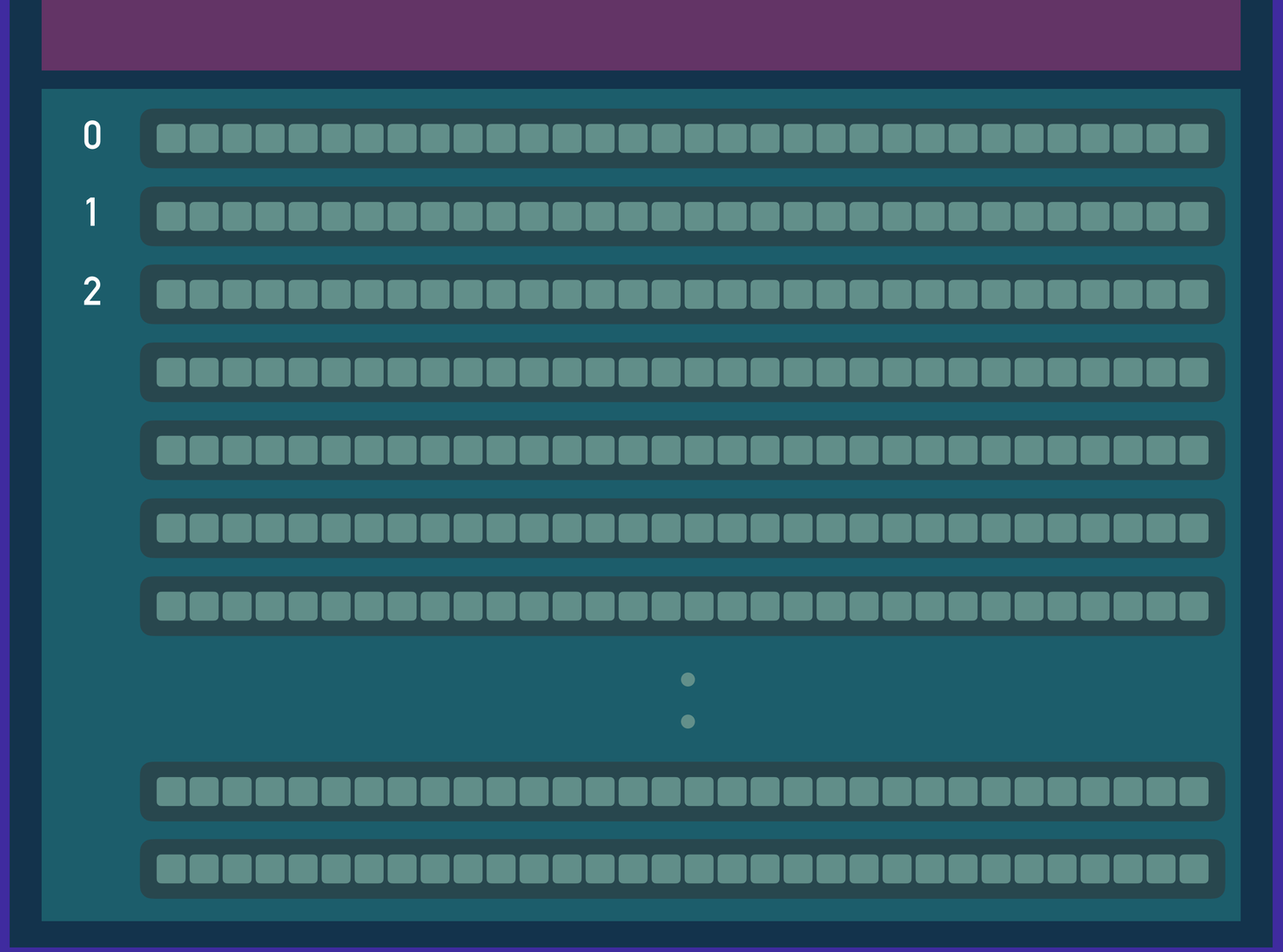
STORAGE

```
contract Token {


bool isActive = true;









}
```



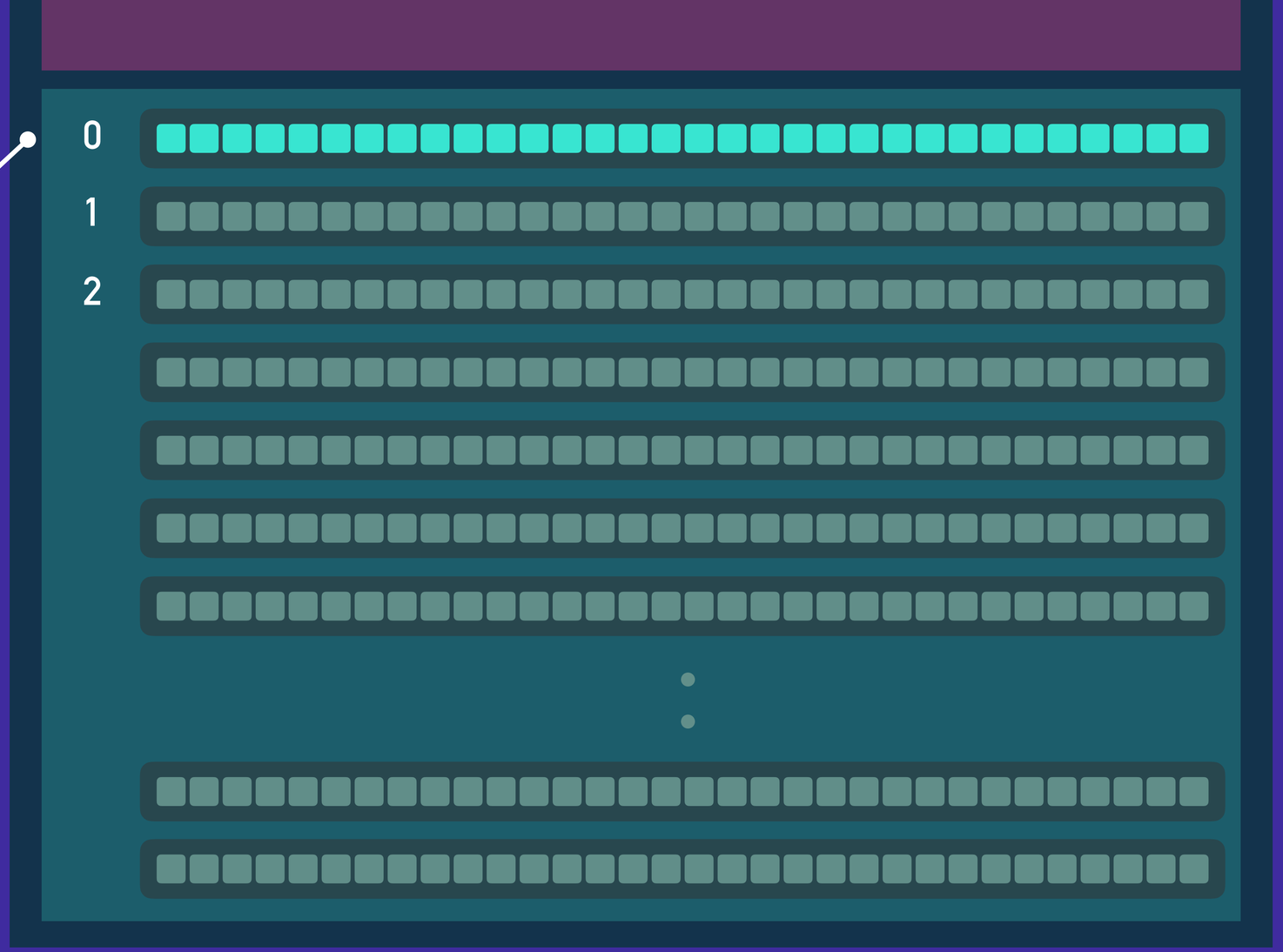
STORAGE

```
contract Token {


0   bool isActive = true;









}
```



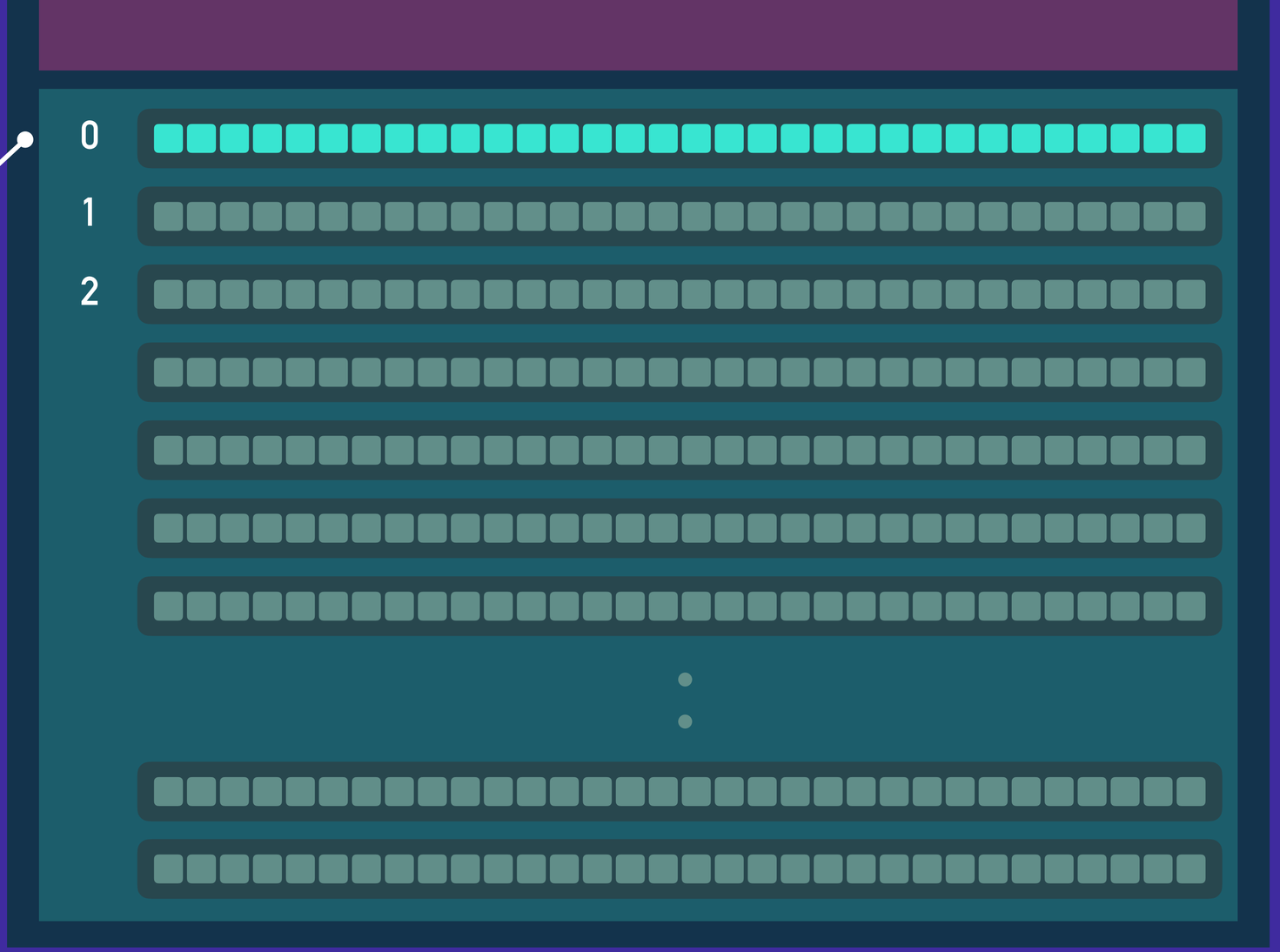
STORAGE

```
contract Token {


0   bool isActive = true;


}
```



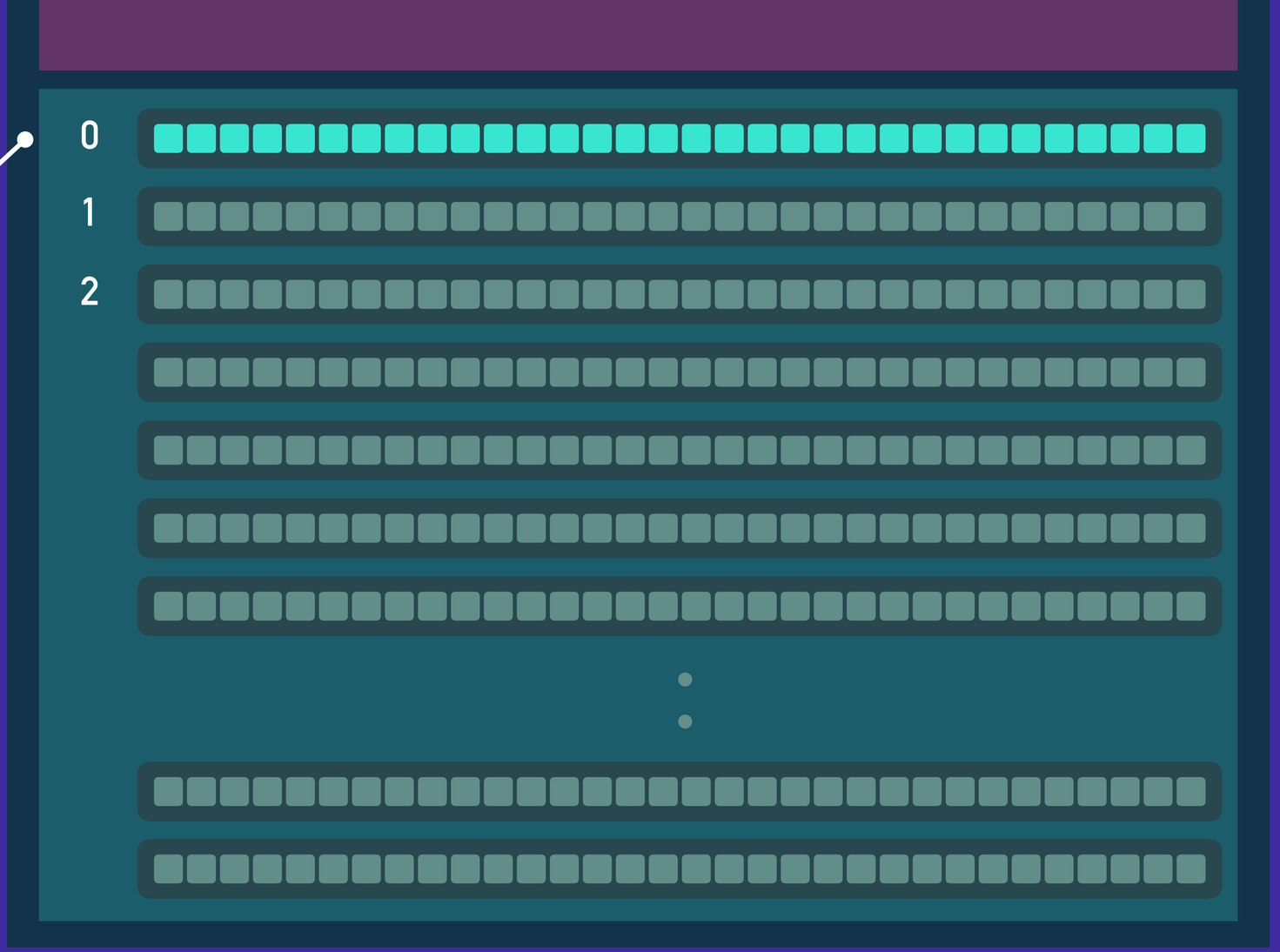
STORAGE

```
contract Token {


0   bool isActive = true;

    address owner = 0x..;









}
```



STORAGE

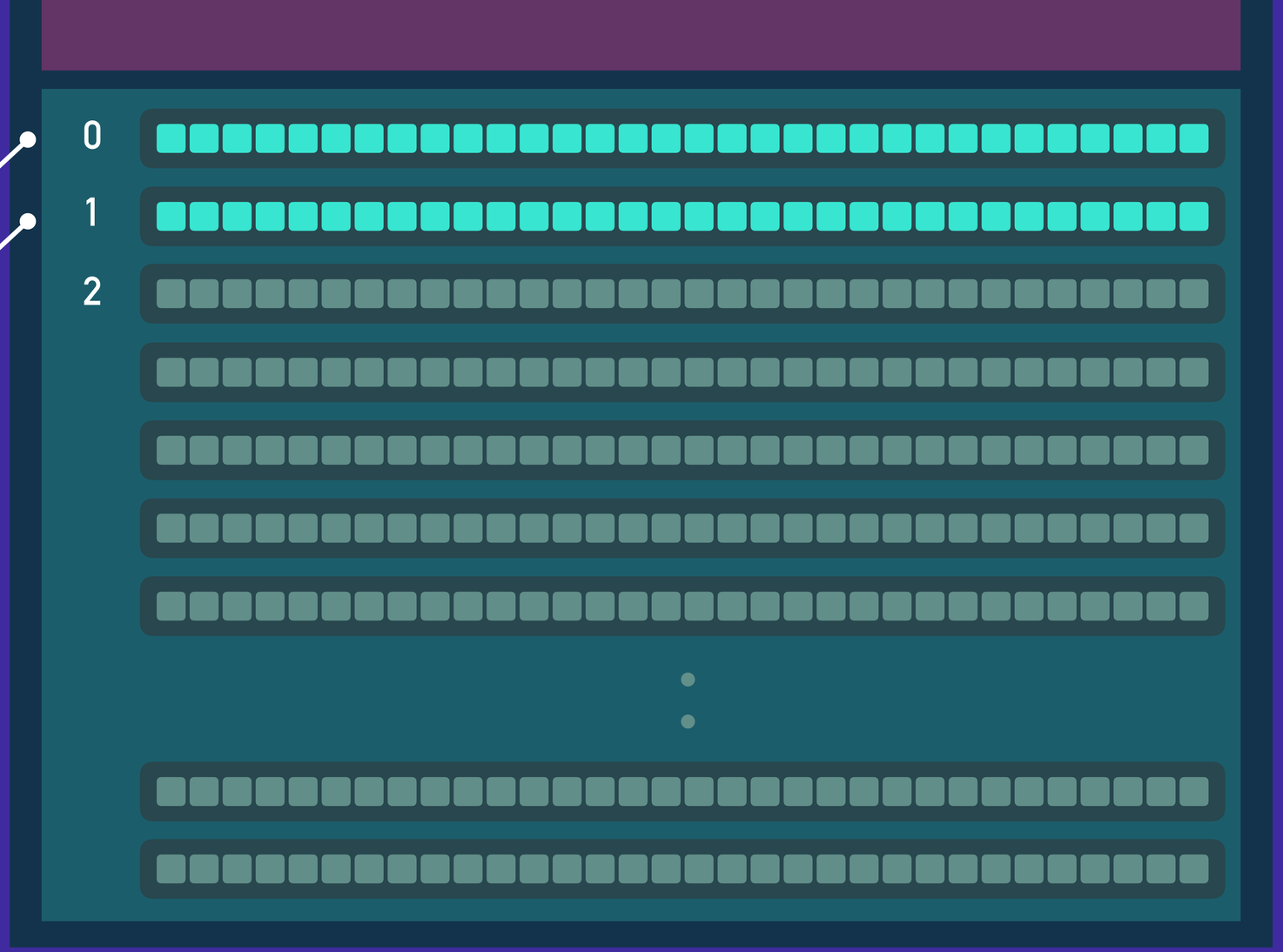
```
contract Token {


0   bool isActive = true;
1   address owner = 0x..;









}
```



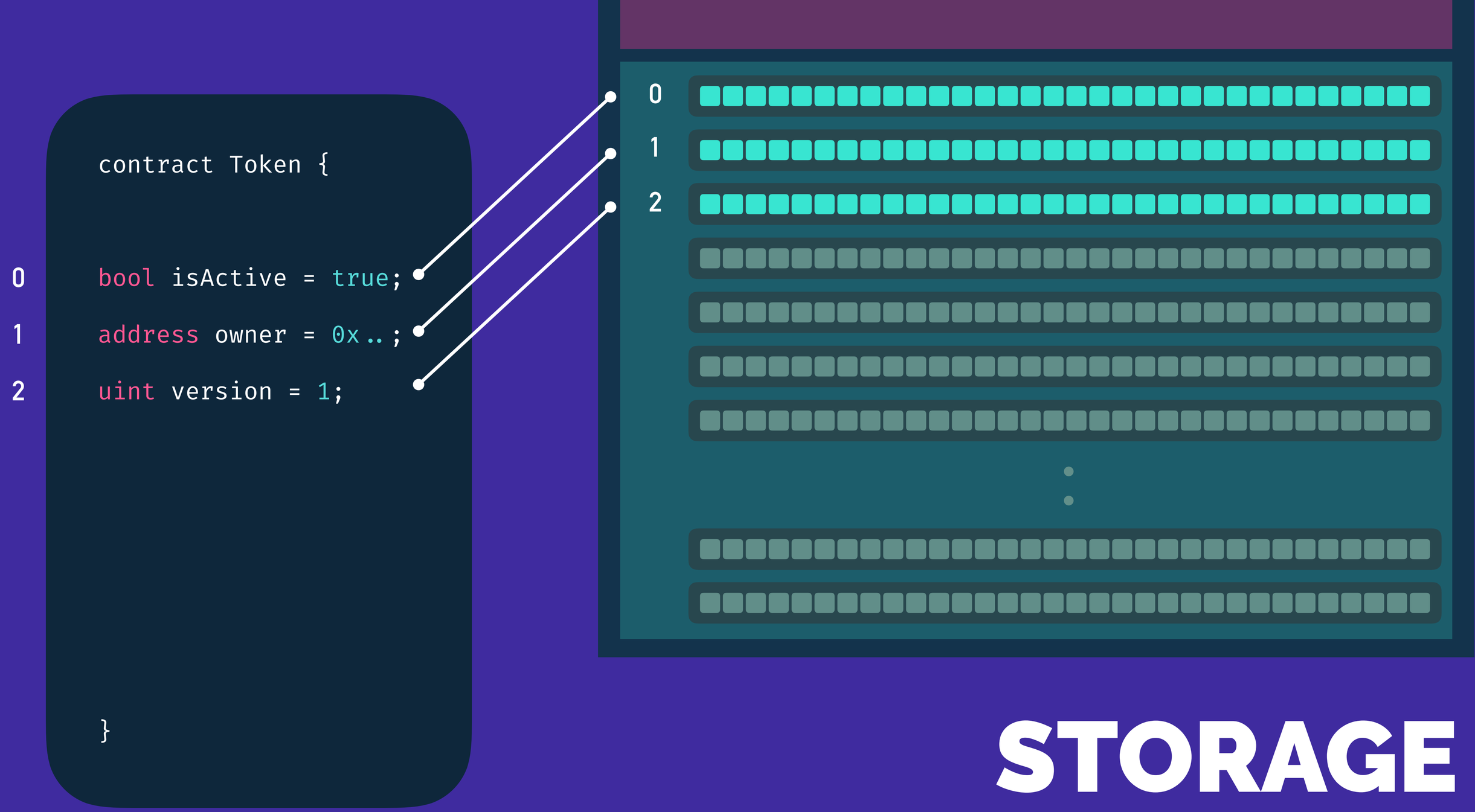
STORAGE

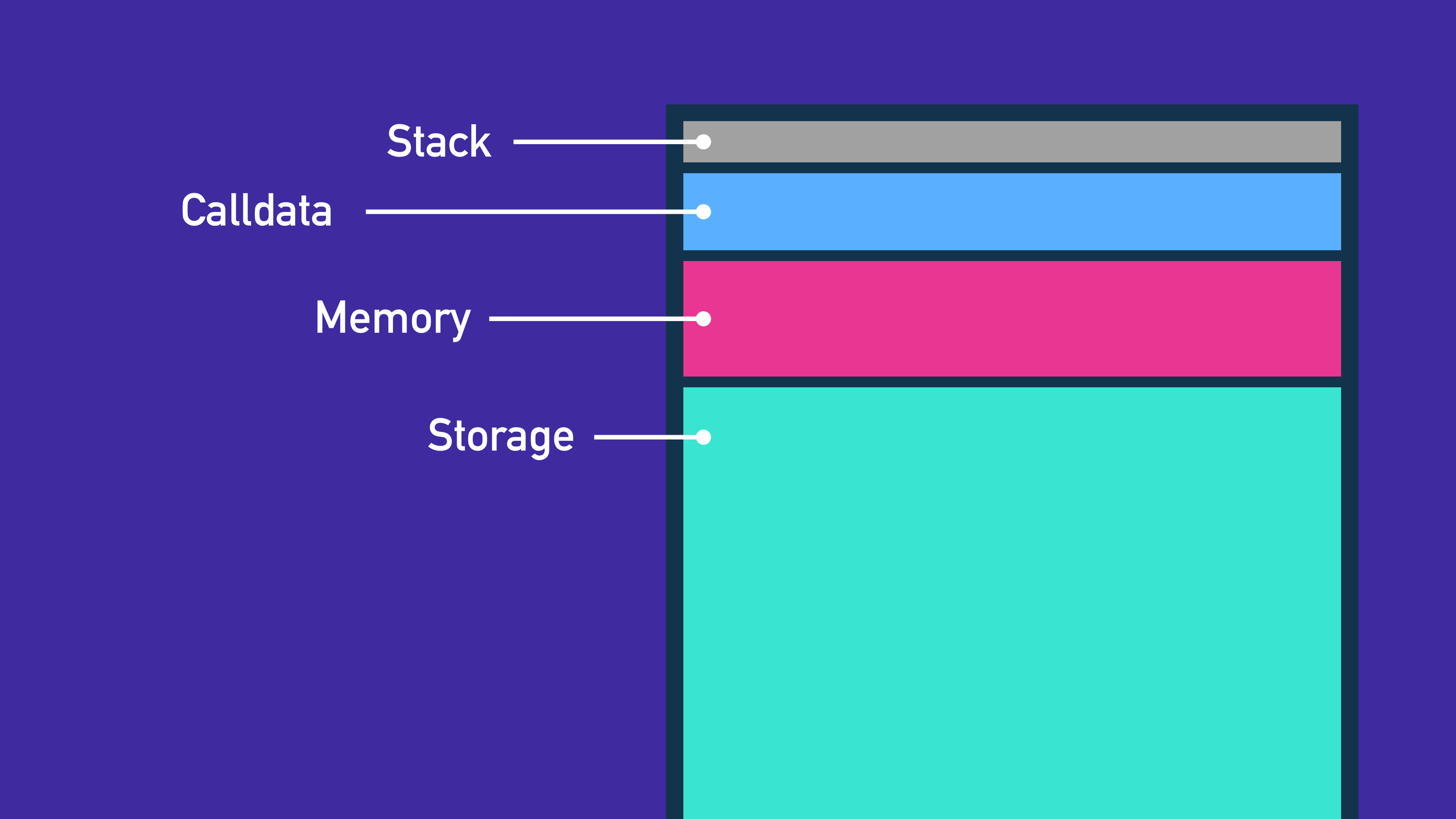

contract Token {
0
bool isActive = true;
1
address owner = 0x..;
}
0
1
2
STORAGE

```
contract Token {


0   bool isActive = true;
1   address owner = 0x..;
    uint version = 1;









}
```



