

The background features abstract geometric shapes in shades of green and blue. A red plus sign is located to the left of the title. Green 'x' marks are scattered in the top-left and bottom-right corners. A red circle is in the bottom-right corner.

Technical Documentation Developer Guide Ethereum Blockchain

By Fajri Nurfauzan | 1103180184

TABLE OF CONTENTS

LAB : 01

Deposit/Withdraw Ether

You can describe the topic of
the section here

LAB : 02

Shared Wallet

You can describe the topic of
the section here

LAB : 03

Supply Chain Project

You can describe the topic of
the section here

Untuk Memenuhi Tugas UTS

Power Point ini merupakan kesatuan dari keseluruhan technical documentation dari semua section LAB





LAB : 01

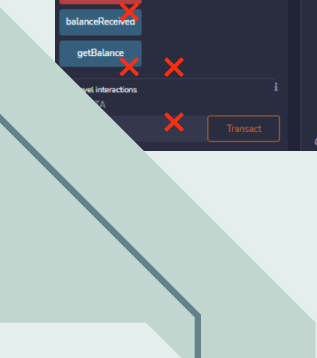
Deposit/Withdraw Ether

Disini akan mempelajari cara sebuah Kontrak Cerdas (Smart Contract) mengelola dananya sendiri.

SendMoney.sol

```
SendMoneyExample.sol X
1 // SPDX-License-Identifier: GPL-3.0
2 // Fajri Nurfauzan || 1103180184
3
4
5 pragma solidity ^0.8.1;
6
7 contract SendMoneyExample {
8
9     uint public balanceReceived;
10
11     function receiveMoney() public payable {
12         balanceReceived += msg.value;
13     }
14
15     function getBalance() public view returns(uint) {
16         return address(this).balance;
17     }
18 }
```

- Buat file remix sederhana seperti berikut dan beri nama SendMoneyExample.sol
- uint public balanceReceived : adalah variabel penyimpanan publik. Variabel publik akan membuat fungsi pengambil secara otomatis di Kepadatan.
- balanceReceived += msg.value : Objek msg adalah objek global yang selalu ada yang berisi beberapa informasi tentang transaksi yang sedang berlangsung
- function getBalance() public view returns(uint) : fungsi tampilan adalah fungsi yang tidak mengubah penyimpanan (Read-only) dan dapat mengembalikan informasi
- address(this).balance : Variabel bertipe address selalu memiliki properti bernama .balance yang memberi Anda jumlah eter disimpan di alamat itu.



- Setelah memahami fungsi fungsi maka hal yang perlu dilakukan adalah mendeploynya dengan cara Buka Plugin Deploy and Run Transactions dan terapkan Smart Contract ke dalam JavaScript VM :
- Setelah mengklik deploy akan muncul sebuah output pada deployed contracts
- Dan di deployed contract terlihat ada pilihan receiveMoney, balance, Received, dan getBalance

Cek value dan transaksi

The image shows the Remix IDE interface. On the left, the 'CONTRACT' tab is active, displaying the 'SendMoneyExample - SendMoneyExample.sol' contract. The 'GAS LIMIT' is set to 3000000, and the 'VALUE' is set to 1 Ether. The 'Deploy' button is highlighted. Below it, there is a checkbox for 'Publish to IPFS'. On the right, the 'Deployed Contracts' tab is active, showing the 'SENDMONEYEXAMPLE AT 0XD91...39138' contract. The 'receiveMoney' button is highlighted in red. Below it, the 'balanceReceived' button is highlighted in blue, showing a value of 0: uint256: 10000000000000000000. The 'getBalance' button is also highlighted in blue, showing a value of 0: uint256: 1000000000000000000000. At the bottom, the 'Low level interactions' section shows the 'CALLDATA' field and a 'Transact' button.

