

Smart Contracts con Solidity, Ethereum

Primeros Recursos Recomendados

<https://github.com/bitcoinbook/bitcoinbook>

<https://github.com/ethereumbook/ethereumbook>

Recursos adicionales sobre Ethereum:

<https://github.com/ajlopez/AprendiendoSolidity>

<https://github.com/ajlopez/SoliditySamples>

Agenda

- Ethereum
- Smart Contracts
- Estado de Smart Contracts
- Lenguaje de Programación Solidity
- Elementos de web3js
- Usando truffle framework
- Concepto de DApp
- Demos

Ethereum

Ethereum

<https://ethereum.org/>



Ethereum

Fork This Page



→ **Beginners**

Completely new
to Ethereum?

→ **Learn**

Want to dig in and learn
more about Ethereum?

→ **Use**

Want to start
using Ethereum?

→ **Developers**

Looking to build
on Ethereum?



Welcome to the ethereum.org redesign! → [More](#)

Aplicaciones Distribuidas: Características Deseables

- Trabajar en conjunto
- Sin parar
- No hay nodo líder
- Agregar nodo
- Remover nodo

Aplicaciones Distribuidas: Características Deseables

- Trabajar en conjunto
- Sin parar
- No hay nodo líder
- Agregar nodo
- Remover nodo

Caso Blockchain

- Estado compartido
- Geográficamente distribuidos
- Sin permisos
 - cualquiera puede participar como nodo
- Algoritmo de consenso

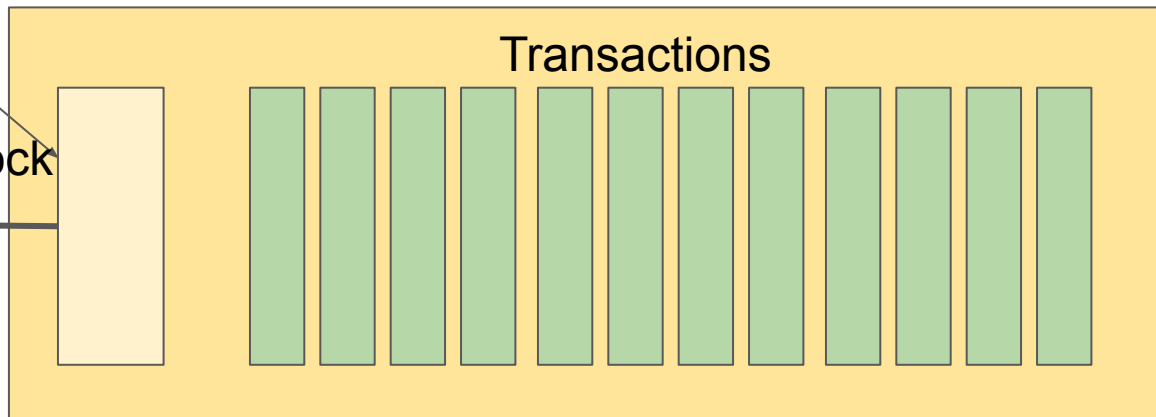
Ethereum Blockchain

- Nodos Distribuidos
- Bloques
- Transacciones
- Estado Compartido
- Consenso
- Contratos Inteligentes

