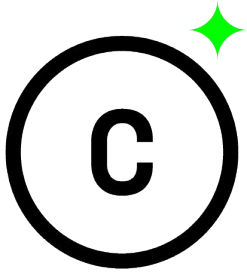
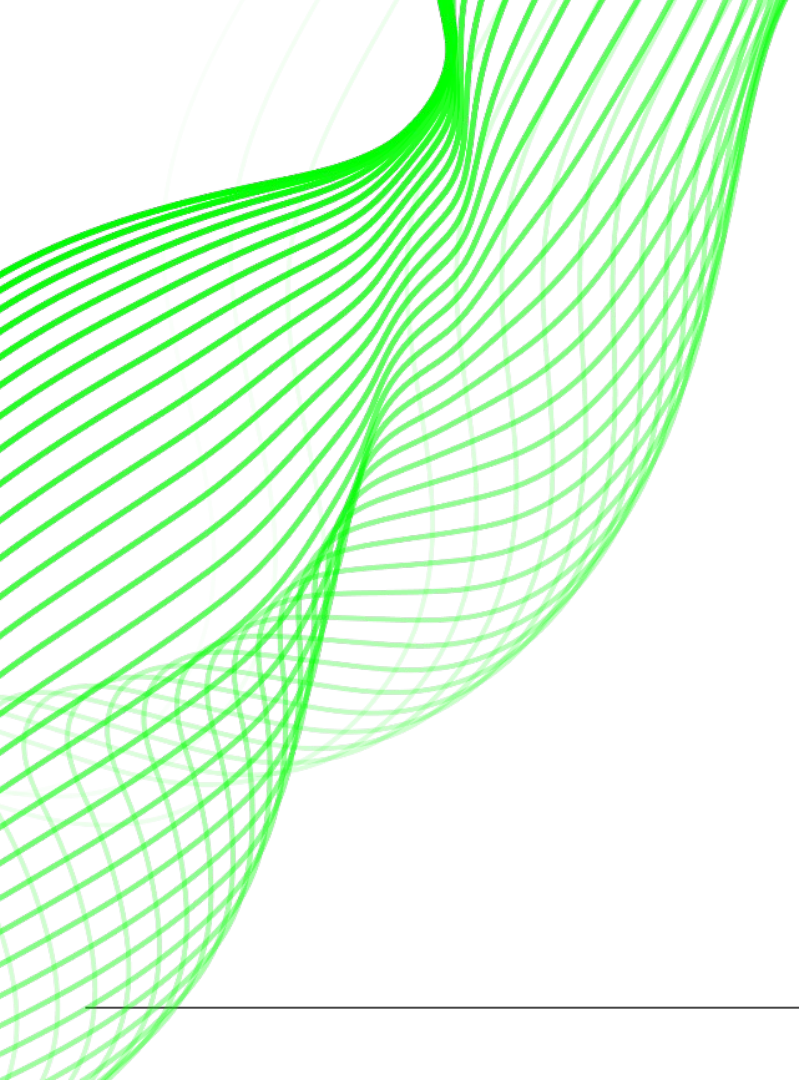


COLLEDGE

web 3.0 community

Desarrollo de contratos inteligentes en Solidity

Sebastián Pérez



COLLEDGE
web 3.0 community

Aspectos básicos de los contratos inteligentes y Solidity

Blockchain

COLLEDGE
web 3.0 community

- Tecnología impulsada por Bitcoin
- Red distribuida
- Refuerza el concepto de descentralización
- Registros inmutables
- Seguridad basada en criptografía

Ethereum

COLLEDGE
web 3.0 community

- Agrega a las transacciones la capacidad de ejecutar software
- Utiliza la EVM para interpretar el código
- Conserva las características de Blockchain
- Utiliza PoW para lograr consenso
- En proceso de migrar a su versión 2.0

Nodos/Clientes

COLLEDGE
web 3.0 community

- Existen diversas implementaciones para los nodos de Ethereum
- La más utilizada es Go Ethereum (geth)
- Corresponden al mismo protocolo pero realizan distintos procesos de sincronización
- En general son open source
- Más versiones: Nethermind, Hyperledger Besu, Erigon, Open Ethereum

Contratos inteligentes

COLLEDGE
web 3.0 community

- Piezas de software que se ejecutan en los nodos
- Se los llama contratos por su similitud con los mismos
- Necesitan cierta interacción de parte de los usuarios
- Garantizan consenso de forma inmediata, agilizando procesos de validación que suelen ser lentos

Lenguaje Solidity

COLLEDGE
web 3.0 community

- Creado especialmente para el desarrollo de contratos de Ethereum
- Orientados a contratos (similar a POO)
- Sintaxis similar a C++/Javascript
- Open Source
- Bien documentado
- Última versión: v0.8.15

Estructura de un contrato

COLLEDGE
web 3.0 community

- Licencia (opcional)
- Versión del compilador (pragma)
- Declaración del contrato
- Estado (variables)
- Funciones

Licencia

COLLEDGE
web 3.0 community

- Parte de la confianza se basa en si nuestro contrato es abierto
- Por eso opcionalmente se pide ingresar el tipo de licencia
- No existe una validación del valor que se ingresa
- Puede ingresarse el valor “UNLICENSED” si no deseamos especificar ninguna licencia

```
// SPDX-License-Identifier: MIT
```