GAS LIMIT
3000000

VALUE
1 Ether

CONTRACT
SendMoneyExample - SendMoneyExample.sol

Deploy

☐ Publish to IPFS

OR

SENDMONEYEXAMPLE

receiveMoney

balanceReceived

0: uint256: 1000000000000000000000

getBalance

0: uint256: 1000000000000000000000

Low level interactions

CALLDATA

Transact

- Setelah itu Isi Value dengan 1 dan ubah mata uangnya menjadi Ether jika sudah langsung klik receive money pada deployed contract
- Setelah mengklik deploy akan muncul sebuah output pada deployed contracts
- Nantinya akan terjadi transaksi baru dan bisa dicek dengan klik balanceReceived dan getBalance seharusnya memiliki nilai yang sama



- Langkah selanjutnya kita coba menambahkan fungsi `withdrawMoney()` dibawah fungsi `getbalance()`:
- Jika sudah lakukan deploy Kembali dan lihat output di Deployed Contract dan lakukan Langkah sebelumnya
- Dan cek Kembali balancenya apakah ada perubahan atau tidak, jika berubah berarti ada yang salah dalam penulisan code jika tidak lanjut kelangkah selanjutnya
- Jika saldo 0, maka periksa kembali kolom nilai
Jika saldo adalah 2 Eter, periksa kembali Instance kontrak yang berinteraksi!



Dan BOOOM lihat hasilnya saldo ether pada account ke dua bertambah



Withdraw to Specific Account

```
SendMoneyAddWithDrawSpecificAcc.sol
1 // SPDX-License-Identifier: GPL-3.0
2 // Fajri Nurfauzan || 1103180184
3
4
5 pragma solidity ^0.8.1;
6
7 contract SendMoneyExample {
8
9     uint public balanceReceived;
10
11     function receiveMoney() public payable {
12         balanceReceived += msg.value;
13     }
14
15     function getBalance() public view returns(uint) {
16         return address(this).balance;
17     }
18
19     function withdrawMoney() public {
20         address payable to = payable(msg.sender);
21         to.transfer(getBalance());
22     }
23
24     function withdrawMoneyTo(address payable _to) public {
25         _to.transfer(getBalance());
26     }
27 }
```



Selanjutnya disini akan mencoba untuk mengirim dana ke rekening atau account tertentu dengan menambahkan fungsi berikut :



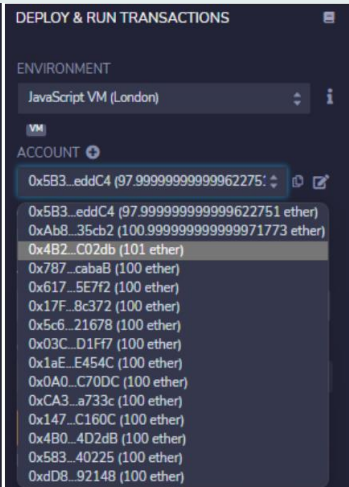
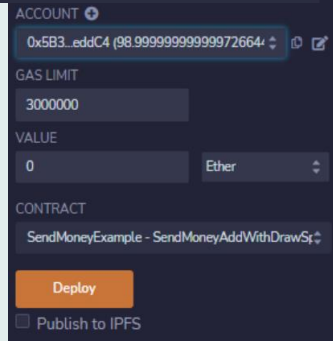
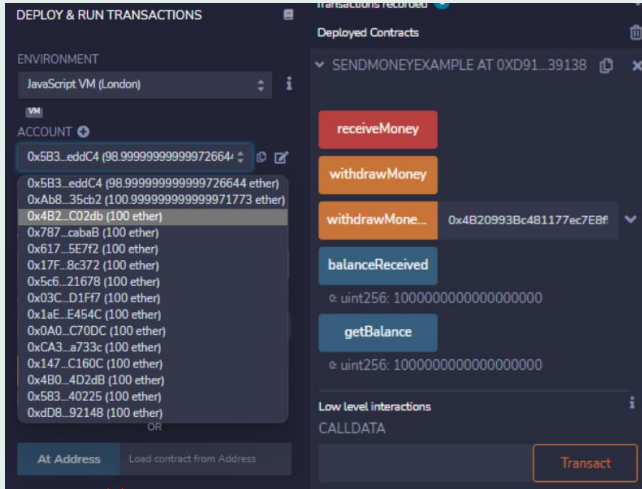
Jika sudah bisa lakukan next step dislide berikutnya

