Développeur blockchain

Solidity

Premieres manipulation

ALYRA

cyril@alyra.fr / benjamin.brucher@alyra.fr

Promo Buterin

Topo du live

- **Vos questions!**
- **Finir les bases :**
  - Les unités
  - Valeurs par défaut
  - Variables globales
  - Les autres variables
  - Structures de contrôle
  - Exercice

ALYRA

Solidity

Unités

Ether

- 1 wei = 1
- 1 gwei = 1e9
- 1 ether = 1e18

Anciennes

- 1 szabo = 1e12
- 1 finney = 1e15

Temps

- 1 == 1 seconds
- 1 minutes == 60 seconds
- 1 hours == 60 minutes
- 1 days == 24 hours
- 1 weeks == 7 days

ALYRA

Solidity

Valeurs par défaut

Une variable qui est déclarée aura une valeur par défaut initiale dont la représentation octale est égale à une suite de zéros. Les « valeurs par défaut » des variables sont les « états zéro » typiques quel que soit le type.

Par exemple, la valeur par défaut d'un **bool** est **false**. La valeur par défaut pour les types **uint** ou **int** est **0**.

Pour les **tableaux de taille statique** et les **bytes1** à **bytes32**, chaque élément individuel sera initialisé à **la valeur par défaut correspondant à son type**. Enfin, pour **les tableaux de taille dynamique**, **les octets** et les **chaînes de caractères**, la valeur par défaut est un **tableau** ou **une chaîne vide**.

ALYRA

Pas de valeur "null" ou "undefined".

Solidity

- **boolean:** false
- **string:** ""
- **int:** 0
- **uint:** 0
- **enum:** *le premier élément de l'enum*
- **address:** 0x0000000000000000000000000000000000000000 (or address(0))
- **bytes** : 0x00
- **mapping:** mapping vide
- **struct:** une structure dont tous les membres ont une valeur par défaut
- **array**
  - **dynamically-sized**: []
  - **fixed-sized:** un tableau de taille fixe dont tous les éléments ont une valeur par défaut

ALYRA

Solidity

Variables globales

Les plus importantes:

- block.timestamp retourne un “uint” qui représente l’horodatage du bloc actuel en secondes.
- msg.sender : retourne une “address payable“ qui représente l’expéditeur du message de l’appel en cours.
- msg.value : retourne un “uint” qui représente le nombre de wei envoyés avec la transaction.

ALYRA

Solidity

Les autres variables

blockhash(uint blockNumber) returns (bytes32) : une fonction qui renvoie le hash du bloc donné - ne fonctionne que pour les 256 blocs les plus récents, à l'exclusion des blocs actuel.
block.coinbase : retourne une “address payable” qui représente l’adresse Ethereum actuelle du mineur du bloc.
block.difficulty : retourne un “uint” qui représente la difficulté actuelle du bloc.
block.gaslimit: retourne un “uint” qui représente le gaslimit du bloc actuel.
block.number: retourne un “uint” qui représente le numéro du bloc actuel.
now : retourne un “uint” qui représente l'horodatage actuel du bloc (alias block.timestamp).
gasleft() returns (uint256) : une fonction qui renvoie un “uint” qui représente le gas restant.
msg.data : retourne des “bytes” qui représentent les données de l’appel.
msg.sig : retourne un “bytes4” qui représente les quatre premiers octets du calldata (c'est-à-dire l'identificateur de fonction).
tx.gasprice : retourne un “uint” qui représente le prix du gas de la transaction.
tx.origin : retourne une “address payable” qui représente l’expéditeur de la transaction.