STORAGE

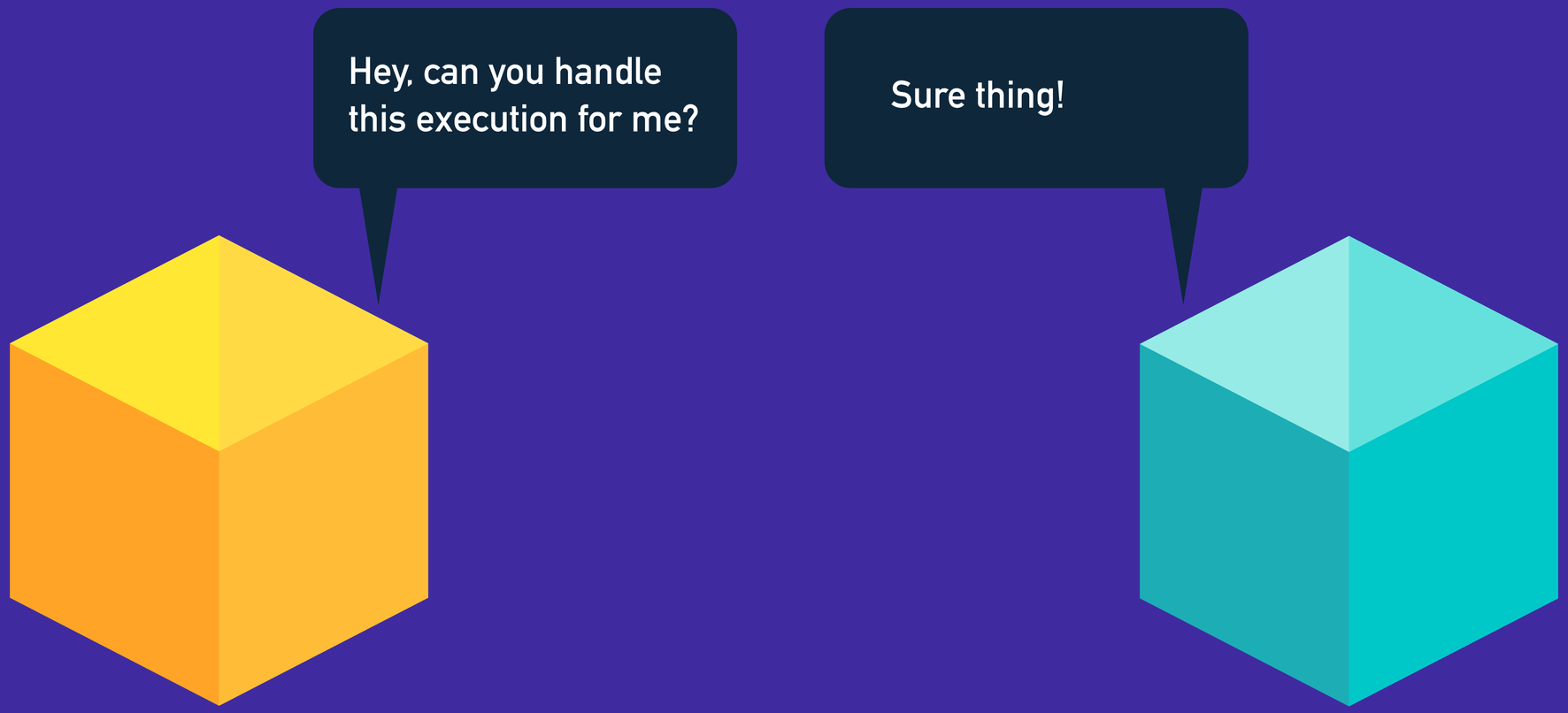

contract Token {
0
bool isActive = true;
1
address owner = 0x..;
2
uint version = 1;
}
0
1
2
STORAGE


Stack
Calldata
Memory
Storage


I am
Sold

INSIDE THE EVM

DELEGATECALL


Hey, can you handle this execution for me?
Sure thing!

CALL

Context A

Token

`getIsActive()`

CALL

Context B

Data Store

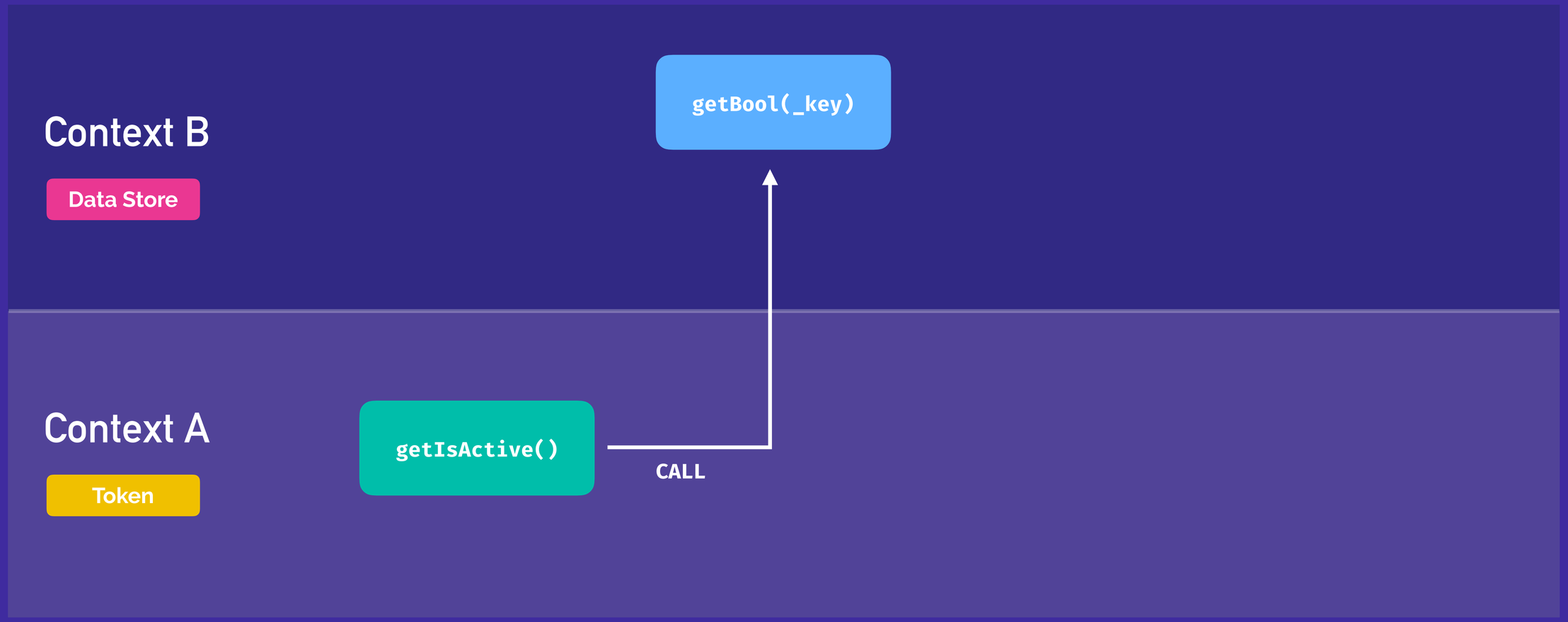
`getBool(_key)`

Context A

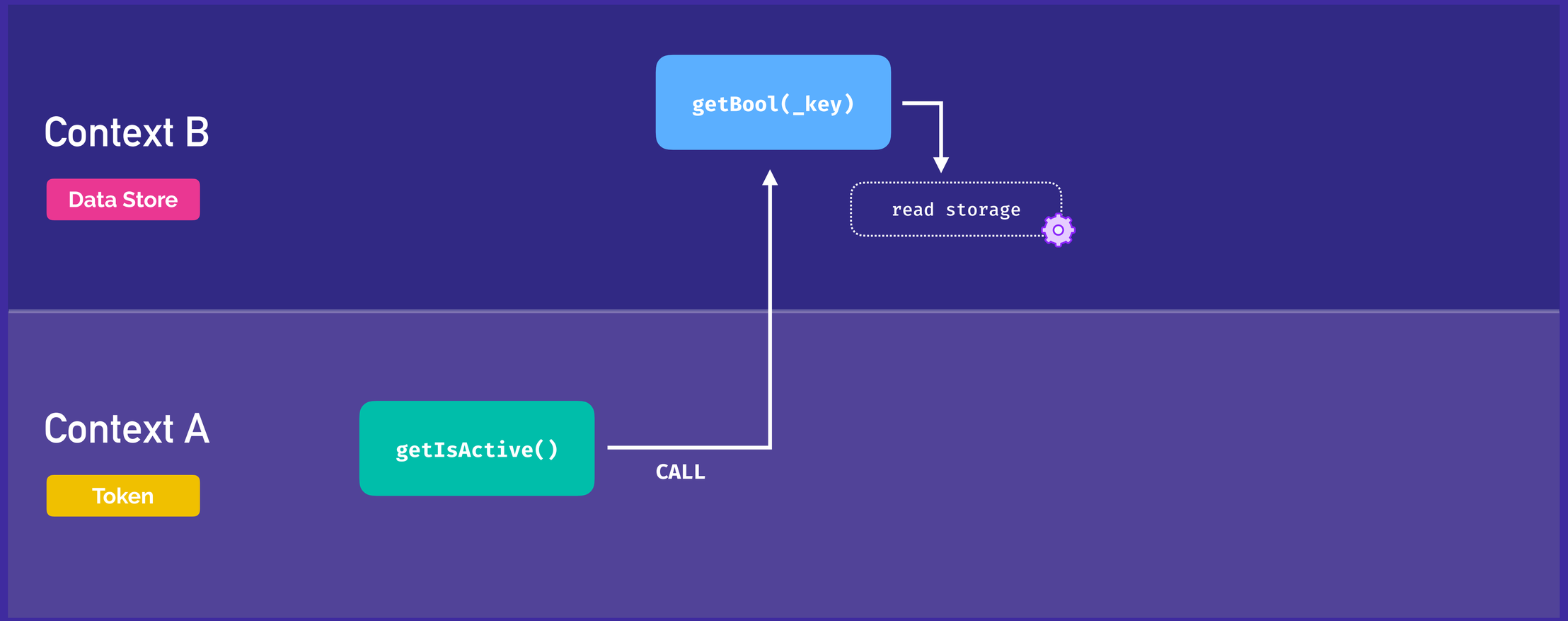
Token

`getIsActive()`

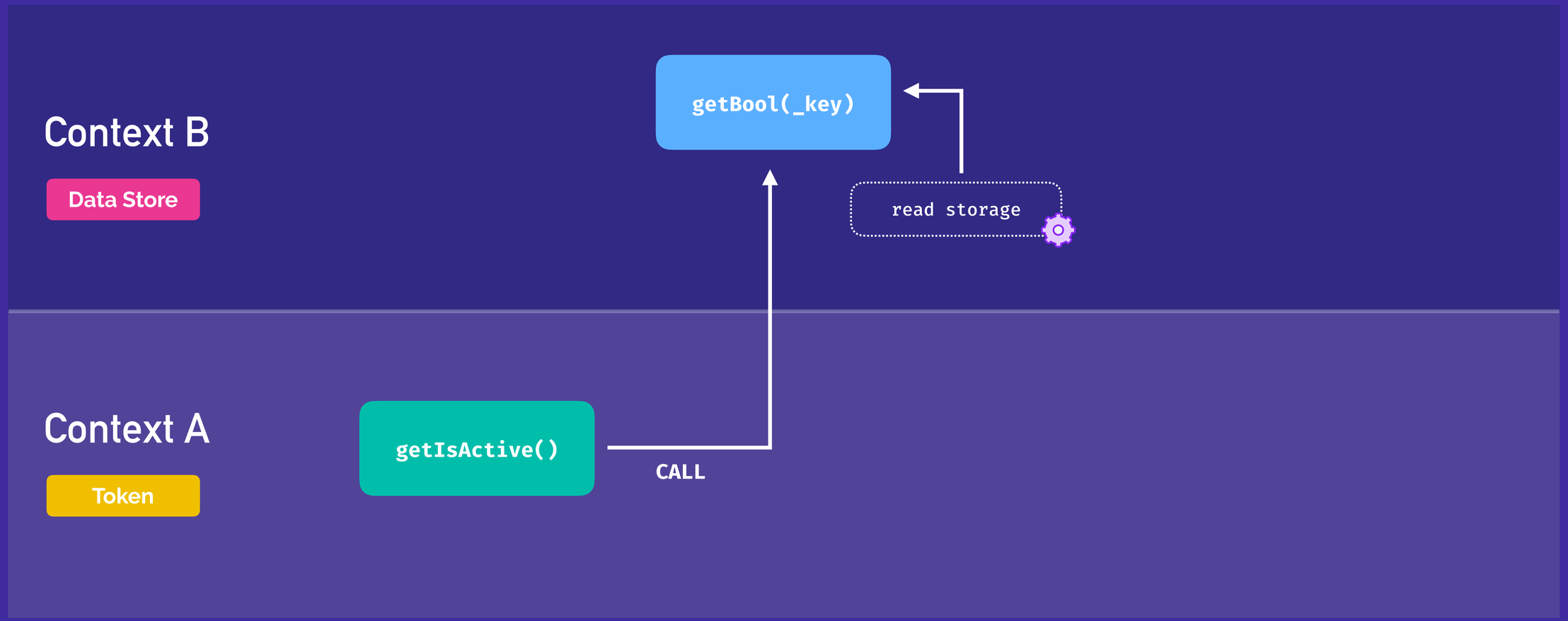
CALL


Context B
Data Store
getBool(_key)
Context A
Token
getIsActive()
CALL

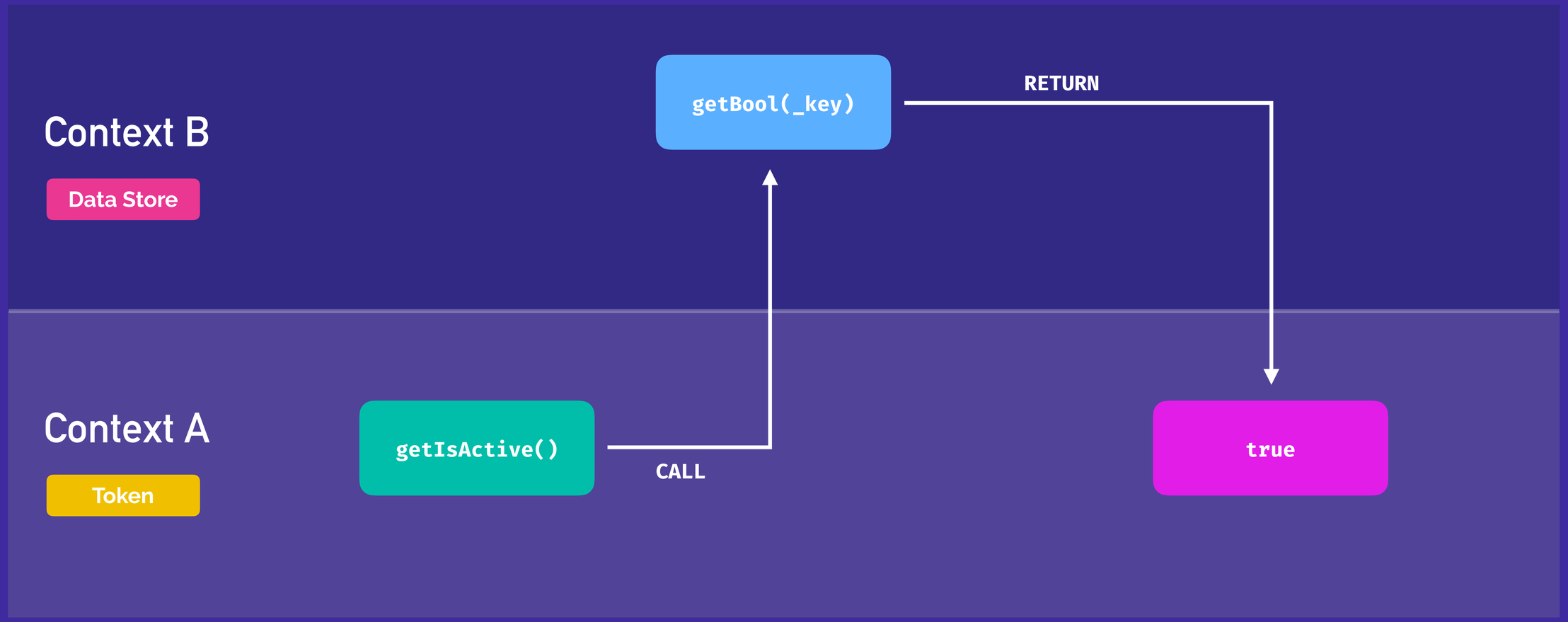
CALL


Context B
Data Store
getBool(_key)
read storage
Context A
Token
getIsActive()
CALL

CALL


Context B
Data Store
getBool(_key)
read storage
Context A
Token
getIsActive()
CALL

CALL


Context B
Data Store
getBool(_key)
RETURN
Context A
Token
getIsActive()
CALL
true