Withdraw to Specific Account

Dan pilih account ke 3 lalu copy Nomor account tersebut setelah itu simpan di "withdrawMoneyTo" dan kembalikan account ke account 1:

Lakukan deploy kembali dan send 1 ether pada bagian input withdrawmonet to

Dan Boom langsung mendapatkan 1 ether penuh





LAB : 02

Shared Wallet

Disini akan mempelajari cara sebuah Wallet yang dapat menyimpan dana dan memungkinkan penggunaanya untuk menarik kembali dananya.



SharedWallet.sol

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 contract SharedWallet {
6     function withdrawMoney(address payable _to, uint _amount) public {
7         _to.transfer(_amount);
8     }
9
10    receive() external payable {
11
12    }
13 }
```

● Buat file remix sederhana seperti berikut dan beri nama SharedWallet.sol

● Ini adalah kontrak pintar (Smart Contract) yang sangat mendasar yang dapat menerima Eter dan dimungkinkan untuk menarik Eter, tetapi secara keseluruhan, sebenarnya ini tidak terlalu berguna atau cukup belum digunakan secara kasus

✗ ✗

✗

Permission Shared Wallet

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 contract SharedWallet {
6
7     address owner;
8
9     constructor() {
10         owner = msg.sender;
11     }
12
13     modifier onlyOwner() {
14         require(msg.sender == owner, "You are not allowed");
15         _;
16     }
17
18     function withdrawMoney(address payable _to, uint _amount) public onlyOwner {
19         _to.transfer(_amount);
20     }
21
22     receive() external payable {
23
24     }
25 }
```

Pada langkah ini dapat dilihat code akan membatasi penarikan kepada pemilik dompet. Bagaimana cara menentukan pemiliknya? Itu adalah dengan pengguna yang menyebarkan kontrak pintar (Smart Contract). Ketahuilah bahwa code yang sudah diketik perlu menambahkan pengubah "onlyOwner" ke fungsi withdrawMoney!

OpenZeppelin Shared Wallet

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

pragma solidity 0.8.1;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";

contract SharedWallet {

    function isOwner() internal view returns(bool) {
        return owner() == msg.sender;
    }

    function withdrawMoney(address payable _to, uint _amount) public {
        _to.transfer(_amount);
    }

    receive() external payable {
    }
}
```

Untuk memiliki logika seperti pemilik secara langsung dalam satu kontrak pintar (Smart Contract) merupakan hal yang tidak mudah untuk diaudit. Maka dari itu perlu diuraikan menjadi bagian-bagian yang lebih kecil dan akan digunakan kembali kontrak pintar (Smart Contract) yang sudah diaudit dari OpenZeppelin untuk itu. Kontrak OpenZeppelin terbaru tidak memiliki isOwner() fungsi lagi, jadi kita harus membuat sendiri. Perhatikan bahwa pemilik() adalah fungsi dari kontrak Ownable.sol

OpenZeppelin Shared Wallet

```
1 |//SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
6
7 contract SharedWallet is Ownable {
8
9     function isOwner() internal view returns(bool) {
10         return owner() == msg.sender;
11     }
12
13     mapping(address => uint) public allowance;
14
15     function addAllowance(address _who, uint _amount) public onlyOwner {
16         allowance[_who] = _amount;
17     }
18
19     modifier ownerOrAllowed(uint _amount) {
20         require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
21         _;
22     }
23
24     function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
25         require(_amount <= address(this).balance, "Contract doesn't own enough money");
26         to: transfer _to.transfer(_amount);
27     }
28
29     receive() external payable {
30     }
31 }
```

Pada langkah ini perlu menambahkan pemetaan sehingga code dapat menyimpan alamat => jumlah uint. Code ini akan seperti array yang menyimpan [0x123546...] alamat, ke nomor tertentu. Jadi, code selalu tahu berapa banyak yang bisa ditarik seseorang. dan juga menambahkan yang baru modifikator yang akan memeriksa: Apakah pemiliknya sendiri atau hanya seseorang dengan uang saku?

Memisahkan Shared Wallet