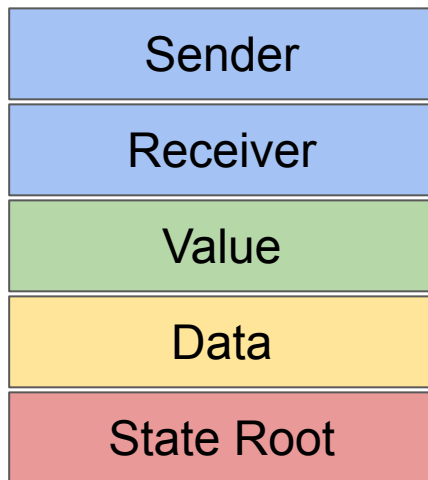
Block

Header

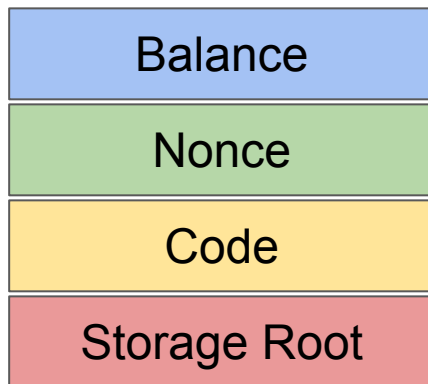
To Parent Block



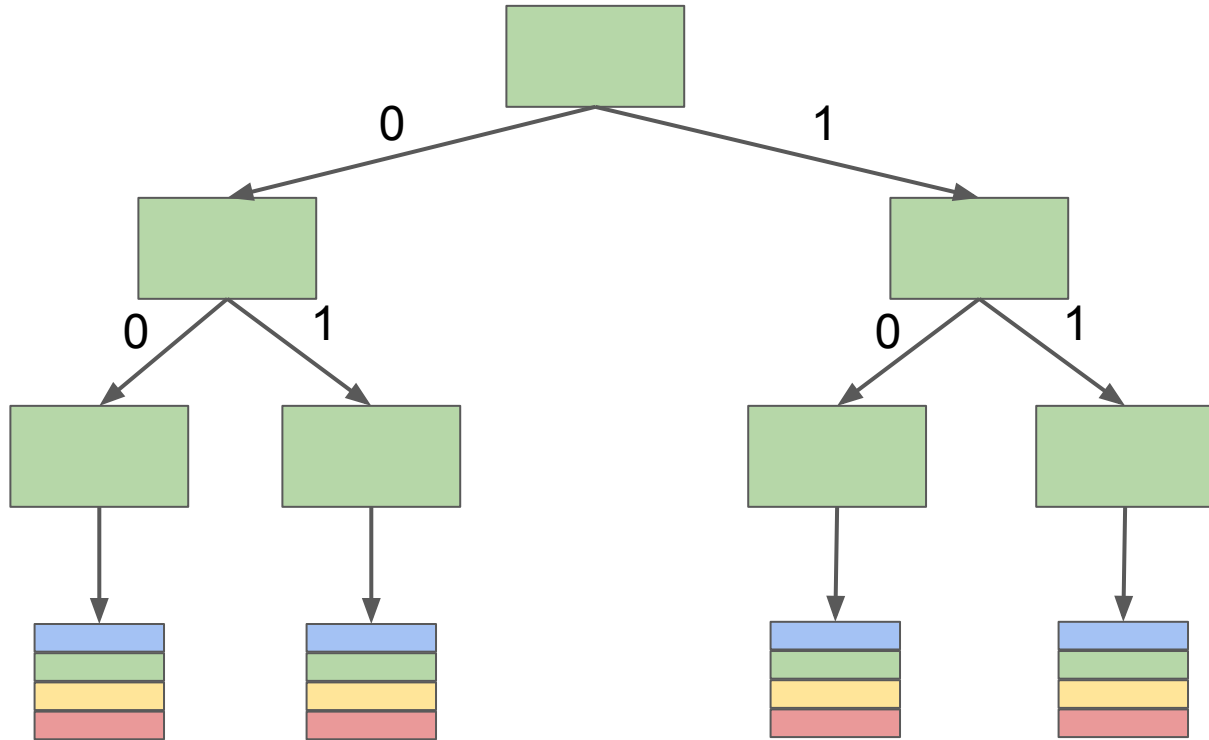
Transaction



Account



Estado en Trie



Ejecutando un Nodo

Ganache CLI

```
npm install -g ganache-cli
```

```
ganache-cli --verbose
```

<https://github.com/trufflesuite/ganache-cli>

Exeth

<https://github.com/ajlopez/exeth>

```
npm install -g exeth
```

```
exeth <filename>
```


Primeros Ejemplos

- <https://github.com/ajlopez/SoliditySamples/tree/master/eth/simple>
- Conseguir las cuentas (accounts.eth)
- Conseguir los balances (balances.eth)
- Transferencia (transfer.eth)
- Envío y recepción de JSON RPC
<https://github.com/ethereum/wiki/wiki/JSON-RPC>
-

Smart Contracts

Contrato

```
contract Counter {  
    uint counter;  
    function Counter() {  
        counter = 1;  
    }  
}
```

Smart Contracts en Ethereum

- Son Cuentas
- Con saldo
- Con estado
- Con código
- Se ejecuta alguna función al enviar una transacción

Comunicación con el Nodo

- Mediante JSON RPC
- Protocolo de Remote Procedure Call
- Implementado en varias tecnologías
- Bitcoin expone JSON RPC
- Ethereum expone JSON RPC

Ethereum Virtual Machine

- Basada en bytecodes
- Accede y actualiza el estado de un contrato
- Usa memoria transitoria
- Usa una pila de valores
- Consume gas por opcode/almacenamiento

Almacenamiento

- Clave/Valor
- Son de 32 bytes cada uno
- Valor asumido es 0x00
- La pila también maneja valores de 32 bytes

Solidity

- Lenguaje de programación dedicado a Smart Contracts
- Contratos (similar a clases)
- Posibilidad de herencia
- Librerías
- Lenguaje tipado
- Cada instancia de contrato tiene
 - Código compilado
 - Dirección
 - Estado