CALL

Context B

Data Store

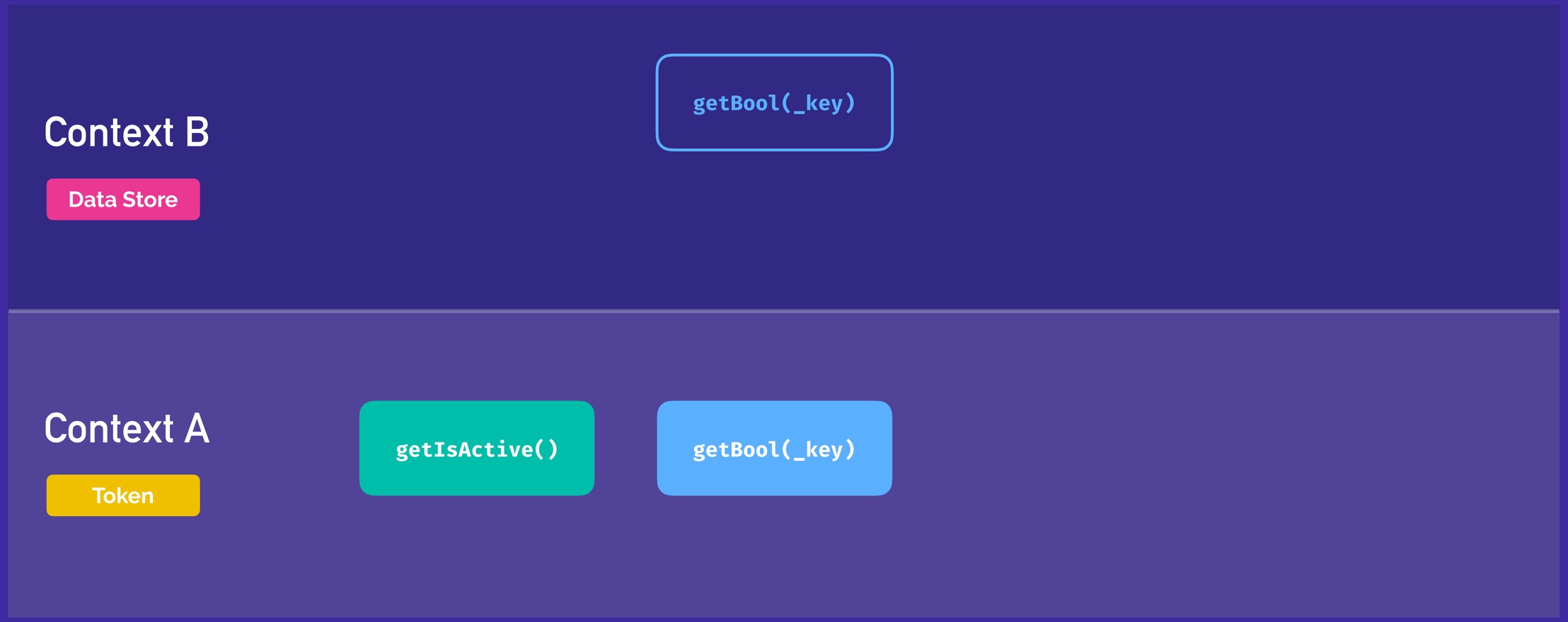
`getBool(_key)`

Context A

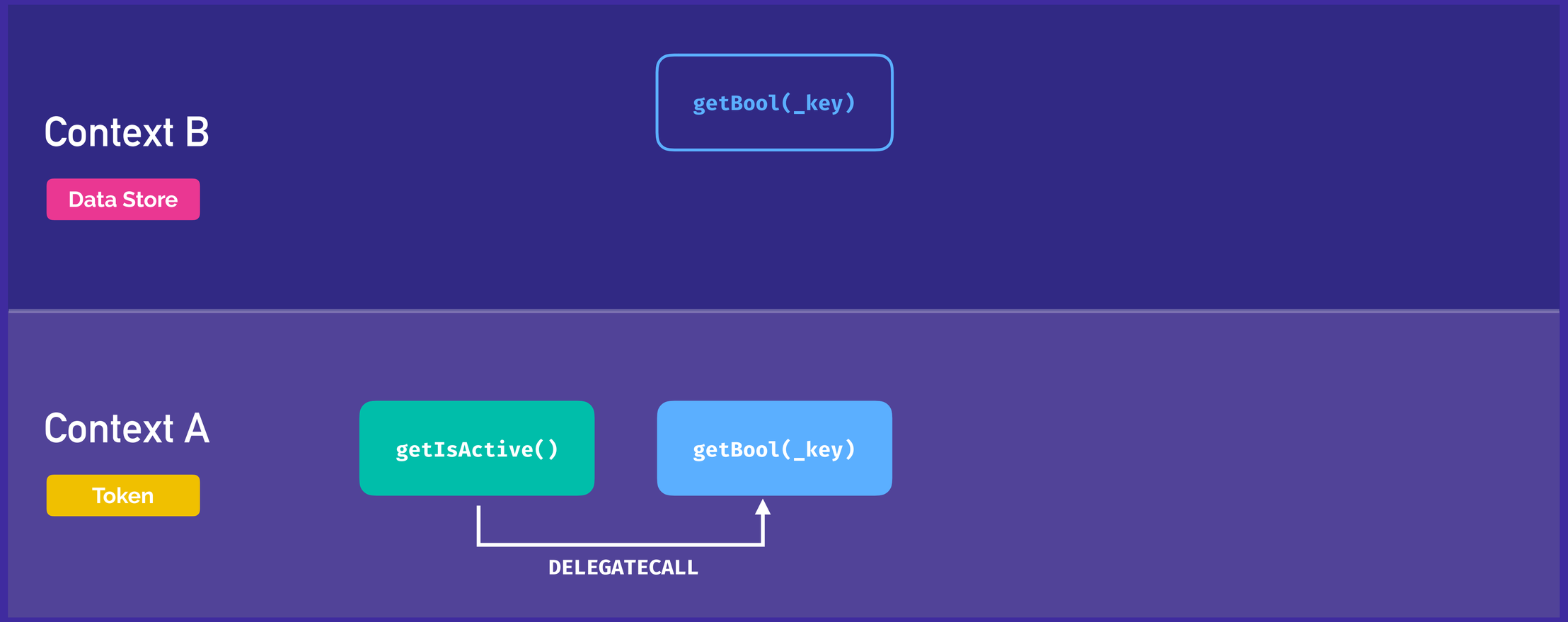
Token

`getIsActive()`

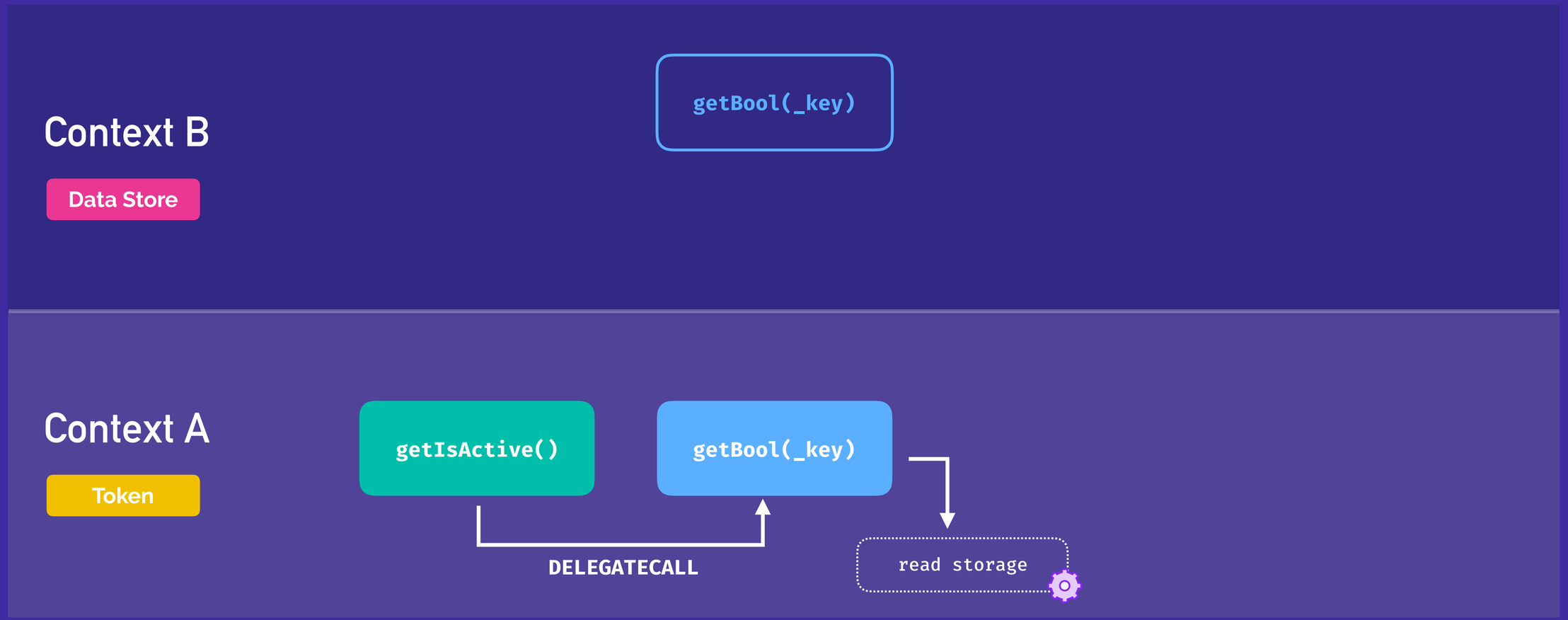
DELEGATECALL



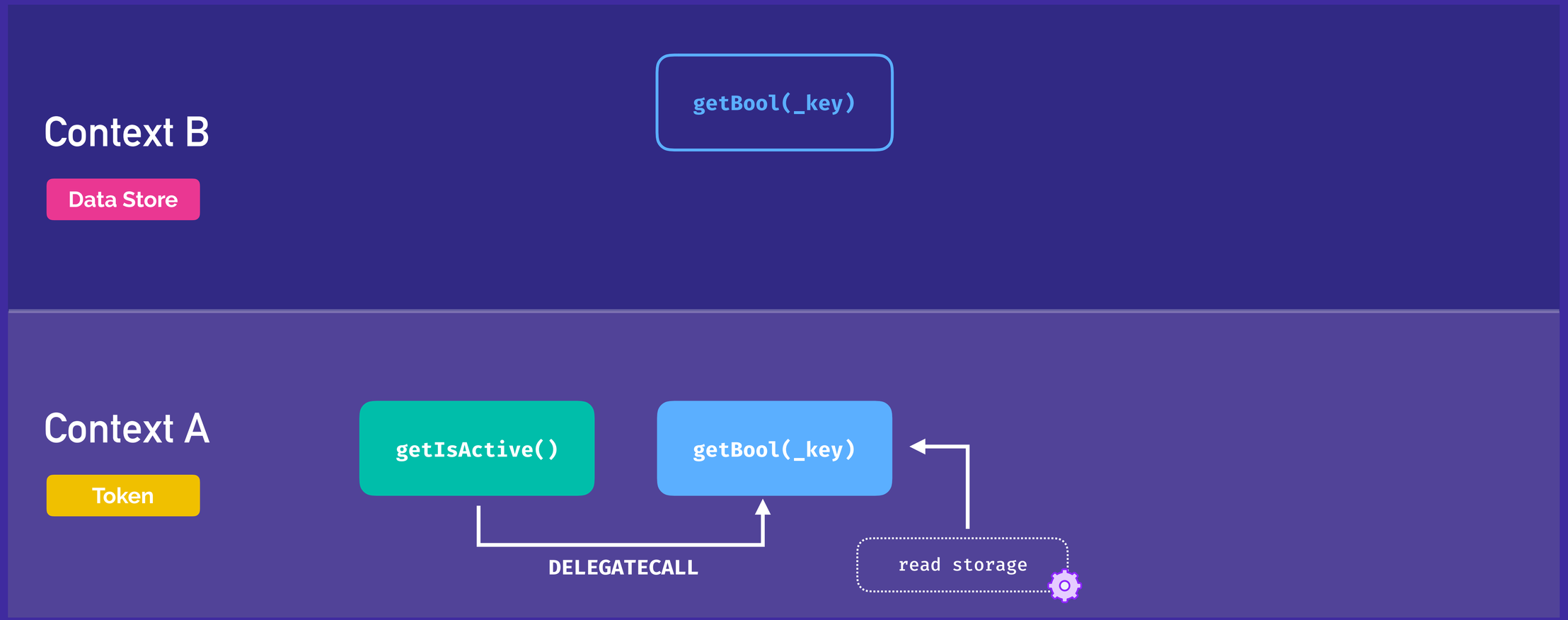
DELEGATECALL



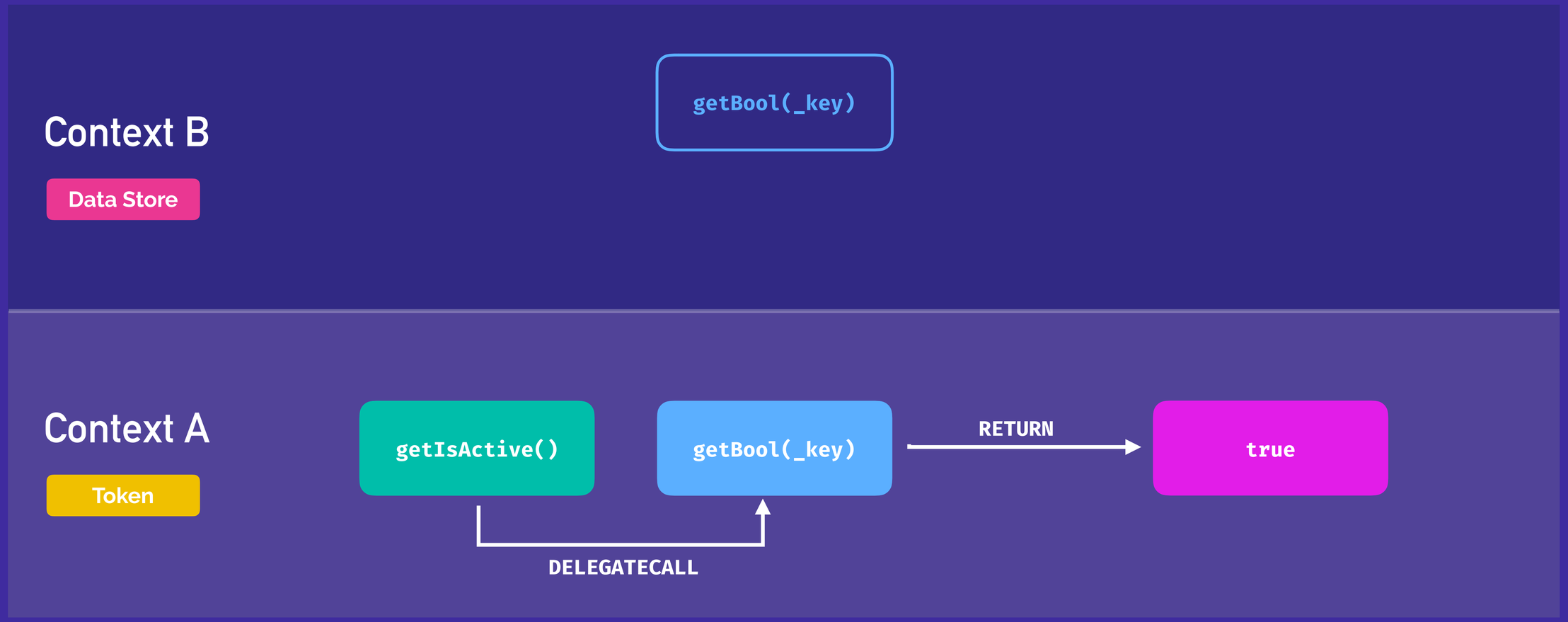
DELEGATECALL



DELEGATECALL



DELEGATECALL



DELEGATECALL

CALL

DELEGATECALL

CALL

- Happens in the context of the executing contract

DELEGATECALL

CALL

- Happens in the context of the executing contract

DELEGATECALL

- Happens in the context of the calling contract

CALL

- Happens in the context of the executing contract
- Uses its own storage

DELEGATECALL

- Happens in the context of the calling contract

CALL

- Happens in the context of the executing contract
- Uses its own storage

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract

CALL

- Happens in the context of the executing contract
- Uses its own storage
- Contract address:
  Executing contract's address

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract

CALL

- Happens in the context of the executing contract
- Uses its own storage
- Contract address:
  Executing contract's address

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract
- Contract address:
  Is NOT changed

CALL

- Happens in the context of the executing contract
- Uses its own storage
- Contract address:
  Executing contract's address
- Contract balance:
  Executing contract's balance

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract
- Contract address:
  Is NOT changed

CALL

- Happens in the context of the executing contract
- Uses its own storage
- Contract address:
  Executing contract's address
- Contract balance:
  Executing contract's balance

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract
- Contract address:
  Is NOT changed
- Contract balance:
  Is NOT changed

CALL

- Happens in the context of the executing contract
- Uses its own storage
- Contract address:
  Executing contract's address
- Contract balance:
  Executing contract's balance
- Msg.sender:
  Contract that issued the call

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract
- Contract address:
  Is NOT changed
- Contract balance:
  Is NOT changed

CALL

- Happens in the context of the executing contract
- Uses its own storage
- Contract address:
  Executing contract's address
- Contract balance:
  Executing contract's balance
- Msg.sender:
  Contract that issued the call

DELEGATECALL

- Happens in the context of the calling contract
- Uses the storage of the calling contract
- Contract address:
  Is NOT changed
- Contract balance:
  Is NOT changed
- Msg.sender:
  Is NOT changed

ISSUING A DELEGATE CALL

```
contract Proxy {









}
```

```
contract Proxy {

address delegateContract = 0×685…

}
```

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {









}

}
```

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {
  assembly {









  }
}

}
```

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {
  assembly {
    let ptr := mload(0×40)
    calldatacopy(ptr, 0, calldatasize)
    let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
    let size := returndatasize
    returndatacopy(ptr, 0, size)

    switch result {
      case 0 { revert(ptr, size) }
      default { return(ptr, size) }
    }
  }
}

}
```

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {
  assembly {
    let ptr := mload(0×40)
    calldatacopy(ptr, 0, calldatasize)
    let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
    let size := returndatasize
    returndatacopy(ptr, 0, size)

    switch result {
      case 0 { revert(ptr, size) }
      default { return(ptr, size) }
    }
  }
}

}
```

Read the address of the next available memory slot

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {
  assembly {
    let ptr := mload(0×40)
    calldatacopy(ptr, 0, calldatasize)
    let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
    let size := returndatasize
    returndatacopy(ptr, 0, size)

    switch result {
      case 0 { revert(ptr, size) }
      default { return(ptr, size) }
    }
  }
}

}
```

Read the address of the next available memory slot

Copy the call data to memory

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {
  assembly {
    let ptr := mload(0×40)
    calldatacopy(ptr, 0, calldatasize)
    let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
    let size := returndatasize
    returndatacopy(ptr, 0, size)

    switch result {
      case 0 { revert(ptr, size) }
      default { return(ptr, size) }
    }
  }
}

}
```

Read the address of the next available memory slot

Copy the call data to memory

Issue the DELEGATECALL to the delegate contract, passing all available gas

For call data, use the value we stored in memory

Store the result of the DELEGATECALL operation in the “result” variable

```
contract Proxy {

address delegateContract = 0×685…

function () payable public {
  assembly {
    let ptr := mload(0×40)
    calldatacopy(ptr, 0, calldatasize)
    let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
    let size := returndatasize
    returndatacopy(ptr, 0, size)

    switch result {
      case 0 { revert(ptr, size) }
      default { return(ptr, size) }
    }
  }
}

}
```

Read the address of the next available memory slot

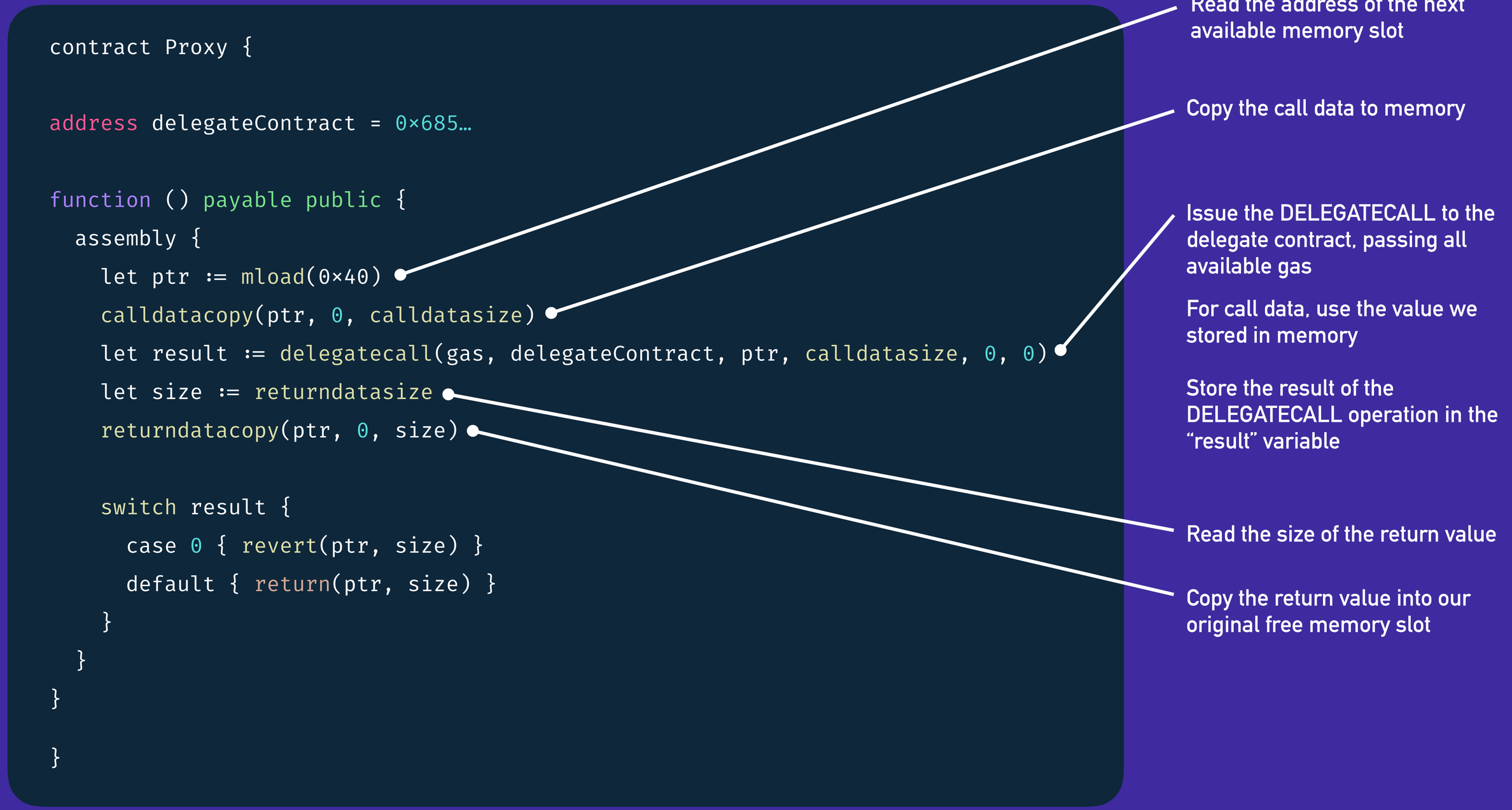
Copy the call data to memory

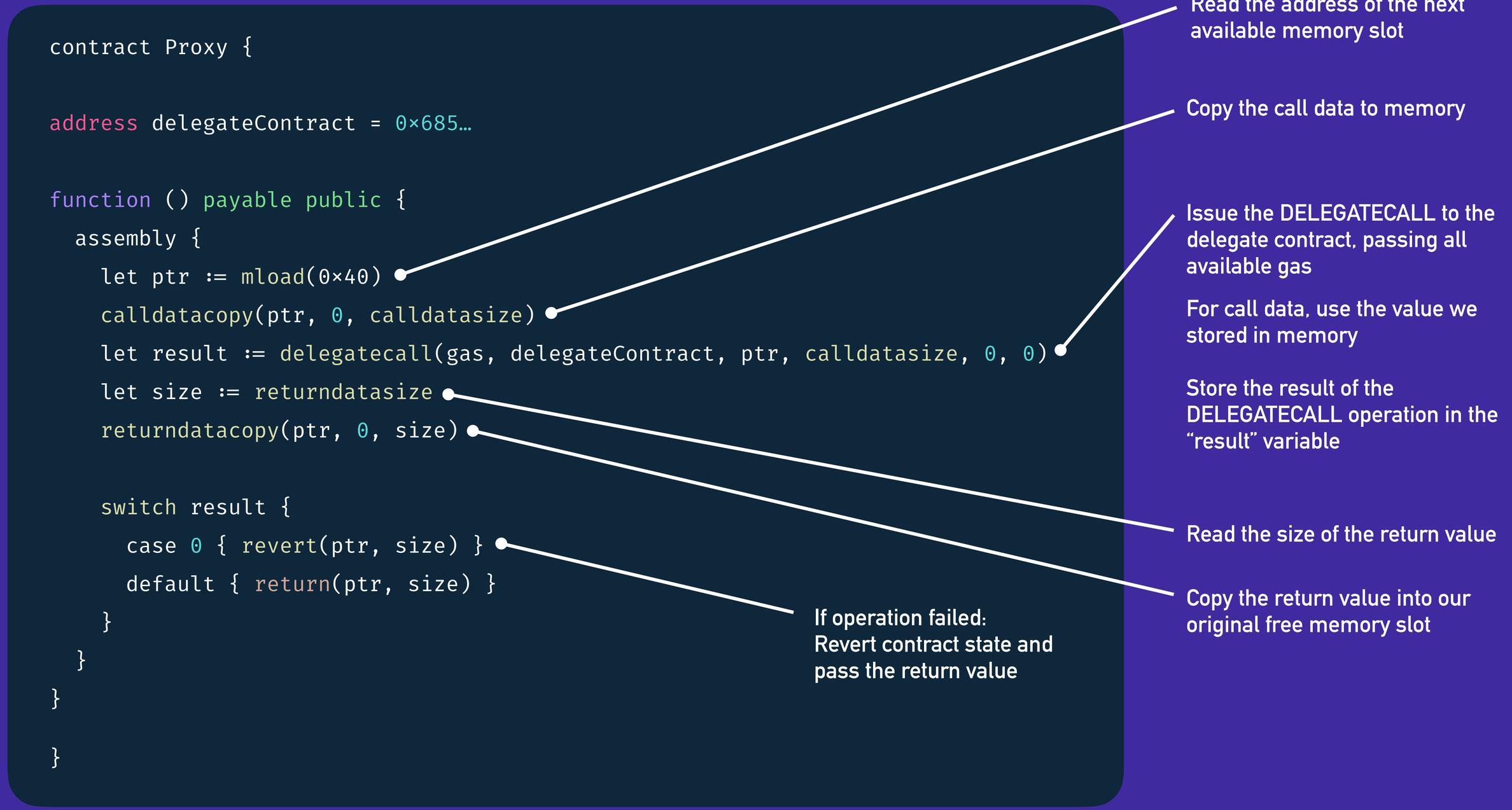
Issue the DELEGATECALL to the delegate contract, passing all available gas

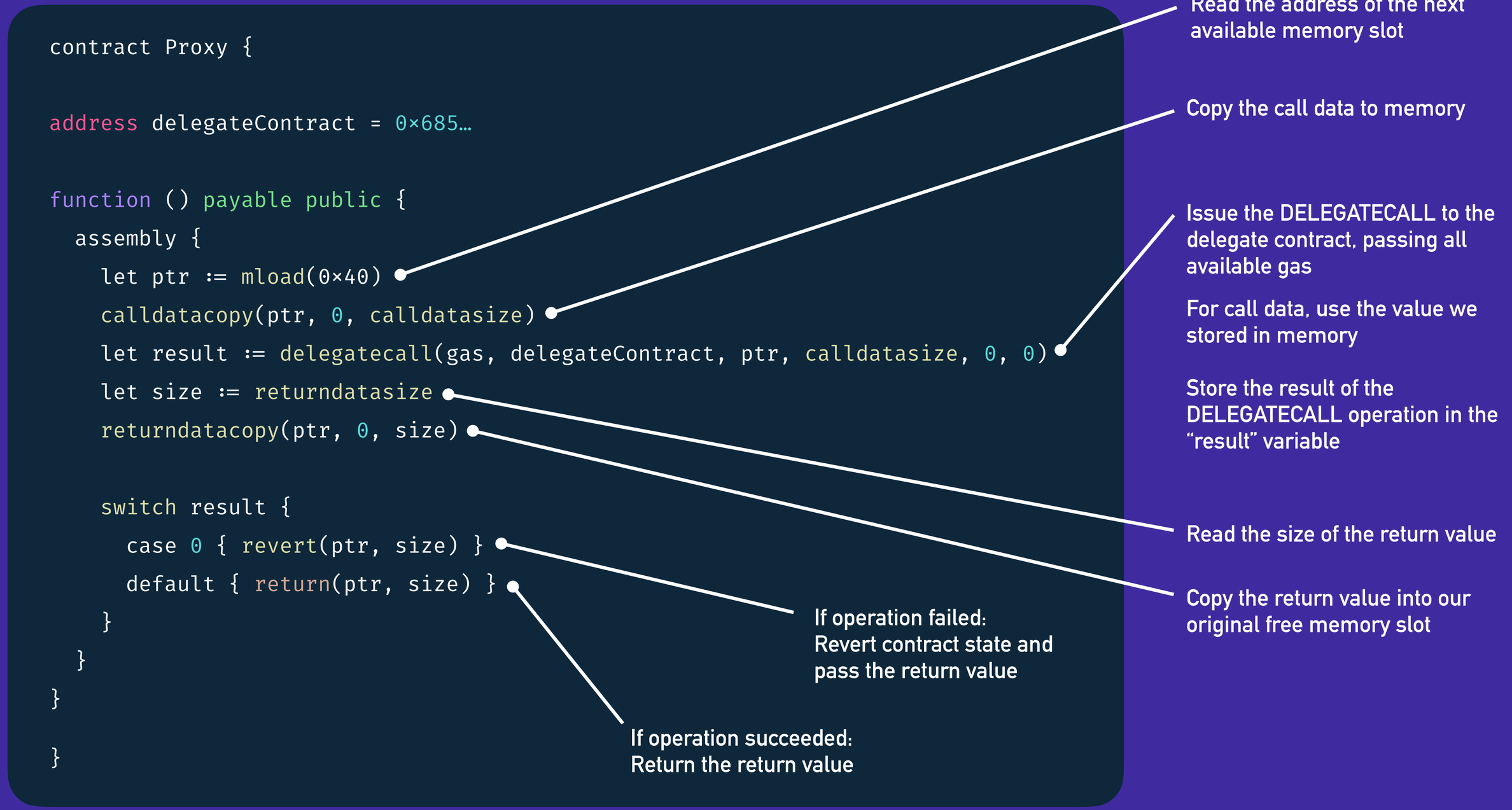
For call data, use the value we stored in memory