```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
6
7 contract Allowance is Ownable {
8
9     function isOwner() internal view returns(bool) {
10         return owner() == msg.sender;
11     }
12
13     mapping(address => uint) public allowance;
14
15     function setAllowance(address _who, uint _amount) public onlyOwner {
16         allowance[_who] = _amount;
17     }
18
19     modifier ownerOrAllowed(uint _amount) {
20         require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
21         _;
22     }
23
24     function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
25         allowance[_who] -= _amount;
26     }
27
28 }
29
30 contract SharedWallet is Allowance {
31     function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
32         require(_amount <= address(this).balance, "Contract doesn't own enough money");
33         if(!isOwner()) {
34             reduceAllowance(msg.sender, _amount);
35         }
36         _to.transfer(_amount);
37     }
38
39     receive() external payable {
40     }
41 }
```



Sekarang setelah mengetahui fungsi dasar hal selanjutnya dapat menyusun kontrak pintar (Smart Contract) secara berbeda. Untuk membuatnya lebih mudah dibaca, mungkin dengan bisa mengistirahatkan fungsionalitas menjadi dua kontrak pintar (Smart Contract) yang berbeda

✗

✗

✗

✗

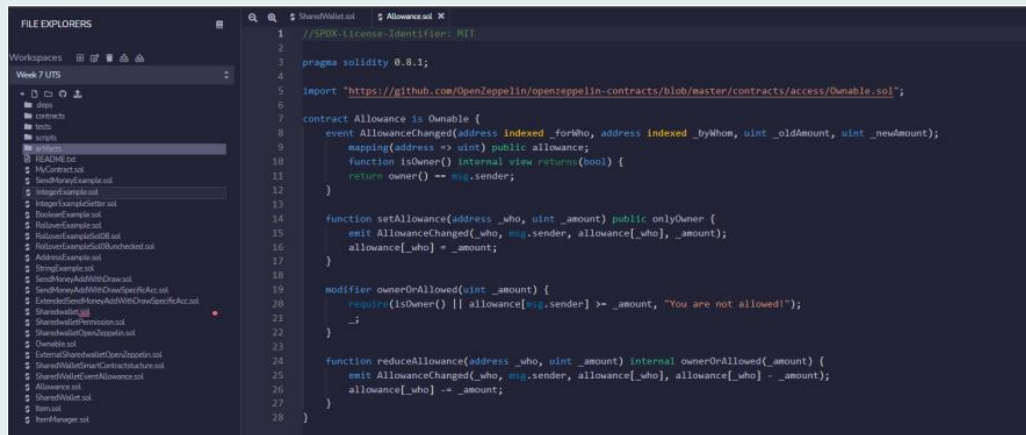
Allowance Shared Wallet

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
6
7 contract Allowance is Ownable {
8
9     event AllowanceChanged(address indexed _forwho, address indexed _bywhom, uint _oldAmount, uint _newAmount);
10
11     mapping(address => uint) public allowance;
12
13     function isOwner() internal view returns(bool) {
14         return owner() == msg.sender;
15     }
16
17     function setAllowance(address _who, uint _amount) public onlyOwner {
18         emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
19         allowance[_who] = _amount;
20     }
21
22     modifier ownerOrAllowed(uint _amount) {
23         require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
24         _;
25     }
26
27     function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
28         emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
29         allowance[_who] -= _amount;
30     }
31 }
32
33 contract SharedWallet is Allowance {
34
35     event MoneySent(address indexed _beneficiary, uint _amount);
36     event MoneyReceived(address indexed _from, uint _amount);
37
38     function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
39         require(_amount <= address(this).balance, "Contract doesn't own enough money");
40         if(!isOwner()) {
41             reduceAllowance(msg.sender, _amount);
42         }
43         emit MoneySent(_to, _amount);
44         _to.transfer(_amount);
45     }
46
47     receive() external payable {
48         emit MoneyReceived(msg.sender, msg.value);
49     }
50 }
```