Versión del compilador

COLLEDGE
web 3.0 community

- Se especifica a través de la palabra pragma
- Puede definirse una versión puntual o un rango
- Una buena práctica es que el rango no sea mayor a dos versiones
- La versión actual es la 0.8.6 pero se suele poner <0.9 así si sale una nueva sub versión no es necesario corregir el contrato

```
pragma solidity ^0.5.2;

pragma solidity >=0.7.0 <0.9.0;
```

Importar archivos

COLLEDGE
web 3.0 community

- En caso de necesitar de un archivo externo se puede referenciar al mismo con la sentencia import
- La ruta debe ser relativa al archivo del contrato actual

```
import ”../archivo.sol";
```

Comentarios

COLLEDGE
web 3.0 community

- Existen dos estilos:
  - Linea simple: //
  - Linea múltiple: /* */
- Pueden utilizarse para dejar comentarios útiles a quienes vean el código del contrato
- Una buena práctica para las auditorías es explicar la intención principal del código

Declaración del contrato

COLLEDGE
web 3.0 community

- Se marca con la palabra contract y se rodea entre { }
- Es similar a la definición de clases en POO

```
contract SimpleStorage {
        ..
}
```

Declaración de variables

COLLEDGE
web 3.0 community

- Las variables en Solidity son tipadas
- El orden es:
  - Tipo de dato
  - Visibilidad (opcional)
  - Nombre

```
uint public storedData;
```

Tipo de dato

COLLEDGE
web 3.0 community

- bool (true/false)
- int/uint
- No hay soporte para números decimales
- address
- string
- bytes

Declaración de funciones

COLLEDGE
web 3.0 community

- Las funciones se declaran con la palabra function
- Su sintaxis es diferente a otras plataformas ya que su orden es:
  - Function
  - Nombre de la función
  - Parámetros
  - Visibilidad / Accesibilidad
  - Valor de retorno

```
function helper(uint x) pure returns (uint)
{
        return x * 2;
}
```

Estructuras de control

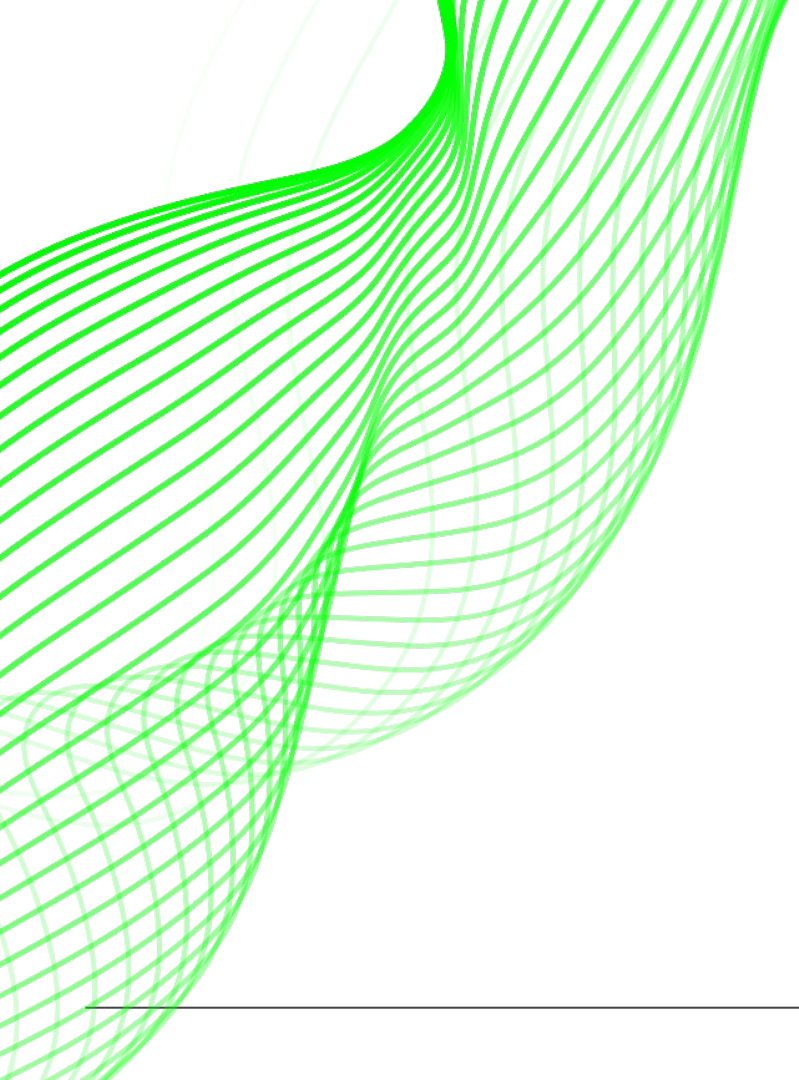
COLLEDGE
web 3.0 community

- Son las mismas utilizadas en la mayoría de los lenguajes
- If / else para sentencias condicionales
- While / for para iteraciones
- No se puede ignorar los paréntesis para las condiciones
- Pueden obviarse las { } cuando es de una sola línea
- Existe la sentencia try/catch pero sólo para funciones externas

Constructor

COLLEDGE
web 3.0 community

- Función especial que se ejecuta al crear el contrato, pero el mismo estará creado al finalizar la ejecución y no antes
- No es obligatorio definir uno
- Puede recibir parámetros aunque no es obligatorio



COLLEDGE
web 3.0 community

Ejemplos

COLLEDGE
web 3.0 community

Contacto:
Sebastián Pérez
@sebaleoperez