Store the result of the DELEGATECALL operation in the “result” variable

Read the size of the return value


contract Proxy {
address delegateContract = 0×685…
function () payable public {
assembly {
let ptr := mload(0×40)
calldatacopy(ptr, 0, calldatasize)
let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
let size := returndatasize
returndatacopy(ptr, 0, size)
switch result {
case 0 { revert(ptr, size) }
default { return(ptr, size) }
}
}
}
}
Read the address of the next available memory slot
Copy the call data to memory
Issue the DELEGATECALL to the delegate contract, passing all available gas
For call data, use the value we stored in memory
Store the result of the DELEGATECALL operation in the “result” variable
Read the size of the return value
Copy the return value into our original free memory slot


contract Proxy {
address delegateContract = 0×685…
function () payable public {
assembly {
let ptr := mload(0×40)
calldatacopy(ptr, 0, calldatasize)
let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
let size := returndatasize
returndatacopy(ptr, 0, size)
switch result {
case 0 { revert(ptr, size) }
default { return(ptr, size) }
}
}
}
}
Read the address of the next available memory slot
Copy the call data to memory
Issue the DELEGATECALL to the delegate contract, passing all available gas
For call data, use the value we stored in memory
Store the result of the DELEGATECALL operation in the “result” variable
Read the size of the return value
Copy the return value into our original free memory slot
If operation failed:
Revert contract state and pass the return value


contract Proxy {
address delegateContract = 0×685…
function () payable public {
assembly {
let ptr := mload(0×40)
calldatacopy(ptr, 0, calldatasize)
let result := delegatecall(gas, delegateContract, ptr, calldatasize, 0, 0)
let size := returndatasize
returndatacopy(ptr, 0, size)
switch result {
case 0 { revert(ptr, size) }
default { return(ptr, size) }
}
}
}
}
Read the address of the next available memory slot
Copy the call data to memory
Issue the DELEGATECALL to the delegate contract, passing all available gas
For call data, use the value we stored in memory
Store the result of the DELEGATECALL operation in the “result” variable
Read the size of the return value
Copy the return value into our original free memory slot
If operation failed:
Revert contract state and pass the return value
If operation succeeded:
Return the return value


ForGIFs.com

EVM STORAGE & DELEGATECALL

PROXY PATTERN

PROXY PATTERN

PROXY PATTERN

Proxy

PROXY PATTERN

- Holds an address to the logic contract

PROXY PATTERN

Proxy

- Holds an address to the logic contract
- Uses the fallback function as a catch-all

PROXY PATTERN

Proxy

- Holds an address to the logic contract
- Uses the fallback function as a catch-all
- Delegates calls to a logic contract

PROXY PATTERN

- Holds an address to the logic contract
- Uses the fallback function as a catch-all
- Delegates calls to a logic contract
- Returns the value produced by the logic contract

PROXY PATTERN

PROXY PATTERN

Logic

PROXY PATTERN

- Executes its logic within the context of the proxy contract

PROXY PATTERN

- Executes its logic within the context of the proxy contract
- Returns execution result to the proxy

LET'S DEPLOY

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

DELEGATECALL

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

DELEGATECALL

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

return 1

Proxy

0×580 …

Poll

0×732 …


contract Proxy {
address implementation;
function setAddress(address _implementation) {
implementation = _implementation;
}
function () payable public {
…
delegatecall(
gas,
implementation,
ptr,
calldatasize,
0,
0)
…
}
}
contract Poll {
uint cokeVotes = 0;
uint pepsiVotes = 0;
function voteForCoke() returns(uint) {
cokeVotes++;
return cokeVotes;
}
function voteForPepsi() returns(uint) {
pepsiVotes++;
return pepsiVotes;
}
}
voteForCoke()
DELEGATECALL
return 1
return 1
Proxy
0×580 …
Poll
0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForPepsi()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForPepsi()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

DELEGATECALL

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …


Cyber Attack

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

DELEGATECALL

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

DELEGATECALL

Proxy

0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

voteForCoke()

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

DELEGATECALL

Proxy

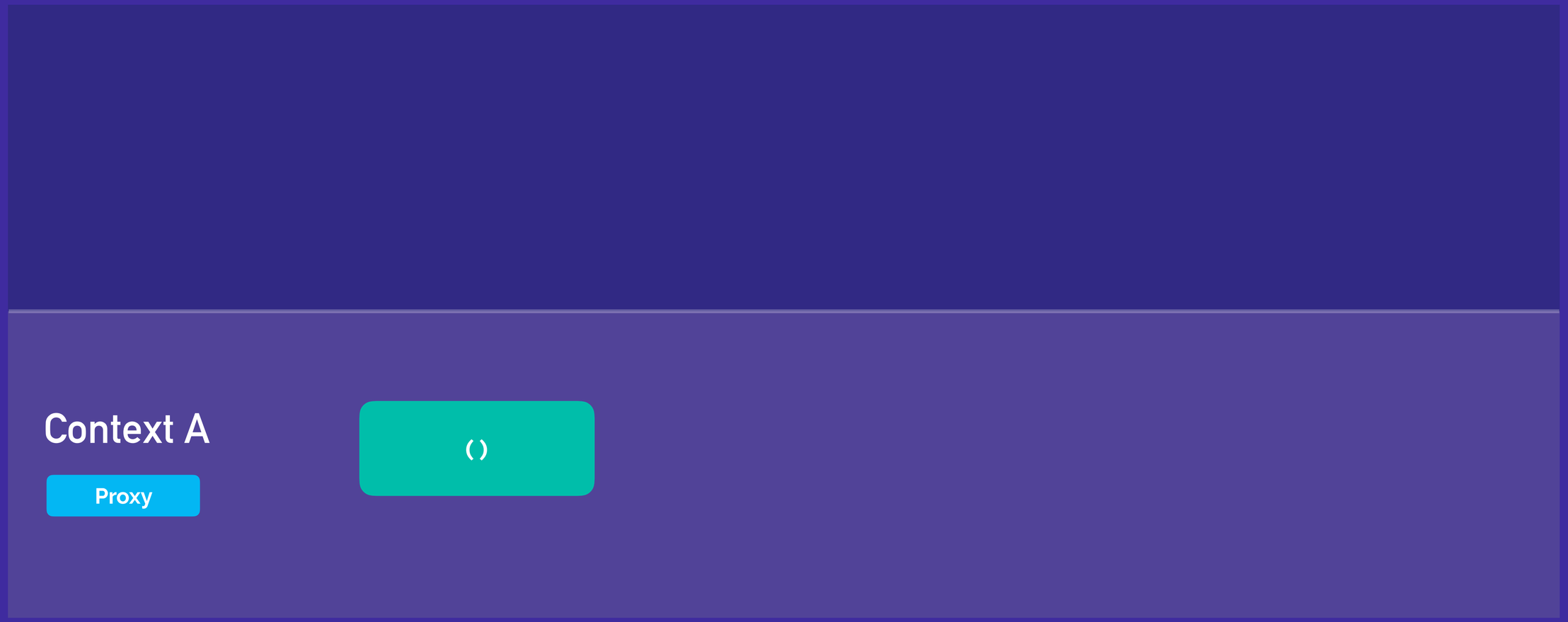
0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