Creando Instancia de Contrato

- Se envía el código y los argumentos de constructor en una transacción
- Se le asigna una dirección (20 bytes en Ethereum)
- Al crearlo, se le puede enviar éter

Invocando a Contrato

- Se invoca a una función, indicando su hash y sus argumentos, en una transacción dirigida a la dirección del contrato

Enteros

```
// signed integer (32 bytes)
```

```
int signed;
```

```
// unsigned integer (32 bytes)
```

```
uint unsigned;
```

```
// integer with bit size
```

```
uint16 short;
```

Arreglos de Longitud Fija

```
// fixed size arrays
```

```
bytes1 onebyte;
```

```
bytes20 twentybytes;
```

```
bytes32 thirtytwobytes;
```

```
int[10] tenintegers;
```

```
string[5] fivemessages;
```

Arreglos Dinámicos

```
// dynamically-sized arrays
```

```
bytes data;
```

```
string message;
```

```
function f(uint len) {
```

```
    uint[] memory a = new uint[] (7);
```

```
    bytes memory b = new bytes(len);
```

```
}
```

Estructuras

```
struct Voter {  
    address delegate;  
    bool voted;  
}
```

```
struct Proposal {  
    bytes32 name;  
    uint voteCount;  
}
```

Maapeos

```
mapping (address => uint) public balances;
```

Remix Solidity IDE (browser)

The screenshot displays the Remix Solidity IDE interface. The main editor shows a Solidity contract named `Ballot` with the following code:

```
1 pragma solidity ^0.4.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17
18    /// Create a new ballot with $(_numProposals) different ;
19
```

The right sidebar contains configuration options for the environment and execution:

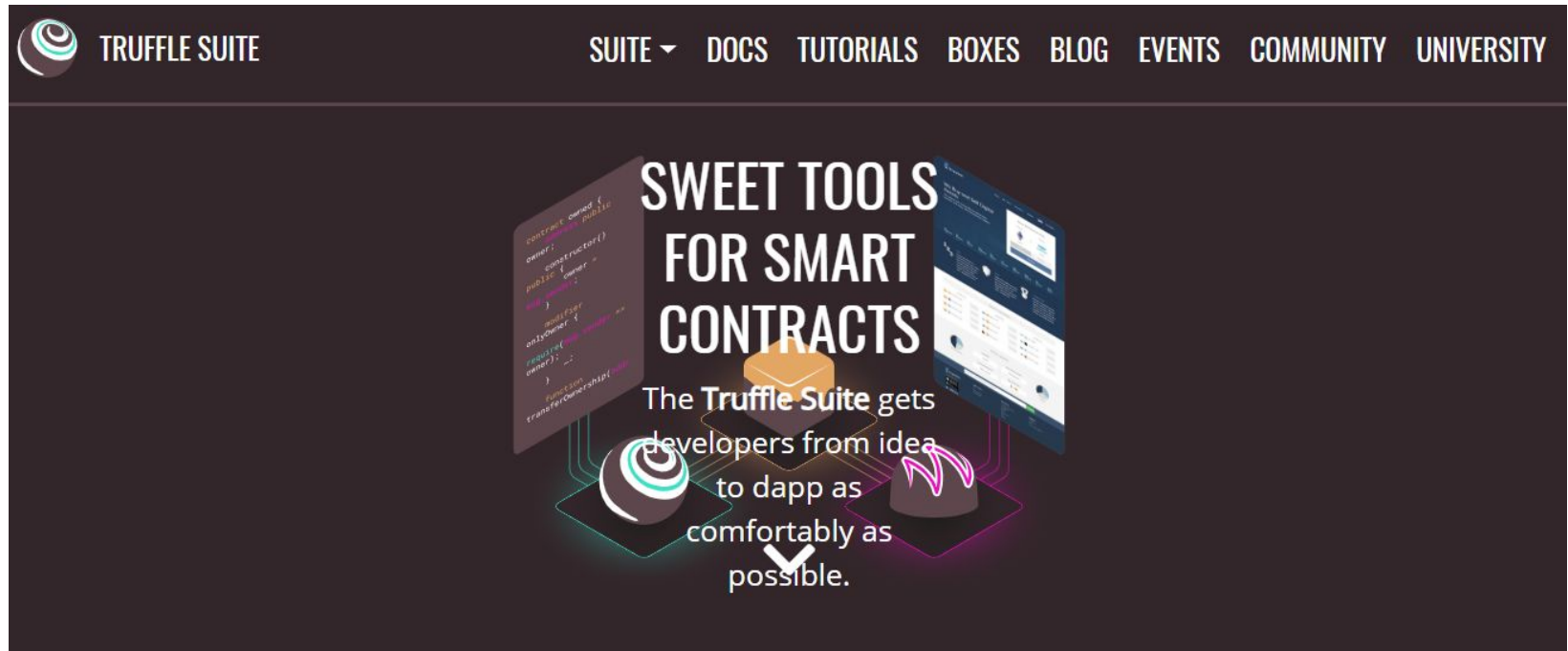
- Environment:** JavaScript VM
- Account:** 0xca3...a733c (100 €)
- Gas limit:** 3000000
- Gas Price:** 0
- Value:** 0

Below these options, the selected contract is `browser/ballot.sol:Ballot`. The **At Address** field is set to `Enter contract's address`, and the **Create** button is labeled `uint8 _numProposals`.