ALYRA

Solidity

Structures de contrôle

La plupart des structures de contrôle connues sont disponibles dans Solidity :

- if, else, while, do, for, break, continue, return …

Solidity prend également en charge la gestion des exceptions sous la forme du try/catch, mais uniquement pour les appels de fonctions externes et les appels de création de contrats.

this fait référence à l'instance de ce contrat déployé
super fait référence au contrat parent direct

ALYRA

Solidity

exercice en incrément

Partie 1:
Faire la base d'un contrat: license, pragma, nom

ALYRA

Solidity

exercice en incrément

Partie 1:
Faire la base d'un contrat: license, pragma, nom

Partie 2:
Avoir une variable d'état de type address
Faire une fonction qui vient set cette variable

ALYRA

Solidity

exercice en incrément

Partie 1:
Faire la base d'un contrat: license, pragma, nom

Partie 2:
Avoir une variable d'état de type address
Faire une fonction qui vient set cette variable

Partie 3:
Faire une fonction qui récupère et renvoie la
de l'address stockée

ALYRA

Solidity

exercice en incrément

Partie 1:
Faire la base d'un contrat: license, pragma, nom

Partie 2:
Avoir une variable d'état de type address
Faire une fonction qui vient set cette variable

Partie 3:
Faire une fonction qui récupère et renvoie la
de l'address stockée

Partie 4:
Faire une fonction qui récupère et renvoie la balance
d'une address passée en paramètre

ALYRA

Solidity

exercice en incrément

Partie 5:
Faire une fonction qui permet de faire un transfert d'eth vers une address passée en paramètre

ALYRA

Solidity

exercice en incrément

Partie 5:
Faire une fonction qui permet de faire un transfert d'eth vers une address passée en paramètre

Partie 6:
Faire une fonction qui permet de faire un transfert d'eth vers l'address stockée, si et uniquement si elle a une balance superieur à une valeur donnée en param
Vous pouvez tester avec 1 wei ou 100 eth

ALYRA

Solidity

exercice en incrément

Partie 5:
Faire une fonction qui permet de faire un transfert d'eth vers une address passée en paramètre

Partie 6:
Faire une fonction qui permet de faire un transfert d'eth vers l'address stockée, si et uniquement si elle a une balance superieur à une valeur donnée en param
Vous pouvez tester avec 1 wei ou 100 eth

Partie 7:
Ajouter une vérification aux fonctions d'envoi: le minimum envoyé est de 1wei

ALYRA

Untitled-1

```solidity
// SPDX-License-Identifier: MIT
pragma solidity 0.8.20;

error InsufficientBalance();

contract SendEther {

    address myAddress;

    function setMyAddress(address _myAddress) external {
        myAddress = _myAddress;
    }

    function getBalance() external view returns(uint) {
        return myAddress.balance;
    }

    function getBalanceOfAddress(address _myAddress) external view returns(uint) {
        return _myAddress.balance;
    }

    function sendViaTransfer(address payable _to) external payable {
        // transfer n'est plus recommandé
        _to.transfer(msg.value);
    }

    function sendViaSend(address payable _to) external payable {
        // send n'est pas recommandé
        bool sent = _to.send(msg.value);
        require(sent, "Failed to send Ether");
    }

    function sendViaCall(address payable _to) external payable {
        // C'est la méthode qu'il faut utiliser selon la documentation
        (bool sent, ) = _to.call{value: msg.value}("");
        require(sent, "Failed to send Ether");
    }

    // Faire une fonction qui permet de faire un transfert d'eth vers l'address stockée, si et uniquement si elle a une balance superieur à une valeur donnée en param. Vous pouvez tester avec 1 wei ou 100 eth */
    function sendIfEnoughEthers(uint _minBalance) external payable {
        if(myAddress.balance <= _minBalance) {
            revert InsufficientBalance();
        }
        (bool sent, ) = myAddress.call{value: msg.value}("");
        require(sent, "Failed to send Ether");
    }

    // Ajouter une vérification aux fonctions d'envoi: le minimum envoyé est de 1wei */
    // require(msg.value >= 1, "Not enough wei sent");
}
```

07/02/2023

Merci de votre attention!

ALYRA