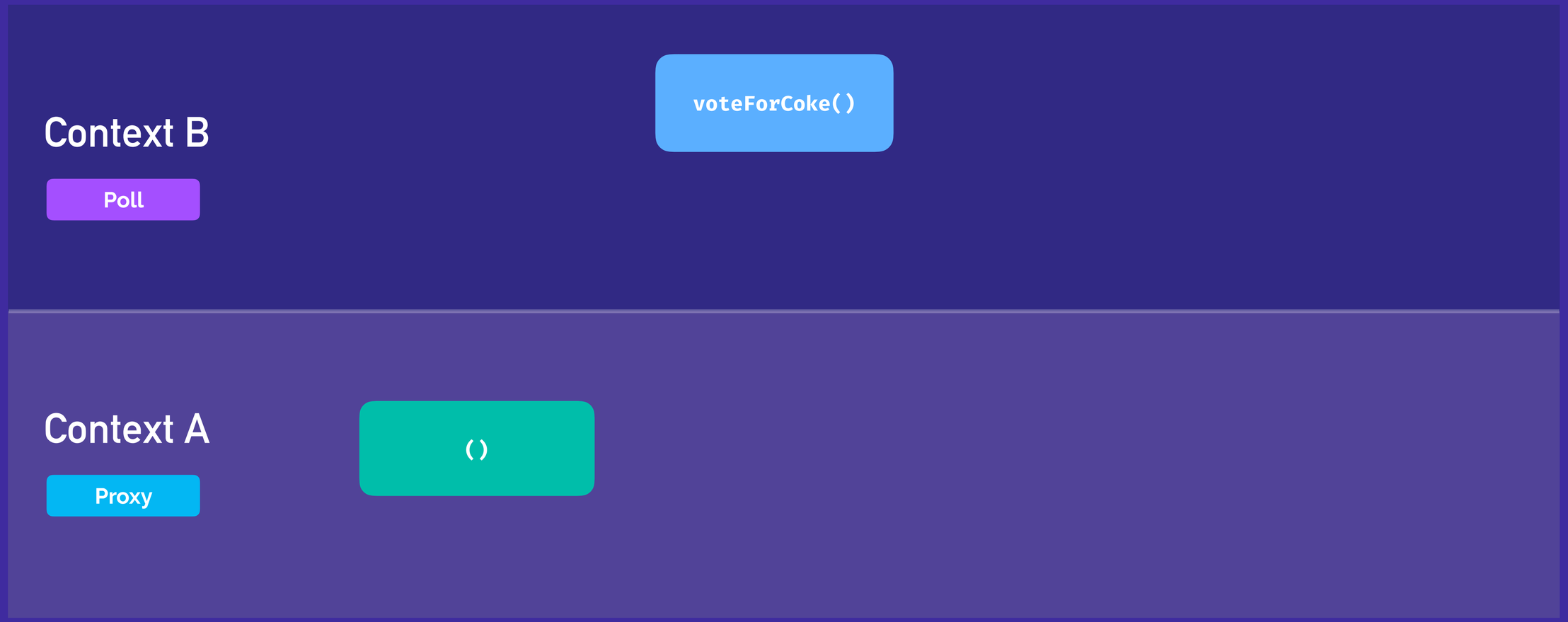
Poll

0×732 …

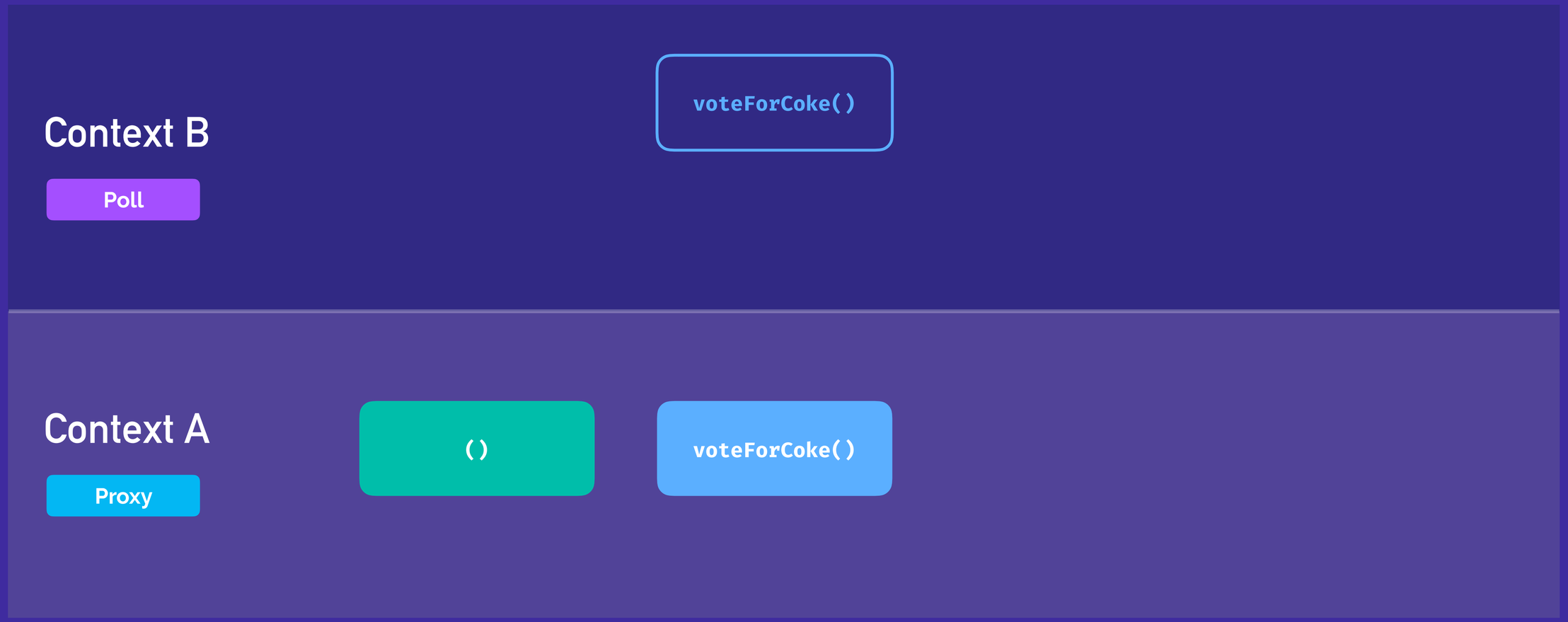
DELEGATECALL



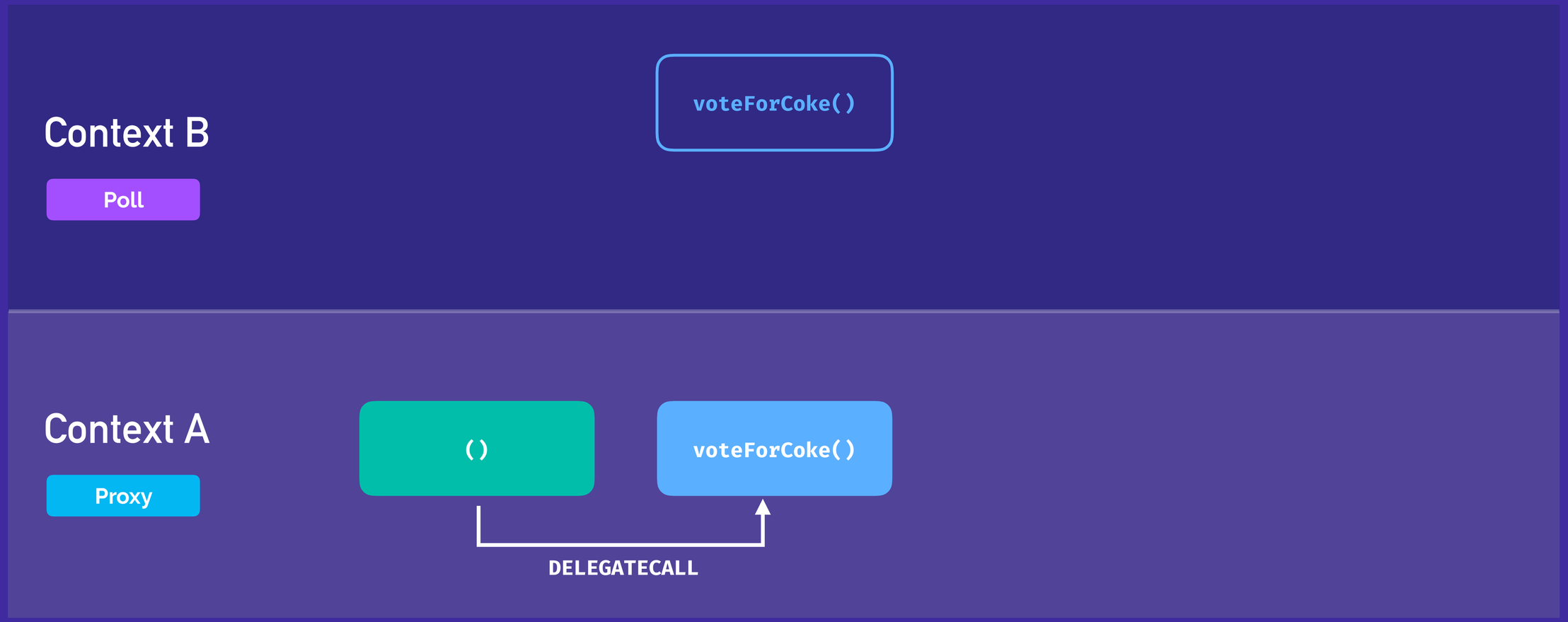
DELEGATECALL



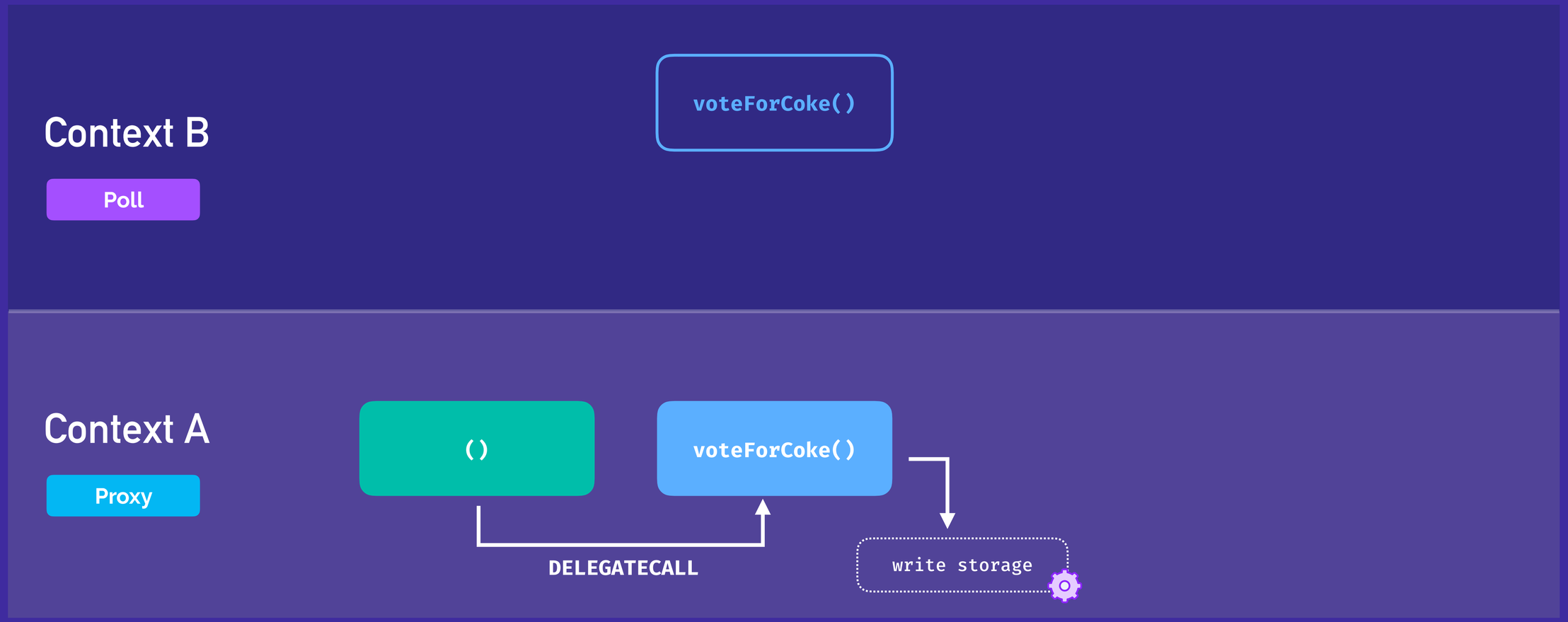
DELEGATECALL



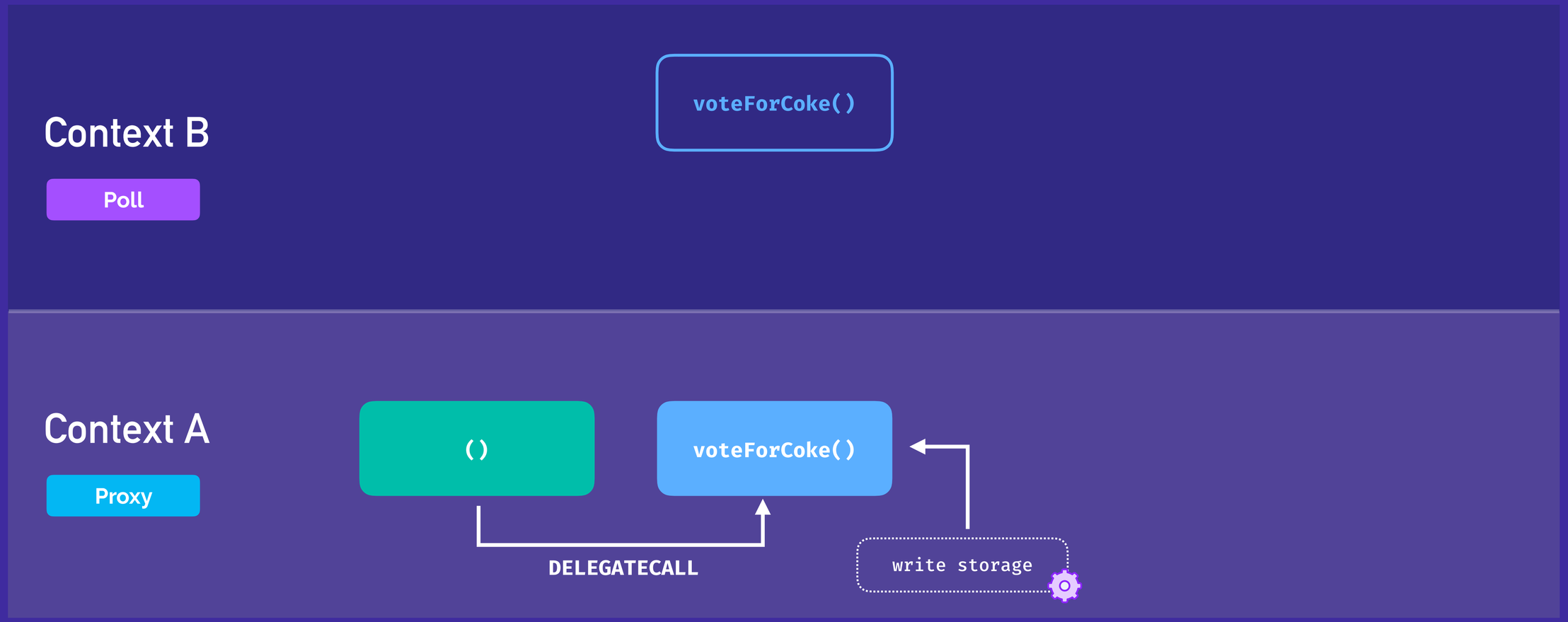
DELEGATECALL



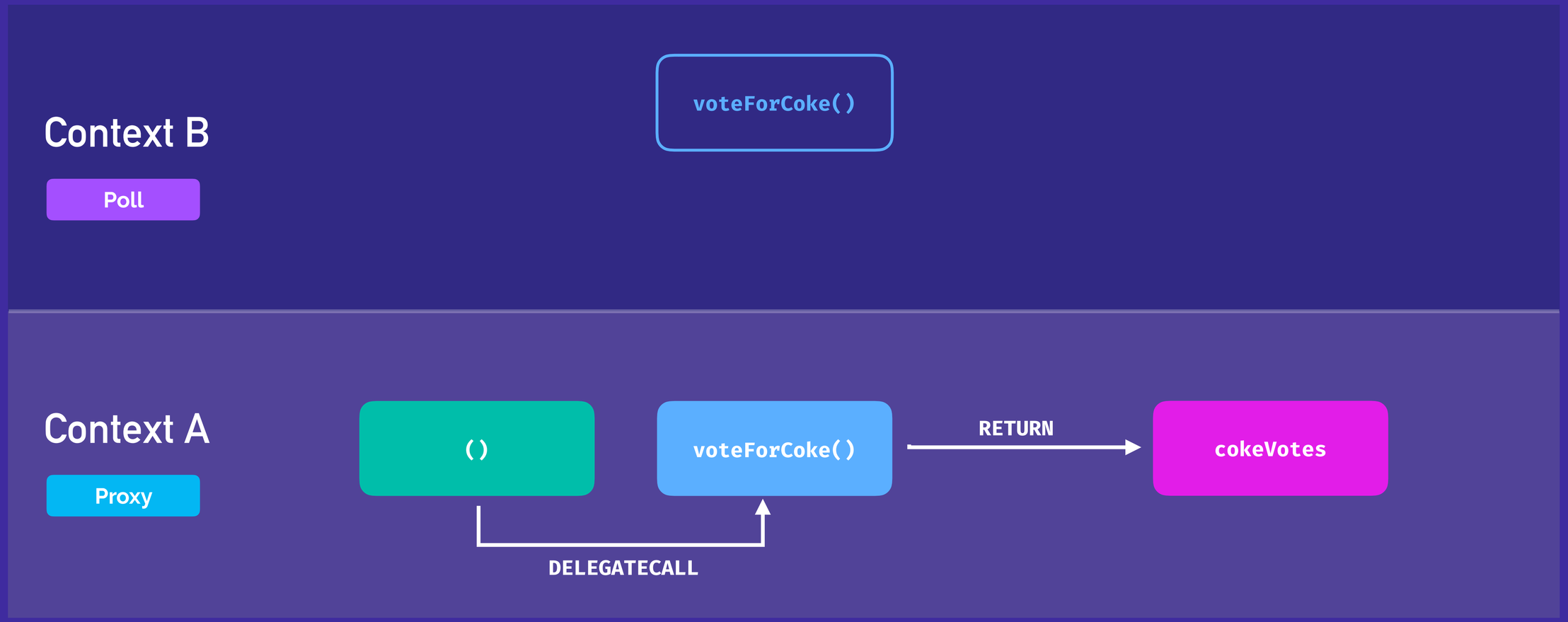
DELEGATECALL



DELEGATECALL



DELEGATECALL



DELEGATECALL

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

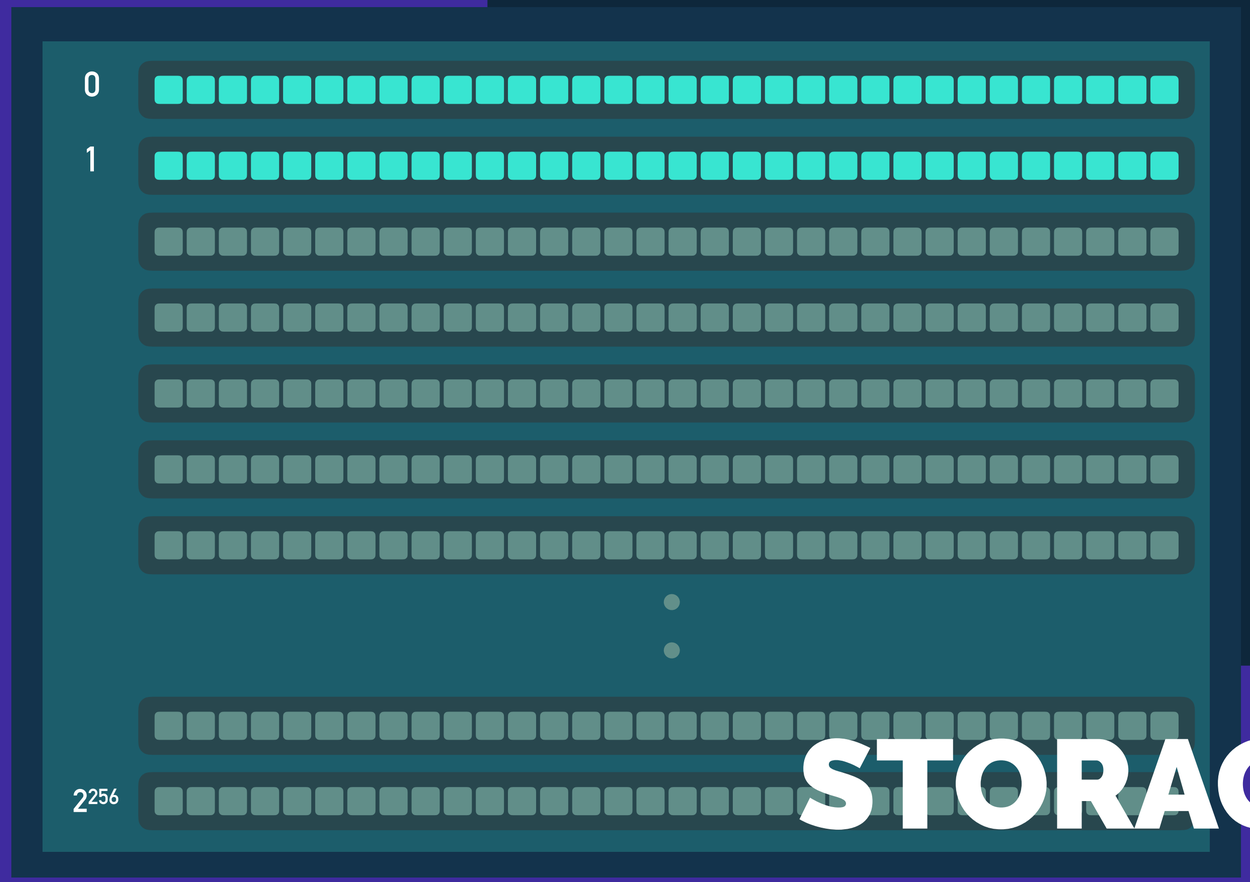
0×580 …

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}
```

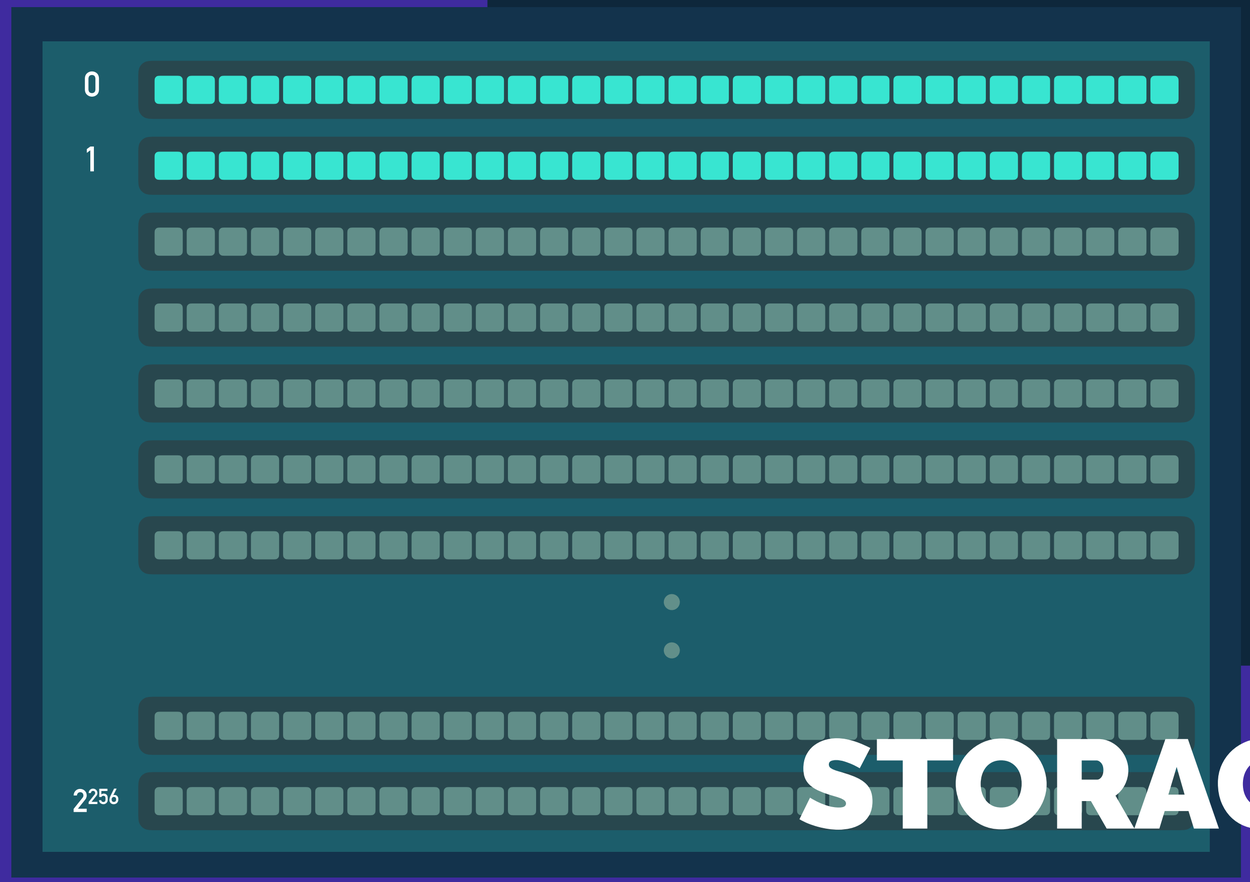



```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}
```




```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

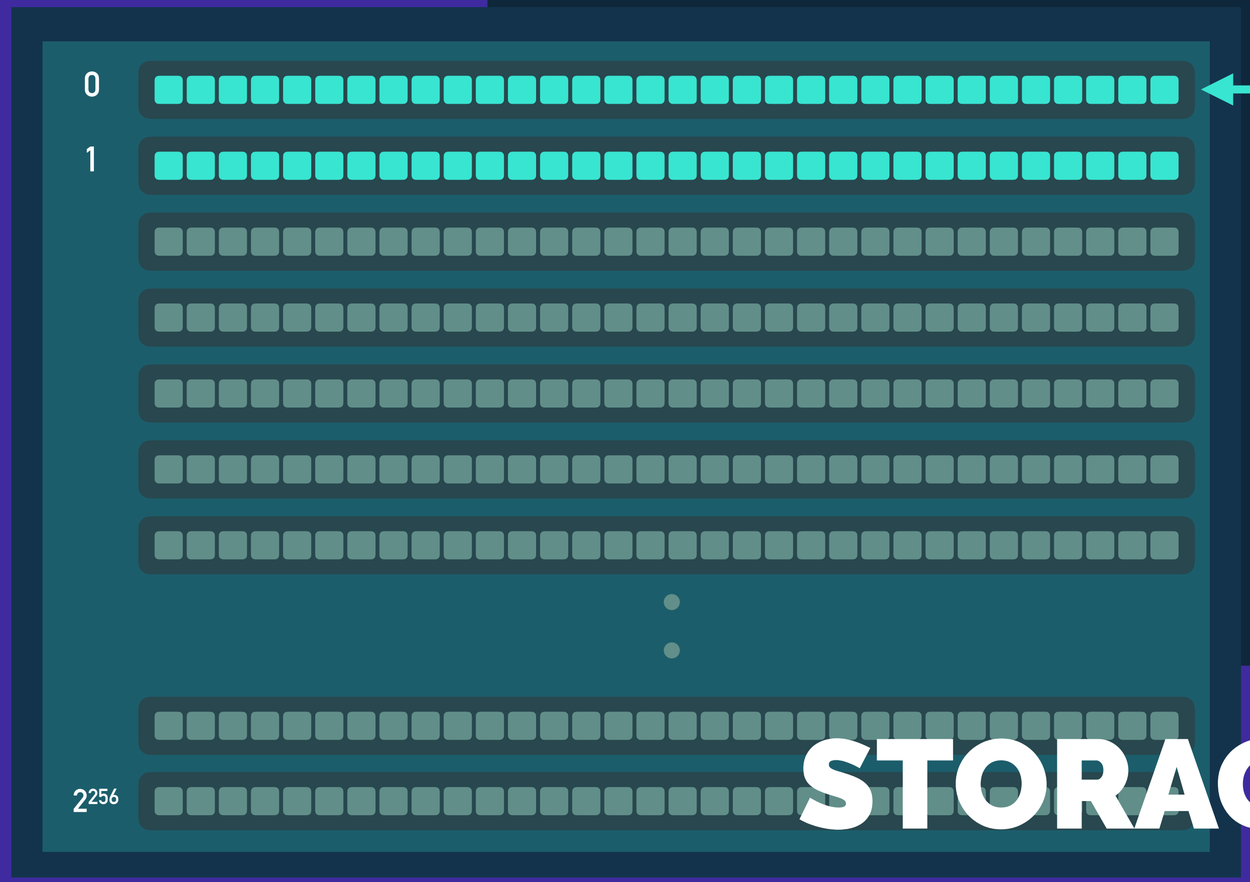
Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}
```

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

0



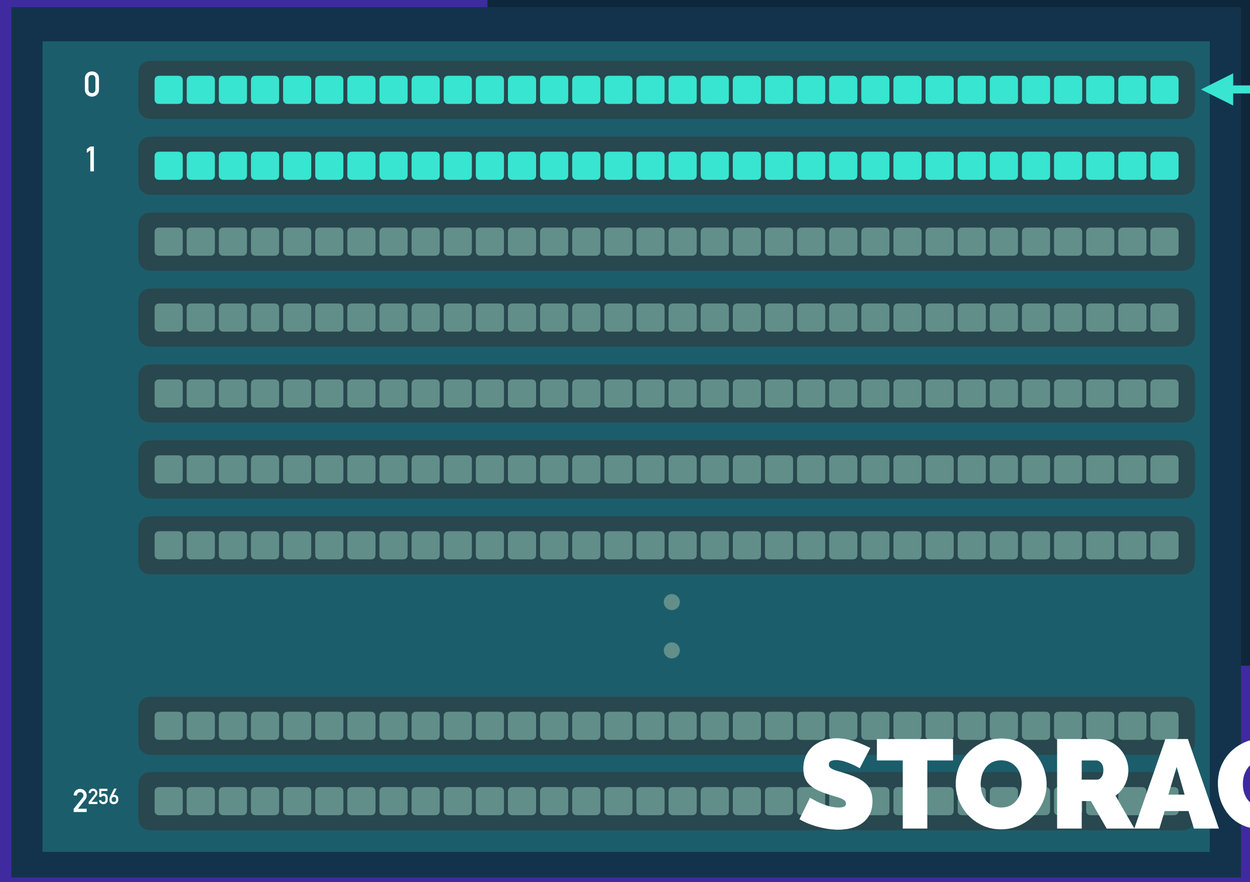
Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}
```

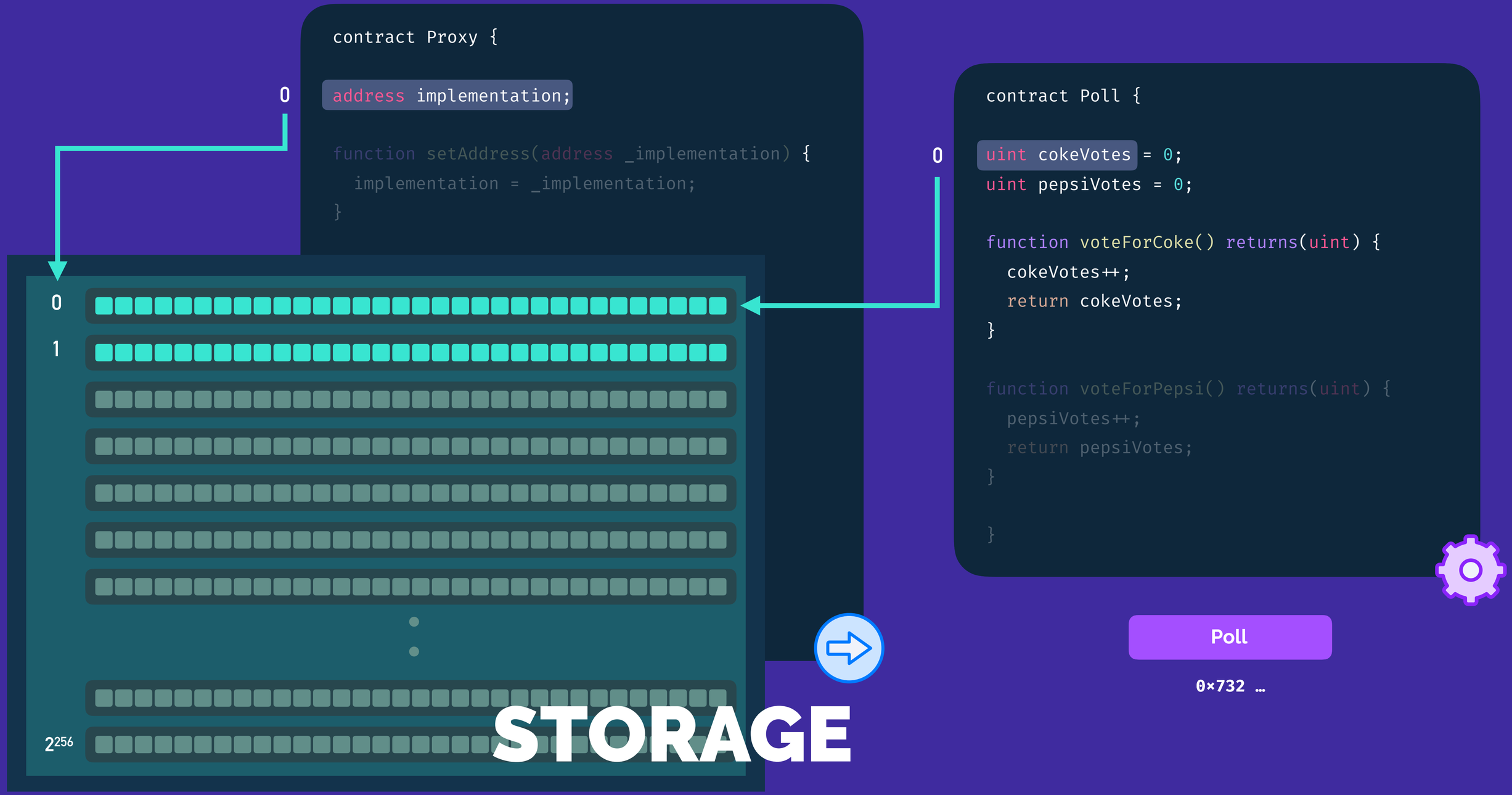
```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

0



Poll

0×732 …


contract Proxy {
0
address implementation;
function setAddress(address _implementation) {
implementation = _implementation;
}
0
0
1
2256
STORAGE
contract Poll {
uint cokeVotes = 0;
uint pepsiVotes = 0;
function voteForCoke() returns(uint) {
cokeVotes++;
return cokeVotes;
}
function voteForPepsi() returns(uint) {
pepsiVotes++;
return pepsiVotes;
}
}
Poll
0×732 …

DATA CORRUPTION CAN HAPPEN

PROXY

STORAGE PATTERNS

PROXY PATTERN:

INHERITED STORAGE

INHERITED STORAGE

Proxy Storage

INHERITED STORAGE

Proxy Storage

Proxy

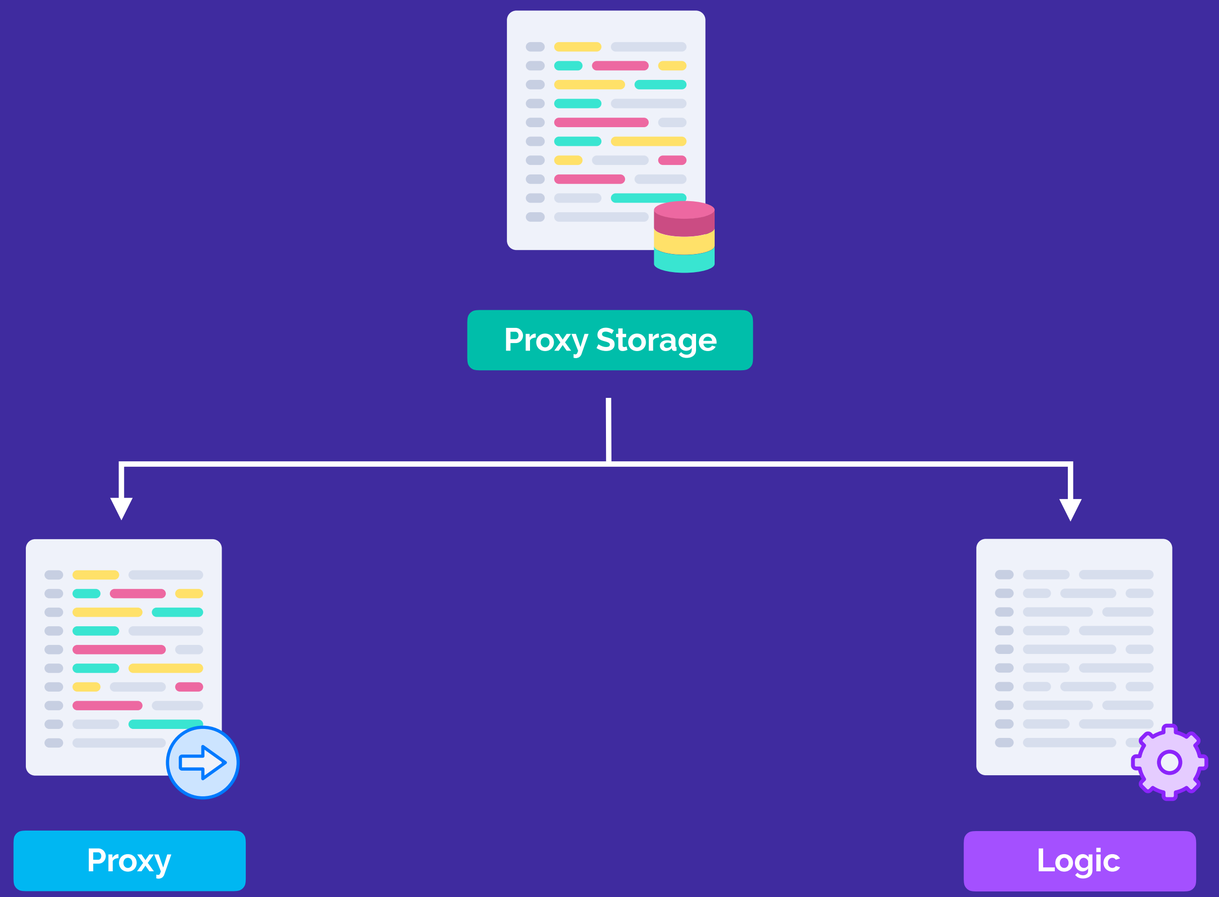
INHERITED STORAGE

Proxy Storage

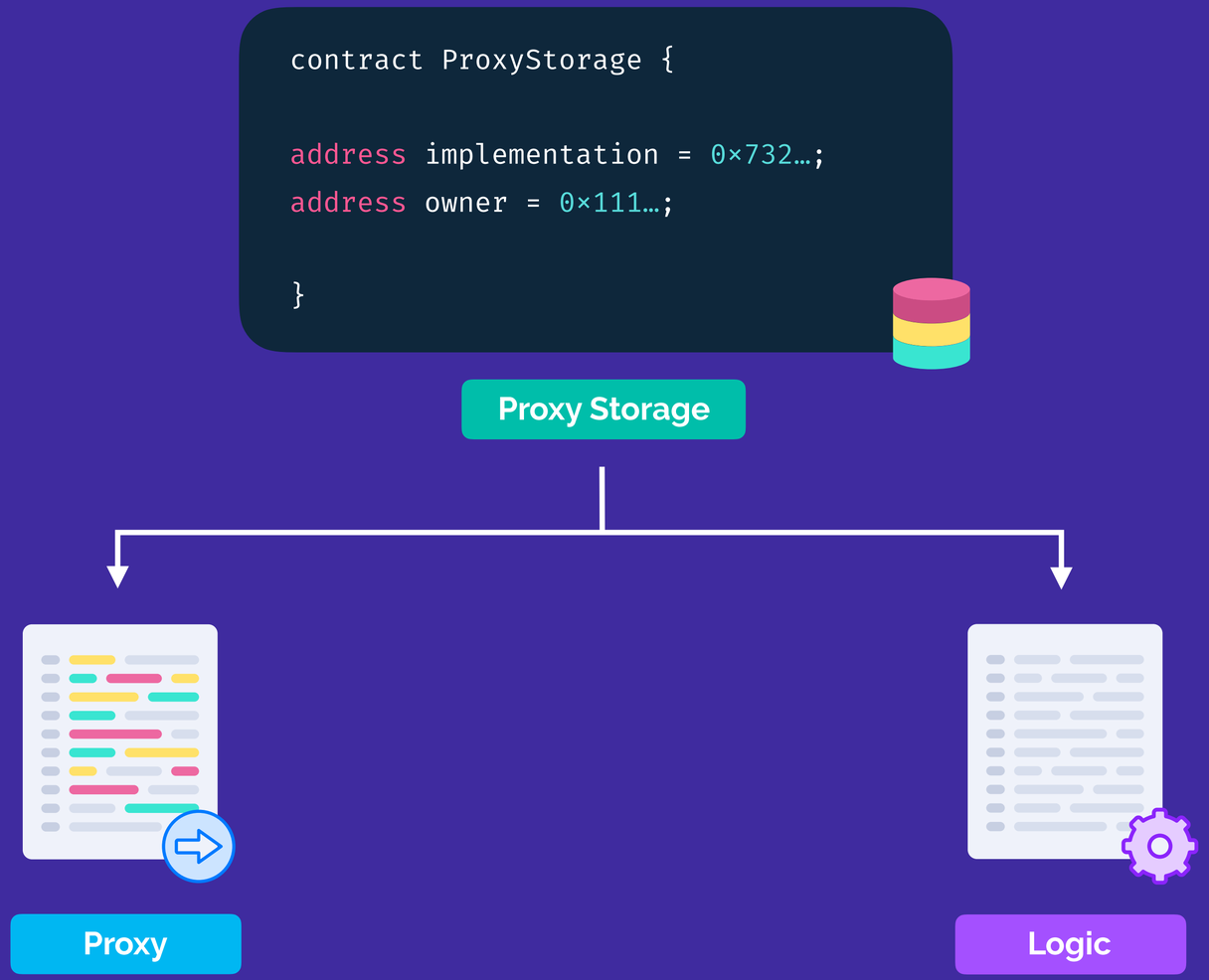
Proxy

Logic

INHERITED STORAGE



INHERITED STORAGE


contract ProxyStorage {
address implementation = 0×732…;
address owner = 0×111…;
}
Proxy Storage
Proxy
Logic