The bottom status bar shows `[2] only remix transactions, script` and a `Listen on network` button. A box at the bottom right indicates `0 pending transactions`.

Truffle Framework o Harhat

<http://truffleframework.com/>





Hardhat



Truffle

<https://www.linkedin.com/pulse/descubre-las-diferencias-entre-hardhat-y-truffle-cu%C3%A1l-bravo-cuero/?originalSubdomain=es>

Truffle Framework

- Proyectos basados en Truffle
- Provee
 - Deploy/Migrate
 - Test (en JavaScript/Solidity)
 - Builtin Client
 - Consola

Instalación

http://truffleframework.com/docs/getting_started/installation

http://truffleframework.com/docs/getting_started/installation#recommendations-for-windows

```
npm install -g truffle
```

Primer Proyecto

http://truffleframework.com/docs/getting_started/project

```
mkdir myproj
```

```
cd myproj
```

```
truffle init
```

```
// alternative
```

```
truffle unbox <project>
```

Comandos

```
truffle compile
```

```
truffle test
```

```
truffle develop
```

Tutoriales y Boxes

<http://truffleframework.com/tutorials/>

<http://truffleframework.com/boxes/>

Web3

- API JavaScript para acceder a un nodo Ethereum
- <https://github.com/ethereum/web3.js/>
- Documentación 1.10.x
<https://web3js.readthedocs.io/en/v1.10.0/>
- Documentación 0.2x.x
<https://github.com/ethereum/wiki/wiki/JavaScript-API>
- Implementada en otros lenguajes (ej: Java)

Ejemplos Web3JS

- <https://github.com/ajlopez/SoliditySamples>
- Directorio web31 (versión 1.0.0beta...)
- Concepto de provider
- Uso de Solc (paquete adicional) para compilar
- Deploy de contrato
- Instancia JavaScript de contrato
- Invocación de Contrato

OpenZeppelin

<https://openzeppelin.org/>



[GitHub](#) [Docs](#) [ZeppelinOS](#) [Forum](#) ●

Build Secure Smart Contracts in Solidity

OpenZeppelin is a battle-tested framework of reusable smart contracts for Ethereum and other EVM and eWASM blockchains.

GET STARTED

CryptoKitties

<https://www.cryptokitties.co/>



CryptoKitties

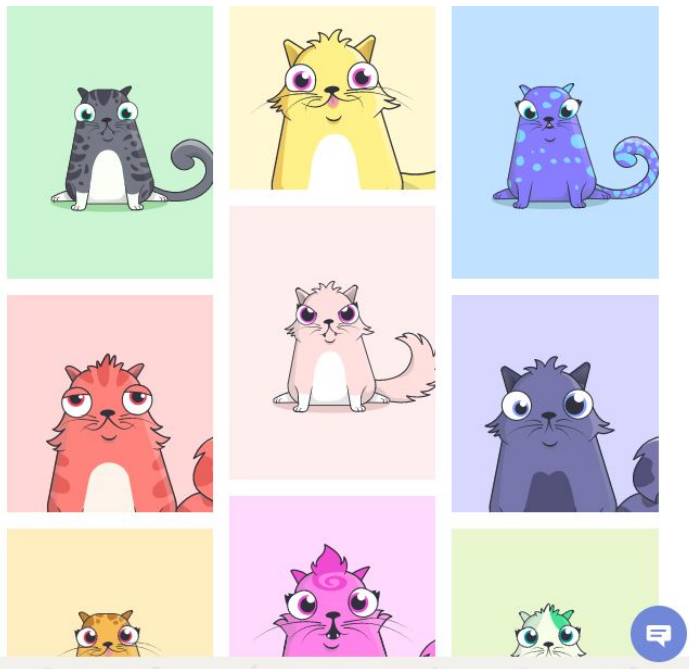
Sign in

Marketplace

**Collectible.
Breederable.
Adorable.**

Collect and breed digital cats.

Start meow



Futuro

- Nuevos lenguajes (viper, ...)
- **Killing Dapp**
- Otras máquinas virtuales (NEO VM,)
- Escalabilidad
- Otros algoritmos de consenso
- Blockchain sin bloques ni cadenas (Iota, HashGraph, ...)