```
contract ProxyStorage {

address implementation = 0×732…;
address owner = 0×111…;

}
```

Proxy Storage

```
contract Proxy is ProxyStorage {

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

Logic

```
contract ProxyStorage {

address implementation = 0×732…;
address owner = 0×111…;

}
```

Proxy Storage

```
contract Proxy is ProxyStorage {

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is ProxyStorage {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
 …
}
function voteForPepsi() returns(uint) {
 …
}
}
```

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}

}
```

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

0

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Proxy

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

0	address implementation;
1	address owner;

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Proxy

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}

}
```

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

0	address implementation;
1	address owner;

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Proxy

Logic

Proxy Storage

```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

0	address implementation;
1	address owner;
2	uint cokeVotes = 0;
	uint pepsiVotes = 0;

```
contract Poll is ProxyStorage {

function voteForCoke() returns(uint) {
…
}

function voteForPepsi() returns(uint) {
…
}

}
```

Proxy

Logic

Proxy Storage

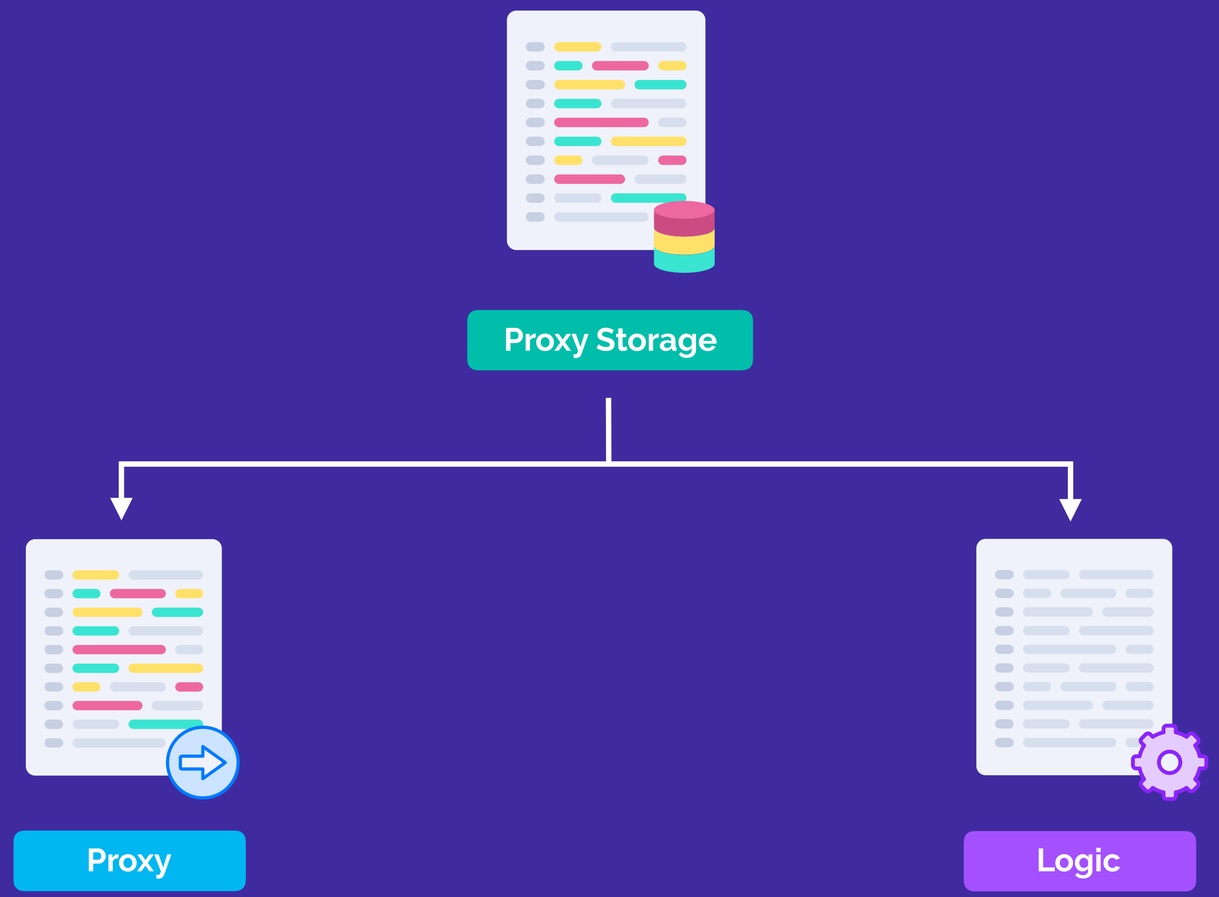
```
contract Proxy is ProxyStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

0	`address implementation;`
1	`address owner;`
2	`uint cokeVotes = 0;`
3	`uint pepsiVotes = 0;`

```
contract Poll is ProxyStorage {

address implementation;
address owner;

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}

}
```

Proxy

Logic



INHERITED STORAGE

👍 No overriding of storage!

INHERITED STORAGE

👍 No overriding of storage!

👍 Easy to implement

INHERITED STORAGE

👍 No overriding of storage!

👍 Easy to implement

👍 Logic contract can introduce new functions and variables over time

INHERITED STORAGE

👎 Proxy contract code does not reflect the state that it stores

INHERITED STORAGE

👎 Proxy contract code does not reflect the state that it stores

👎 The logic contract inherits properties that are only related to the functionality of the proxy

INHERITED STORAGE

👎 Proxy contract code does not reflect the state that it stores

👎 The logic contract inherits properties that are only related to the functionality of the proxy

👎 The logic ends up knowing about the proxy

INHERITED STORAGE

PROXY PATTERN:

ETERNAL STORAGE

ETERNAL STORAGE

Eternal Storage

ETERNAL STORAGE

Eternal Storage

Proxy

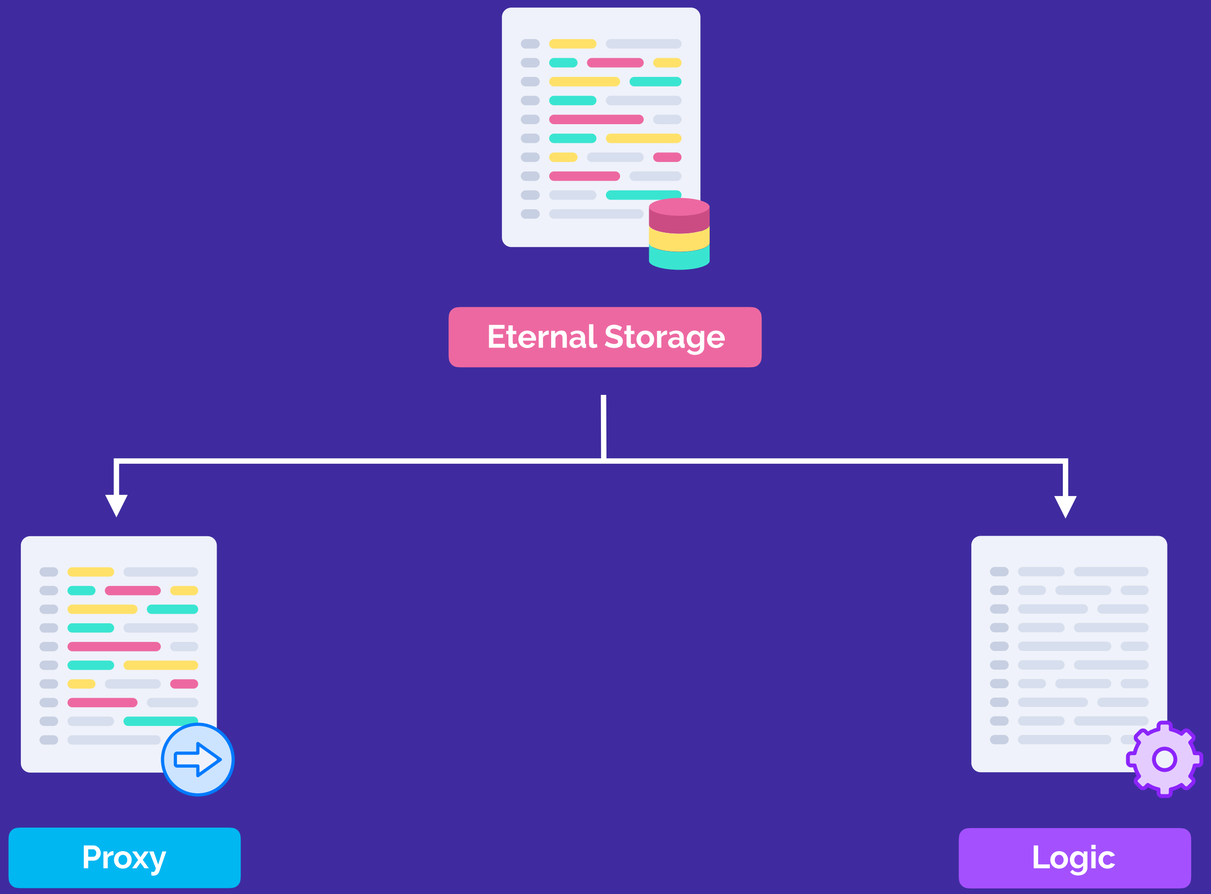
ETERNAL STORAGE

Eternal Storage

Proxy

Logic

ETERNAL STORAGE



ETERNAL STORAGE

```
contract EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

}
```

Eternal Storage

Proxy

Logic

```
contract EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

}
```

Eternal Storage

Proxy

```
contract Poll is EternalStorage {

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}

}
```

Logic

```
contract EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

}
```

Eternal Storage

```
contract Proxy is EternalStorage {

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
 …
}
function voteForPepsi() returns(uint) {
 …
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

  mapping(bytes32⇒bool) _bool;
  mapping(bytes32⇒uint) _uint;
  mapping(bytes32⇒int) _int;
  mapping(bytes32⇒address) _address;
  mapping(bytes32⇒string) _string;
  mapping(bytes32⇒bytes) _bytes;

  address implementation;
  address owner;

  function setAddress(address _implementation) {
    implementation = _implementation;
  }

  function () payable public {
    … delegatecall() …
  }

}
```

Proxy

```
contract Poll is EternalStorage {

  mapping(bytes32⇒bool) _bool;
  mapping(bytes32⇒uint) _uint;
  mapping(bytes32⇒int) _int;
  mapping(bytes32⇒address) _address;
  mapping(bytes32⇒string) _string;
  mapping(bytes32⇒bytes) _bytes;

  function voteForCoke() returns(uint) {
    …
  }

  function voteForPepsi() returns(uint) {
    …
  }

}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

0

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

0
1

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

0	
1	
2	

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
 …
}
function voteForPepsi() returns(uint) {
 …
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;                    0
mapping(bytes32⇒uint) _uint;                    1
mapping(bytes32⇒int) _int;                      2
mapping(bytes32⇒address) _address;              3
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
 …
}
function voteForPepsi() returns(uint) {
 …
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

0	
1	
2	
3	
4	

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
 …
}
function voteForPepsi() returns(uint) {
 …
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

0	mapping(bytes32⇒bool) _bool;
1	mapping(bytes32⇒uint) _uint;
2	mapping(bytes32⇒int) _int;
3	mapping(bytes32⇒address) _address;
4	mapping(bytes32⇒string) _string;
5	mapping(bytes32⇒bytes) _bytes;

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
 …
}
function voteForPepsi() returns(uint) {
 …
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}

function voteForPepsi() returns(uint) {
…
}

}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

address implementation;
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

0	mapping(bytes32⇒bool) _bool;
1	mapping(bytes32⇒uint) _uint;
2	mapping(bytes32⇒int) _int;
3	mapping(bytes32⇒address) _address;
4	mapping(bytes32⇒string) _string;
5	mapping(bytes32⇒bytes) _bytes;

```
contract Poll is EternalStorage {

function voteForCoke() returns(uint) {
…
}

function voteForPepsi() returns(uint) {
…
}

}
```

Logic

Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;                    0
mapping(bytes32⇒uint) _uint;                    1
mapping(bytes32⇒int) _int;                      2
mapping(bytes32⇒address) _address;              3
mapping(bytes32⇒string) _string;                4
mapping(bytes32⇒bytes) _bytes;                  5

address implementation;                         6
address owner;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic

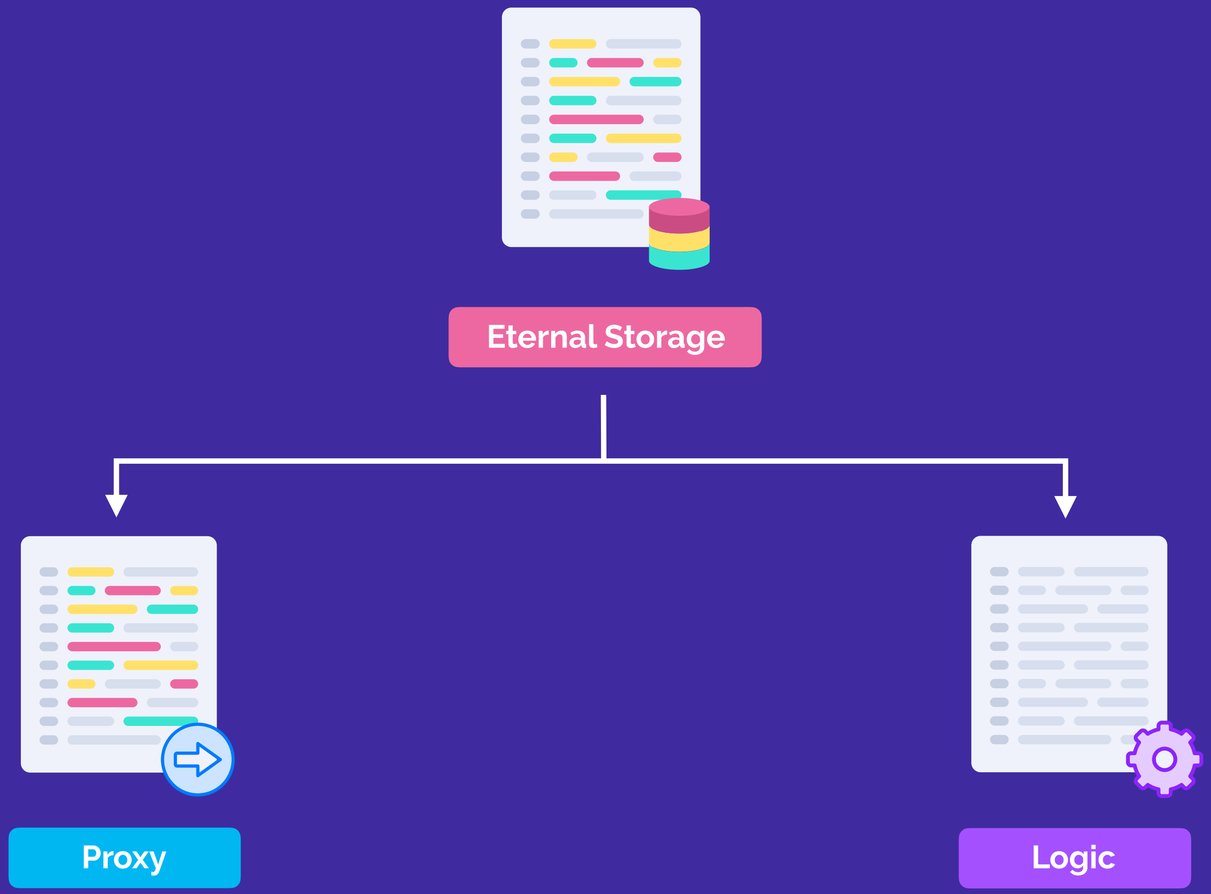
Eternal Storage

```
contract Proxy is EternalStorage {

mapping(bytes32⇒bool) _bool;                    0
mapping(bytes32⇒uint) _uint;                    1
mapping(bytes32⇒int) _int;                      2
mapping(bytes32⇒address) _address;              3
mapping(bytes32⇒string) _string;                4
mapping(bytes32⇒bytes) _bytes;                  5

address implementation;                         6
address owner;                                  7

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll is EternalStorage {

mapping(bytes32⇒bool) _bool;
mapping(bytes32⇒uint) _uint;
mapping(bytes32⇒int) _int;
mapping(bytes32⇒address) _address;
mapping(bytes32⇒string) _string;
mapping(bytes32⇒bytes) _bytes;

function voteForCoke() returns(uint) {
…
}
function voteForPepsi() returns(uint) {
…
}
}
```

Logic



ETERNAL STORAGE

👍 Very flexible data storage

ETERNAL STORAGE

👍 Very flexible data storage

👍 Assign all storage slots on first deployment

ETERNAL STORAGE

ETERNAL STORAGE

- 👍 Very flexible data storage
- 👍 Assign all storage slots on first deployment
- 👍 Logic contract can introduce new functions over time (although NOT variables)

👍 Very flexible data storage

👍 Assign all storage slots on first deployment

👍 Logic contract can introduce new functions over time (although NOT variables)

👍 The logic contract does not need to know about the proxy!

ETERNAL STORAGE

👎 Reading + writing to storage is more complex

ETERNAL STORAGE

👎 Reading + writing to storage is more complex

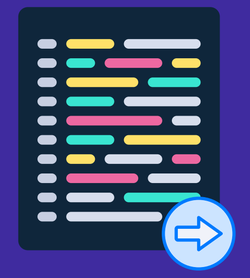
👎 Potentially more boilerplate code

ETERNAL STORAGE

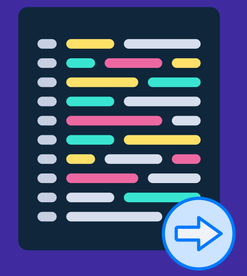
PROXY PATTERN:

UNSTRUCTURED STORAGE

UNSTRUCTURED STORAGE



UNSTRUCTURED STORAGE



Proxy

Logic

UNSTRUCTURED STORAGE

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}

function () payable public {
  …
  delegatecall(
    gas,
    implementation,
    ptr,
    calldatasize,
    0,
    0)
  …
}

}
```

Proxy

0×580 …

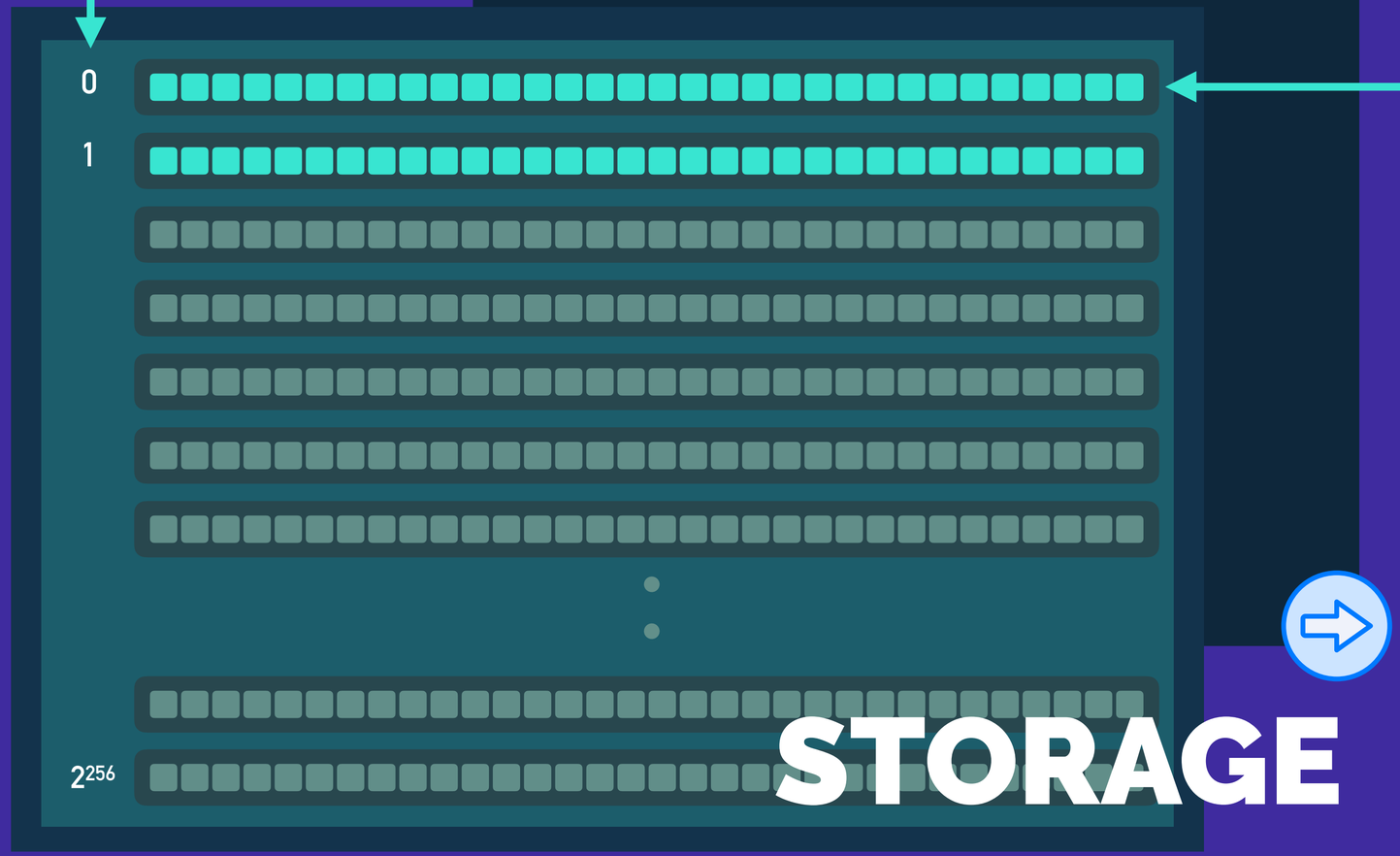
```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

0×732 …

```
contract Proxy {

address implementation;

function setAddress(address _implementation) {
  implementation = _implementation;
}
```

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

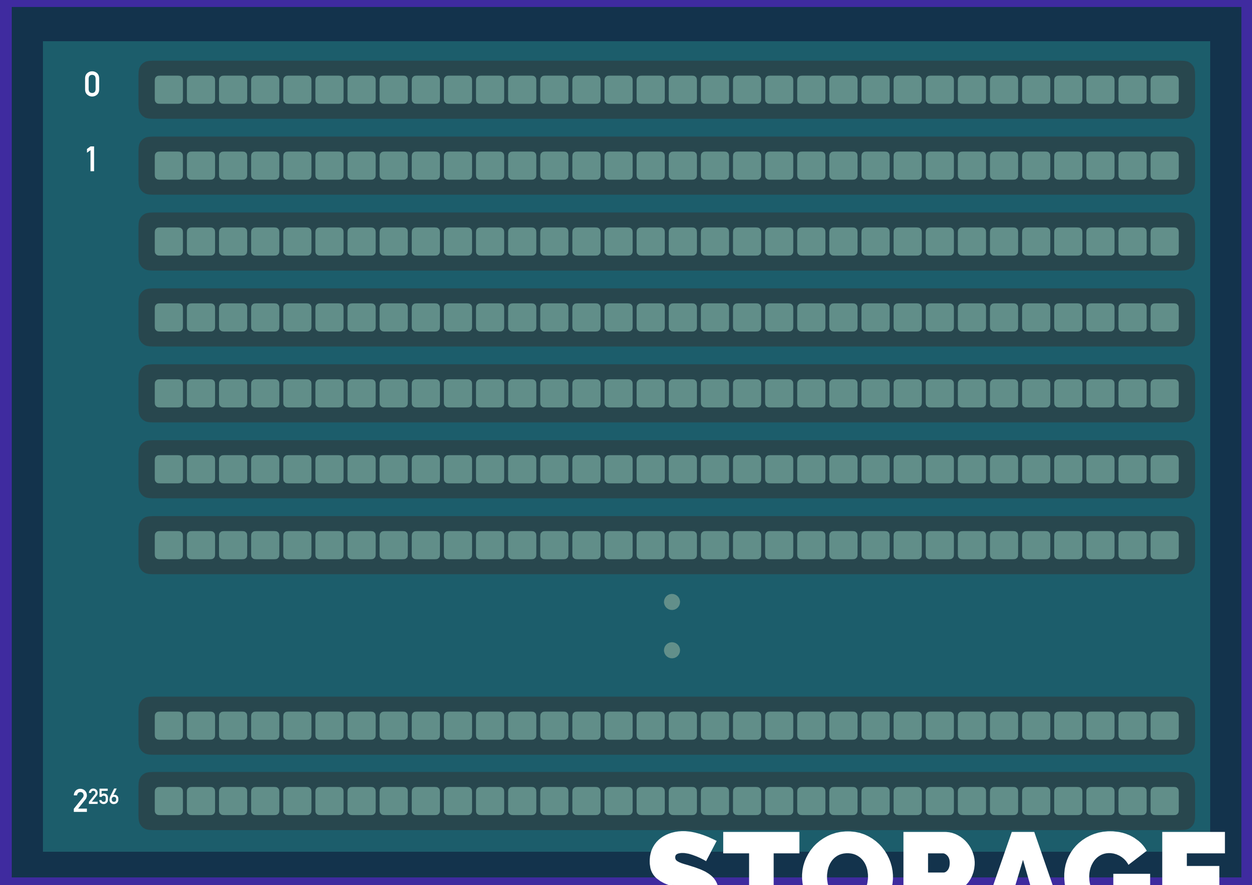


Poll

0×732 …

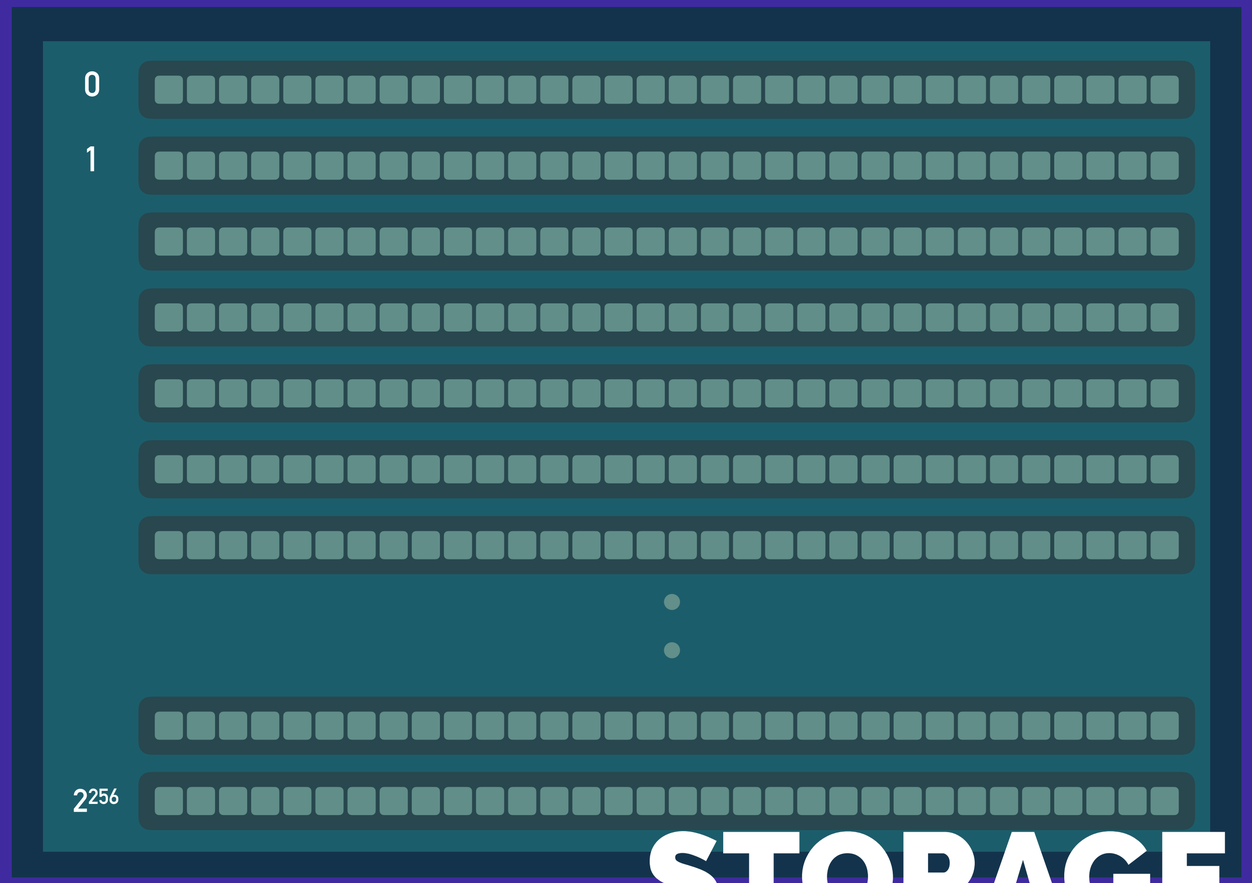
```
contract Proxy {

function () payable public {
  … delegatecall() …
}

}
```

Proxy



```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);









function () payable public {
  … delegatecall() …
}

}
```

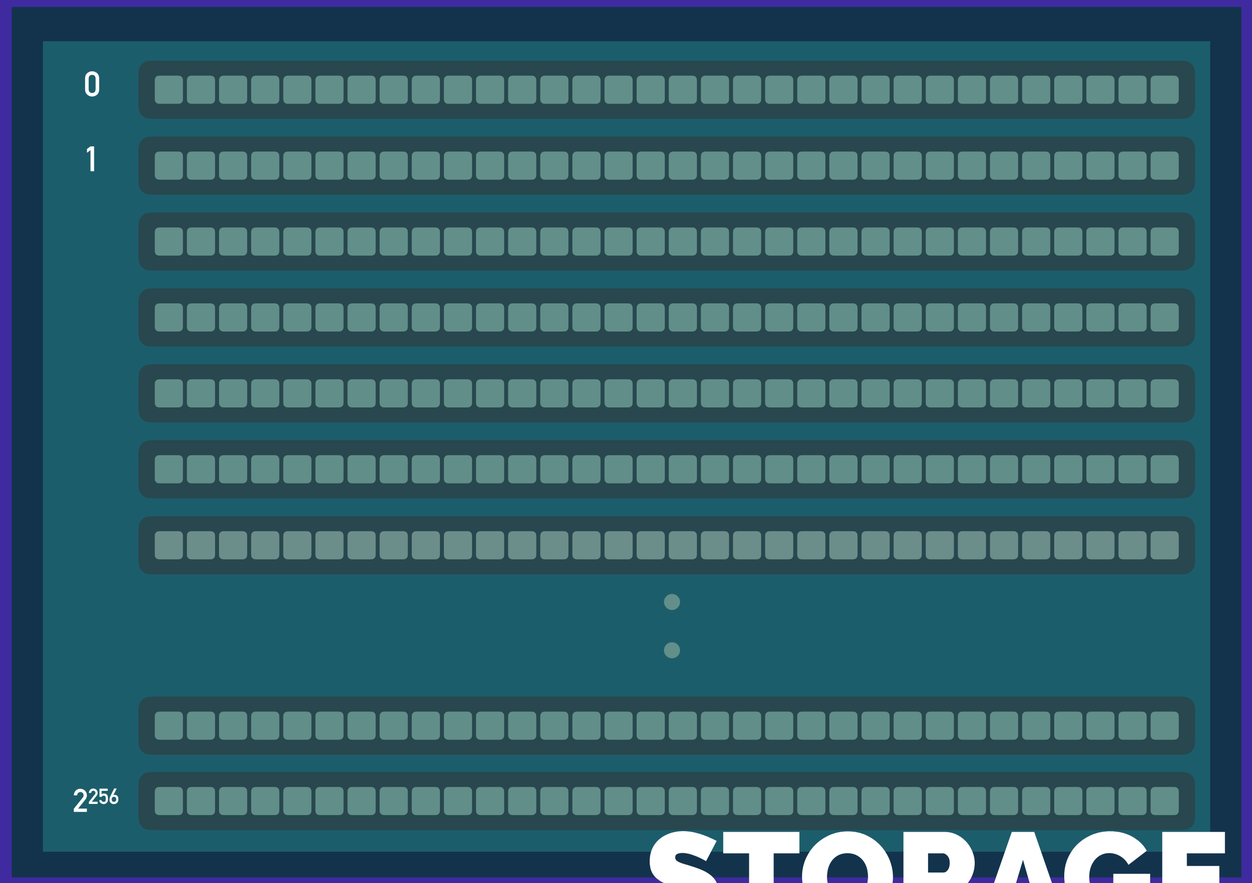
Proxy



```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}
```

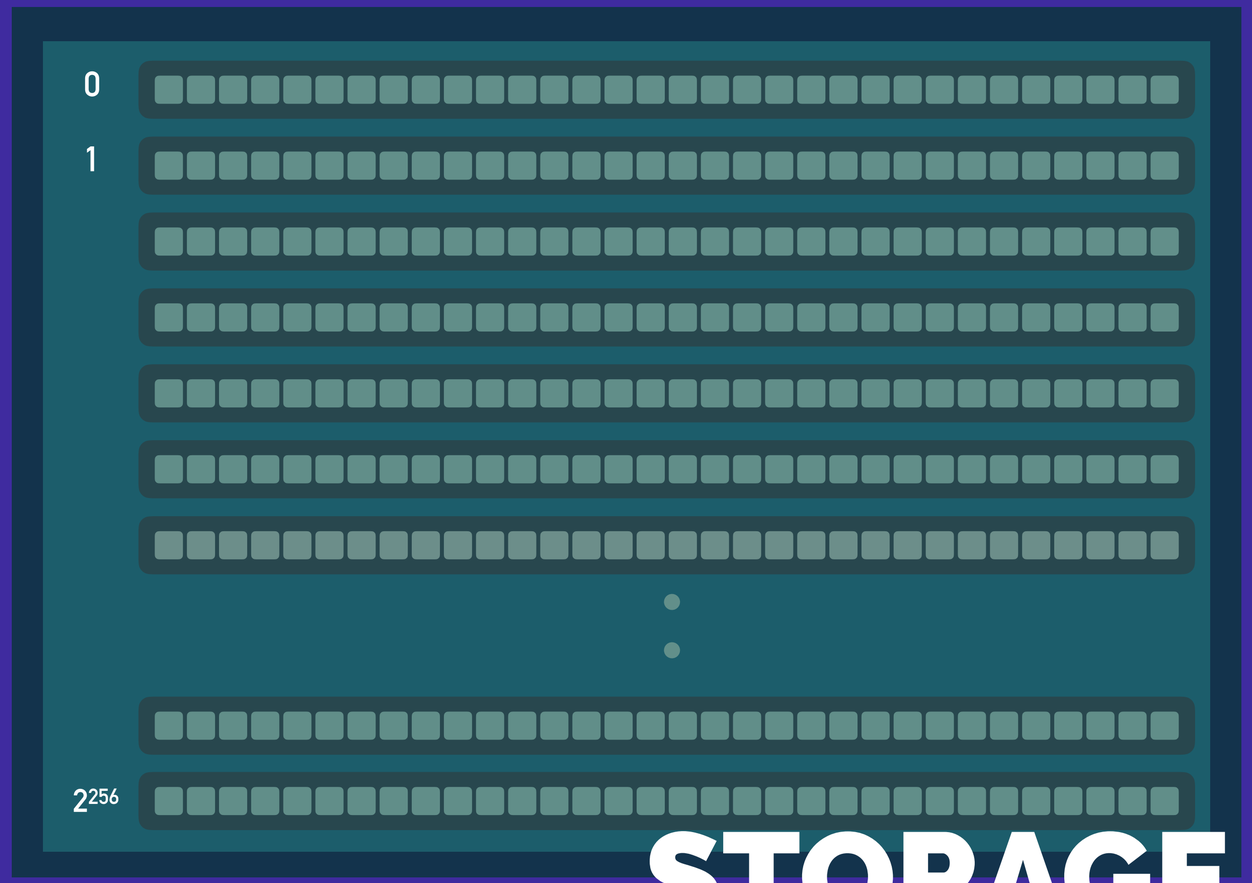
```
function () payable public {
  … delegatecall() …
}

}
```

Proxy

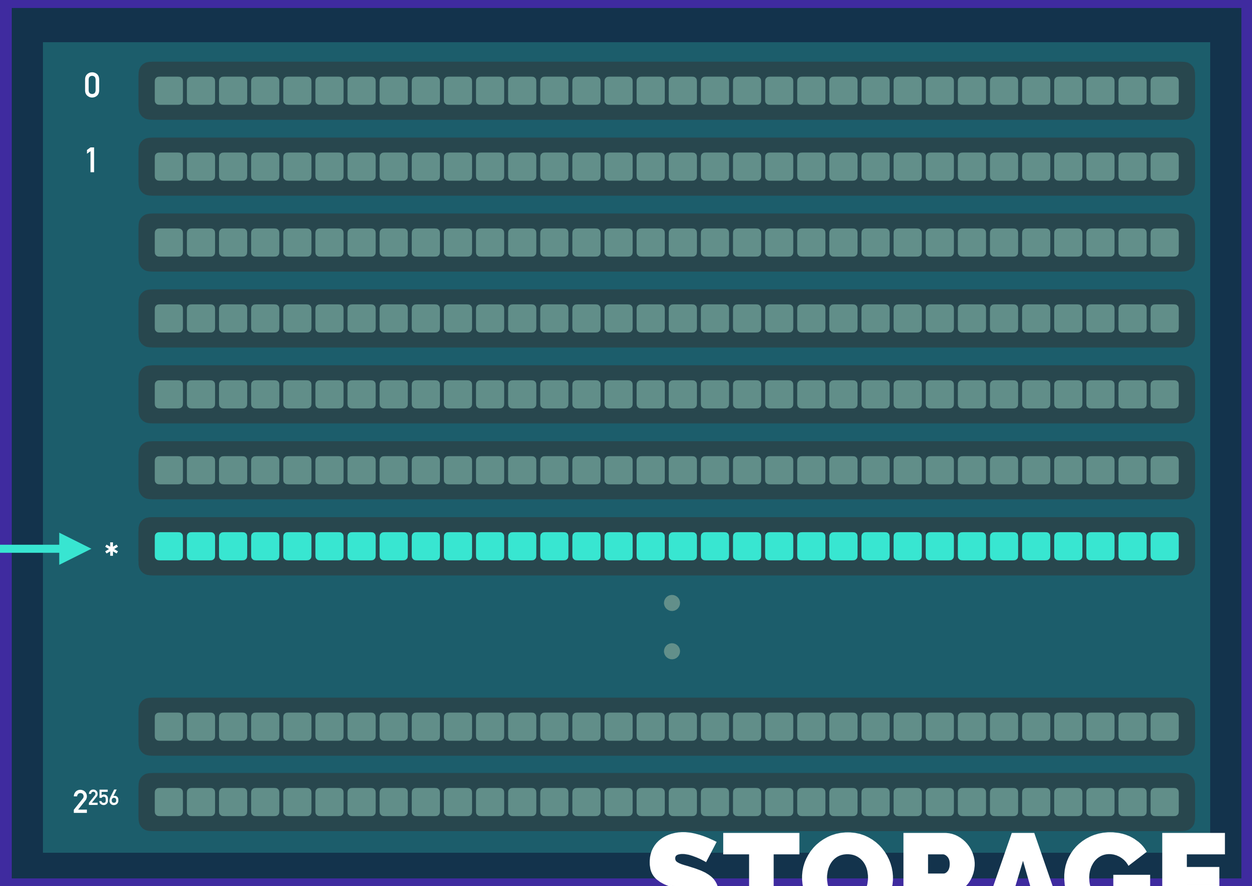


```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}









function () payable public {
  … delegatecall() …
}

}
```

Proxy



```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}




function () payable public {
  … delegatecall() …
}

}
```

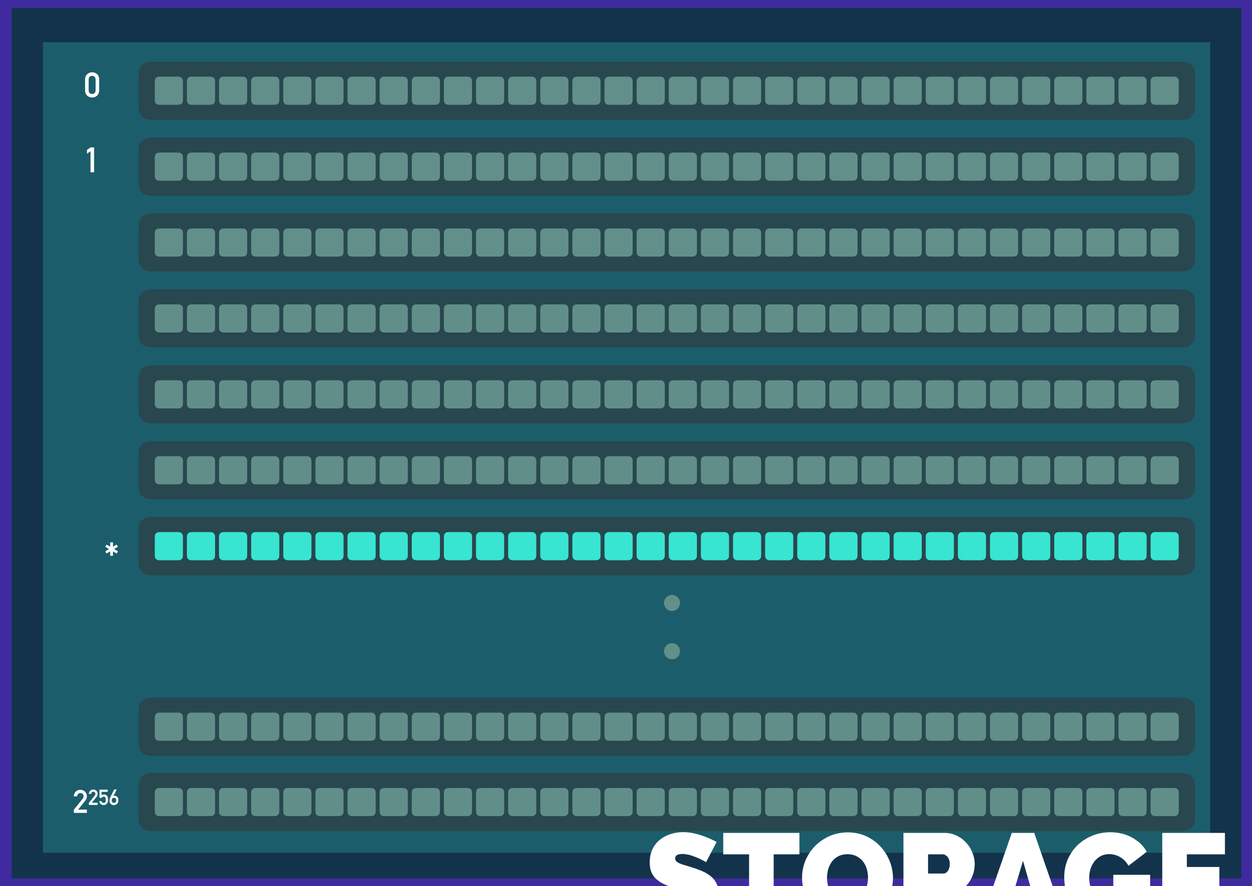


* keccak256(“proxy.implementation”)

Proxy

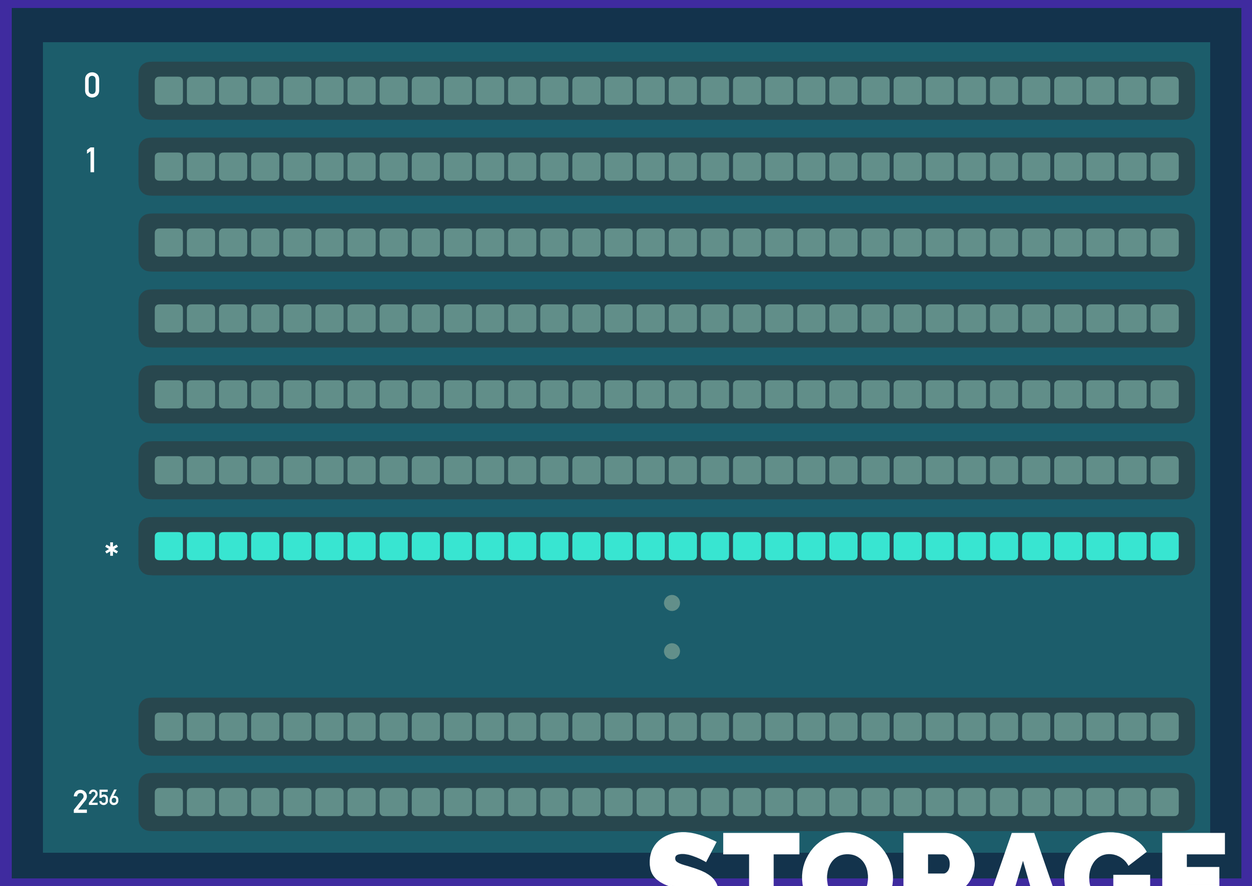
```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}

function getAddress() returns(address _implementation) {
  assembly {
    _implementation := sload(implementationPos)
  }
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy



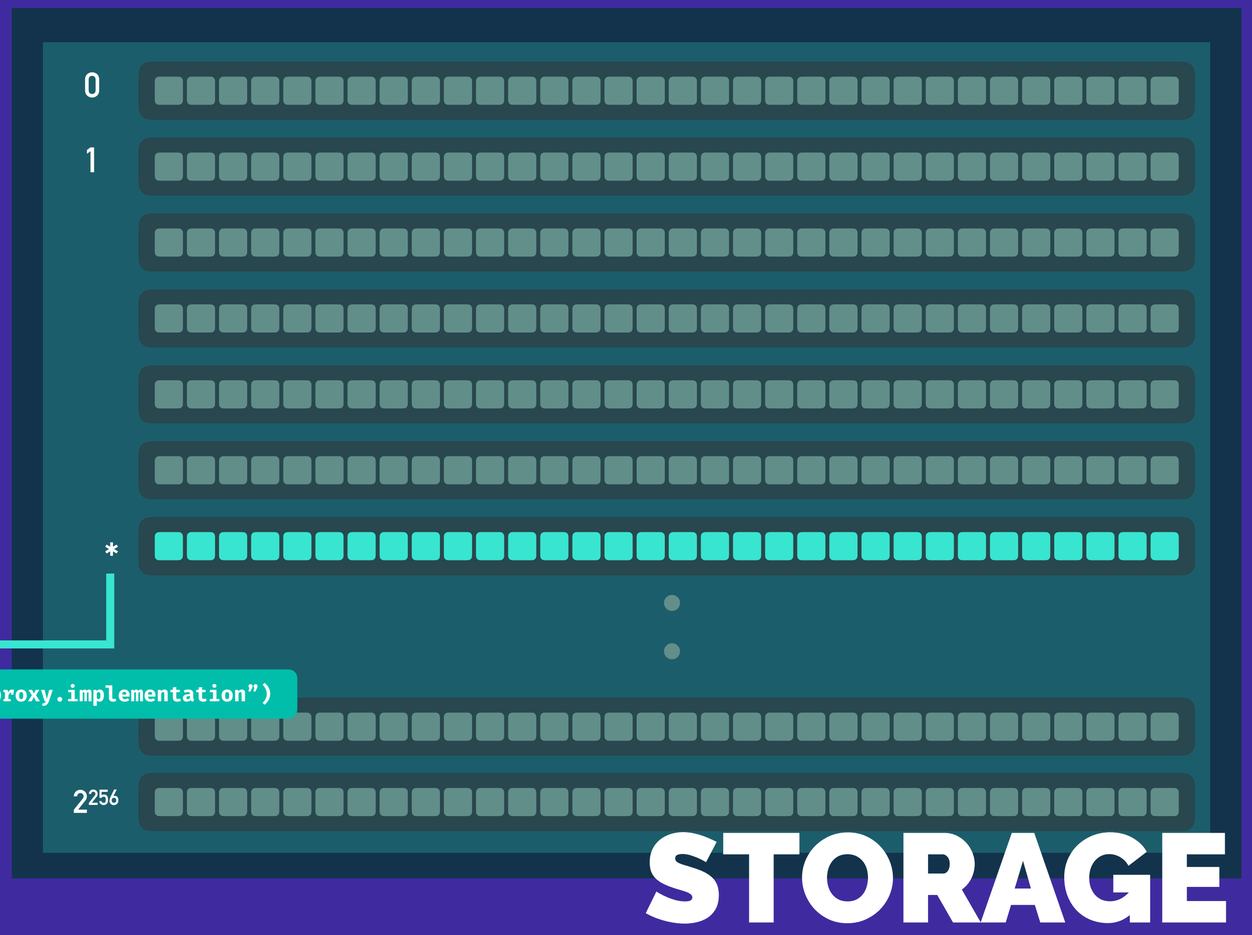
```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}

function getAddress() returns(address _implementation) {
  assembly {
    _implementation := sload(implementationPos)
  }
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy



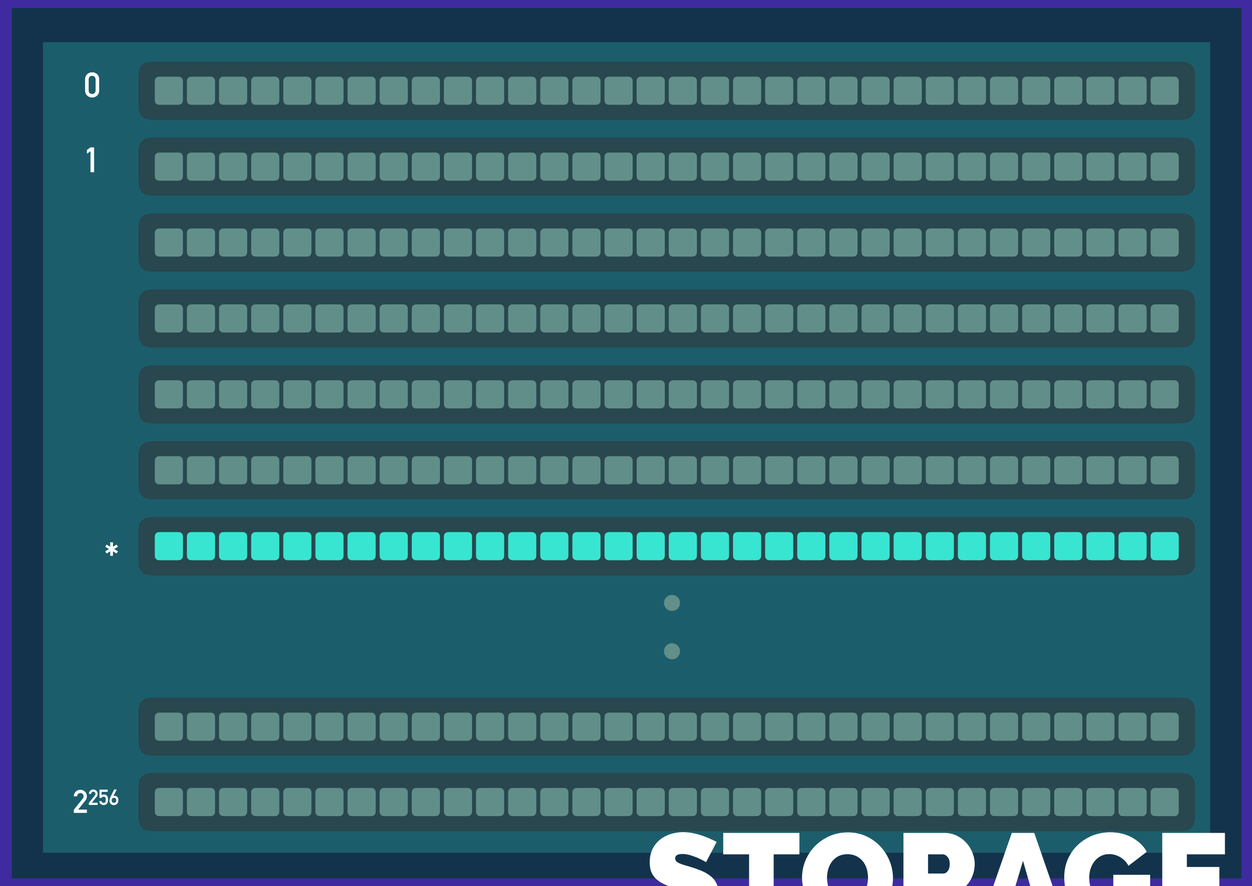
```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}

function getAddress() returns(address _implementation) {
  assembly {
    _implementation := sload(implementationPos)
  }
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy



```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}

function getAddress() returns(address _implementation) {
  assembly {
    _implementation := sload(implementationPos)
  }
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy



```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}

function getAddress() returns(address _implementation) {
  assembly {
    _implementation := sload(implementationPos)
  }
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

Poll

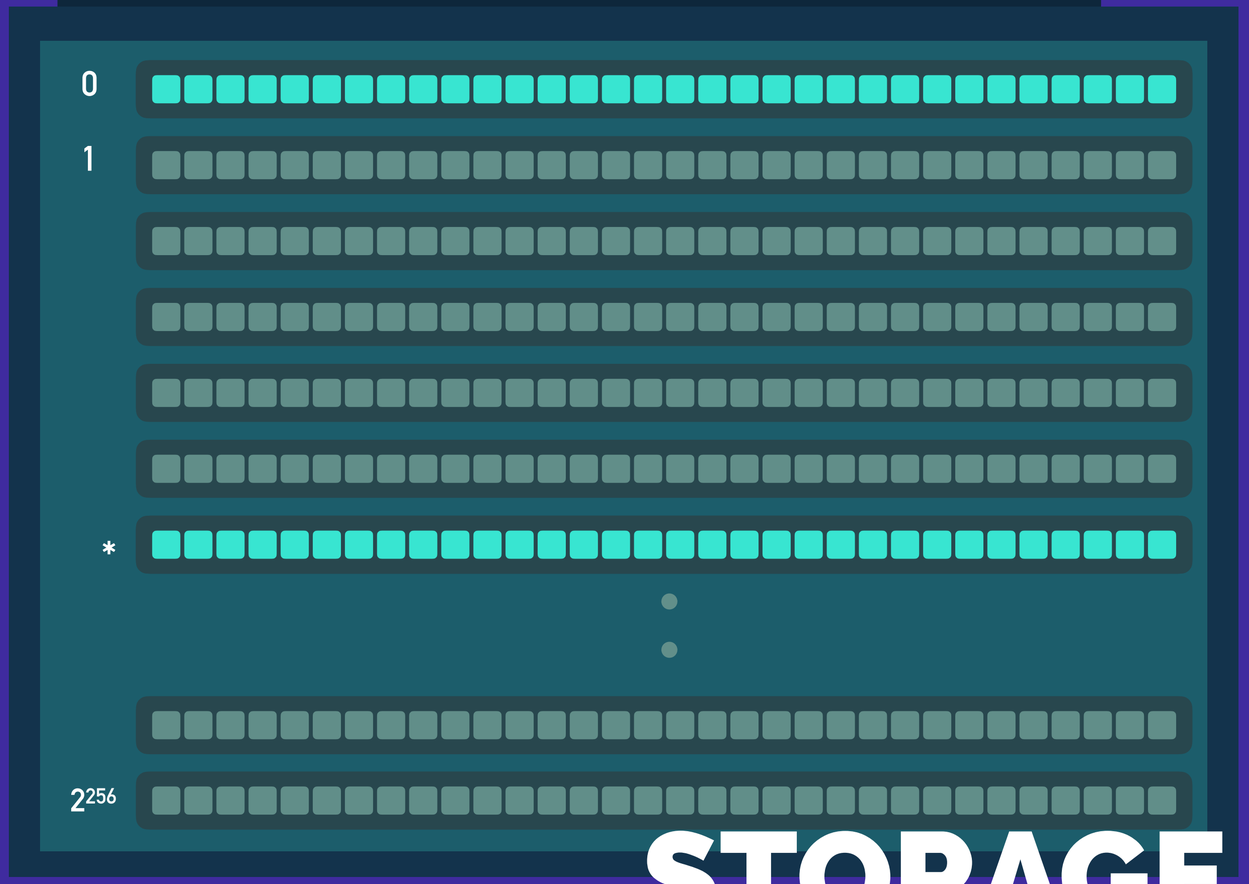
```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
    sstore(implementationPos, _implementation)
  }
}

function getAddress() returns(address _implementation) {
  assembly {
    _implementation := sload(implementationPos)
  }
}

function () payable public {
  … delegatecall() …
}

}
```

Proxy

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
```

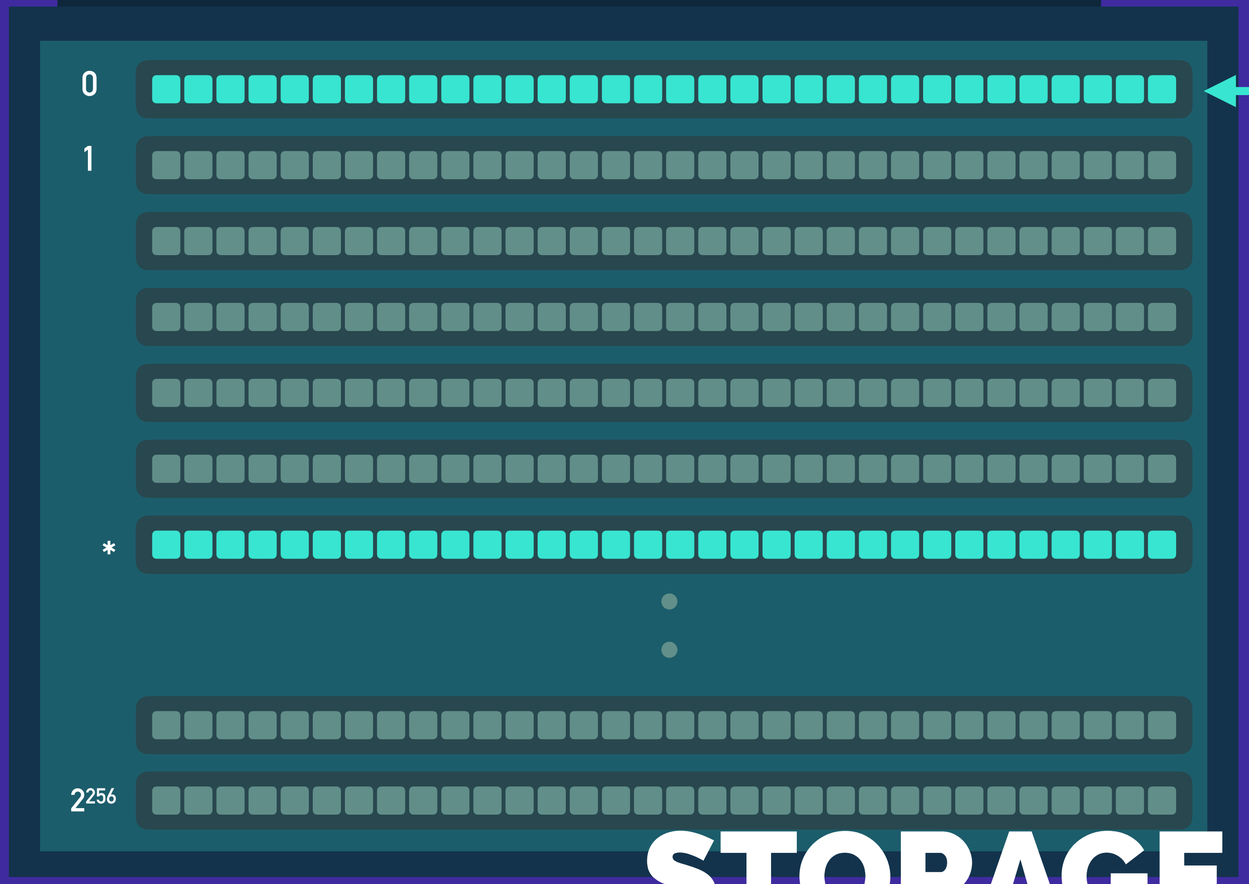


STORAGE

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll

```
contract Proxy {

bytes32 private constant implementationPos =
            keccak256(“proxy.implementation”);


function setAddress(address _implementation) {
  assembly {
```

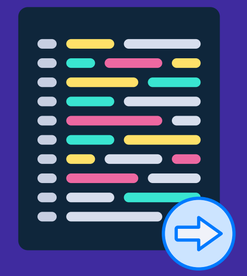


STORAGE

0

```
contract Poll {

uint cokeVotes = 0;
uint pepsiVotes = 0;

function voteForCoke() returns(uint) {
  cokeVotes++;
  return cokeVotes;
}

function voteForPepsi() returns(uint) {
  pepsiVotes++;
  return pepsiVotes;
}

}
```

Poll



Proxy

Logic

UNSTRUCTURED STORAGE

👍 No common superclass inheritance

UNSTRUCTURED STORAGE

👍 No common superclass inheritance

👍 Makes it easy + safe for the logic contract to implement whatever variables it may need

UNSTRUCTURED STORAGE

👍 No common superclass inheritance

👍 Makes it easy + safe for the logic contract to implement whatever variables it may need

👍 Logic contract can introduce new functions and variables over time

UNSTRUCTURED STORAGE

👍 No common superclass inheritance

👍 Makes it easy + safe for the logic contract to implement whatever variables it may need

👍 Logic contract can introduce new functions and variables over time

👍 The logic contract does not need to know about the proxy!

UNSTRUCTURED STORAGE

👎 Careful about using inline assembly

UNSTRUCTURED STORAGE

👎 Careful about using inline assembly

👎 No compiler warnings

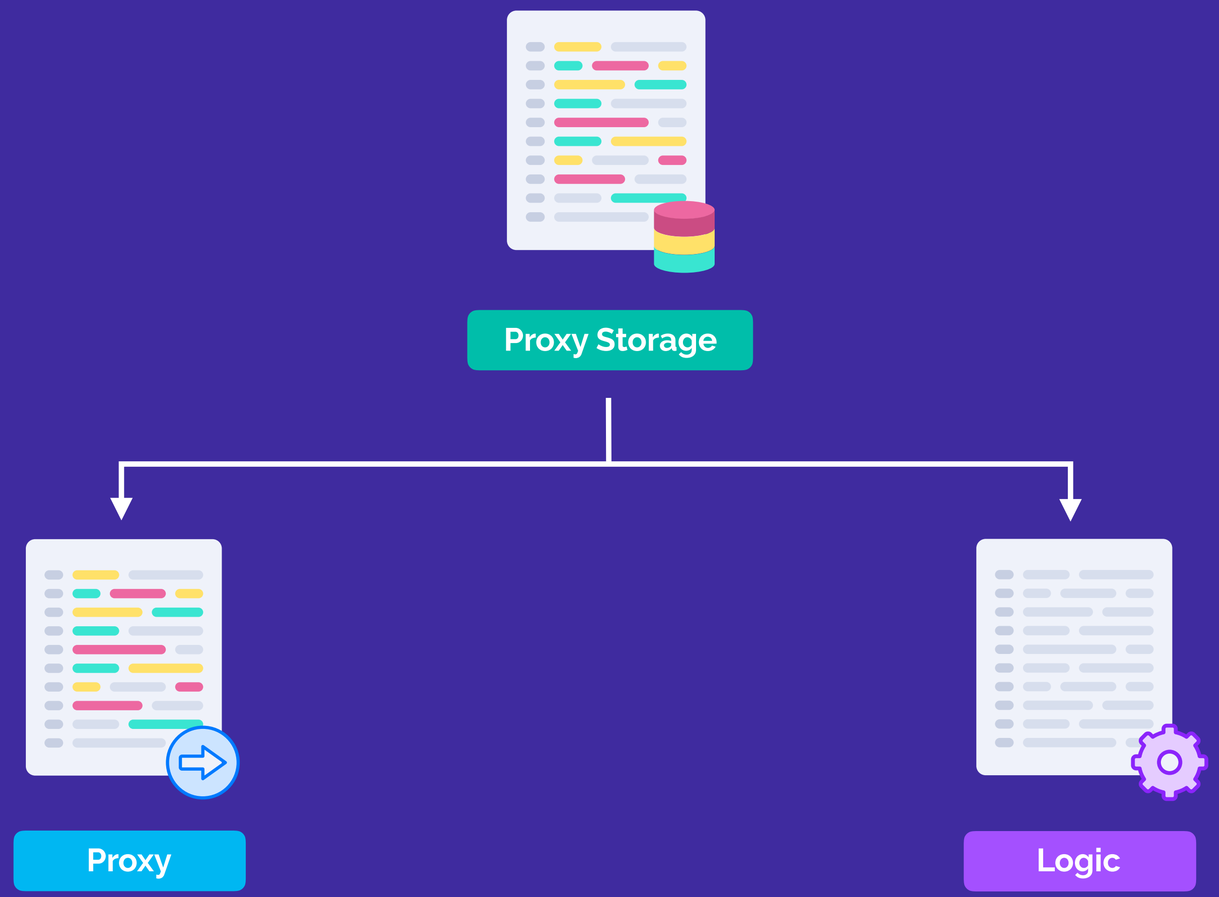
UNSTRUCTURED STORAGE

CONCLUSION

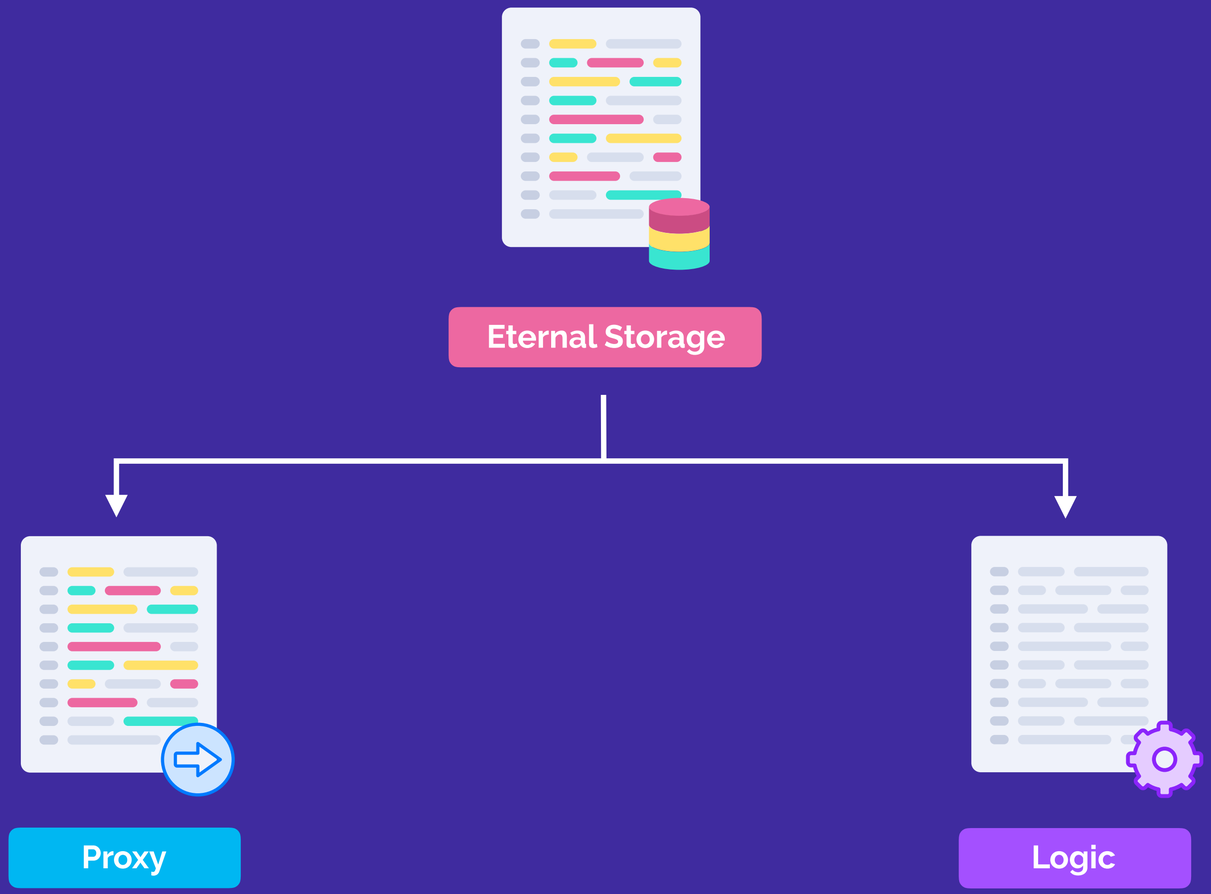
CONTRACT UPGRADES ARE POSSIBLE

HUB & SPOKE

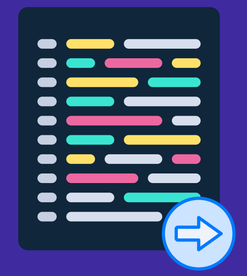
PROXY PATTERN



INHERITED STORAGE



ETERNAL STORAGE



Proxy

Logic

UNSTRUCTURED STORAGE

SHOULD WE DO ANY OF THIS?

IT *DEPENDS*...

👍 Fix bugs

SHOULD WE DO THIS?

- 👍 Fix bugs
- 👍 Improve functionality

SHOULD WE DO THIS?

👍 Fix bugs

👍 Improve functionality

👎 Changes the rules of our logic

SHOULD WE DO THIS?

👍 Fix bugs

👍 Improve functionality

👎 Changes the rules of our logic

🤨 Is it still immutable?

SHOULD WE DO THIS?

DO THESE CHANGES BENEFIT OUR USERS?

Logic_v1

Should we update to v2 ?

Logic_v1

Logic_v2

43% 👍

57% 👎

Logic_v1

Logic_v2

43% 👍

57% 👎

Should we update to v3 ?

Logic_v1

Logic_v2

43% 👍

57% 👎

Logic_v3

71% 👍

29% 👎

Logic_v1

Logic_v2

43% 👍

57% 👎

Logic_v3

71% 👍

29% 👎

ABOVE ALL...


We all face the same questions every day:
What do I want? And how can I get it
How can I live more happily
and work more effectively?
The Decision Book
Fifty models for strategic thinking
Stylish and compact, this little black book is a powerful asset. Whether you need to plan a presentation, assess someone's business idea or get to know yourself better, this unique guide – full of helpful visual tools – will help you simplify any problem and take you towards the right
The Decision Book

USE THIS RESPONSIBLY

THANK YOU!

Matias Seijas
@mseijas