Dan terakhir ini merupakan fungsi penambahan event Allowance dan SharedWallet setelah mengetahui dasar bisa mencoba untuk membuat Smart Contract dengan folder terpisah dan gunakan fungsi impor dan akan menjadi seperti ini

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import "../Allowance.sol";
6
7 contract SharedWallet is Allowance {
8
9     event MoneySent(address indexed _beneficiary, uint _amount);
10     event MoneyReceived(address indexed _from, uint _amount);
11
12     function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
13         require(_amount <= address(this).balance, "Contract doesn't own enough money");
14         if(!isOwner()) {
15             reduceAllowance(msg.sender, _amount);
16         }
17         emit MoneySent(_to, _amount);
18         _to.transfer(_amount);
19     }
20
21     function renounceOwnership() public override onlyOwner {
22         revert("can't renounceOwnership here"); //not possible with this smart contract
23     }
24
25     receive() external payable {
26         emit MoneyReceived(msg.sender, msg.value);
27     }
28 }
```

Allowance Shared Wallet

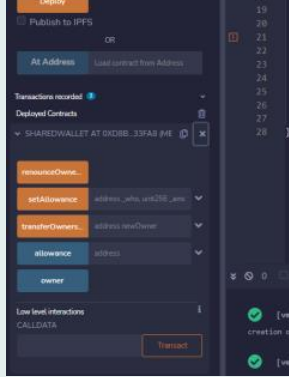
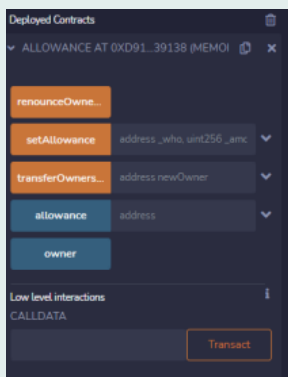



```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
6
7 contract Allowance is Ownable {
8     event AllowanceChanged(address indexed _forWho, address indexed _byWhom, uint _oldAmount, uint _newAmount);
9     mapping(address => uint) public allowance;
10     function isOwner() internal view returns (bool) {
11         return owner() == msg.sender;
12     }
13
14     function setAllowance(address _who, uint _amount) public onlyOwner {
15         emit AllowanceChanged(_who, msg.sender, allowance[_who], _amount);
16         allowance[_who] = _amount;
17     }
18
19     modifier ownerOrAllowed(uint _amount) {
20         require(isOwner() || allowance[msg.sender] >= _amount, "You are not allowed!");
21         _;
22     }
23
24     function reduceAllowance(address _who, uint _amount) internal ownerOrAllowed(_amount) {
25         emit AllowanceChanged(_who, msg.sender, allowance[_who], allowance[_who] - _amount);
26         allowance[_who] -= _amount;
27     }
28 }
```

Jika sudah membuat dua folder terpisah lalu deploy dan hasilnya akan seperti ini



```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity 0.8.1;
4
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol";
6
7 contract SharedWallet is Allowance {
8
9     event MoneySent(address indexed _beneficiary, uint _amount);
10     event MoneyReceived(address indexed _from, uint _amount);
11
12     function withdrawMoney(address payable _to, uint _amount) public ownerOrAllowed(_amount) {
13         require(_amount <= address(this).balance, "Contract doesn't own enough money");
14         if (isOwner()) {
15             reduceAllowance(msg.sender, _amount);
16         }
17         emit MoneySent(_to, _amount);
18         _to.transfer(_amount);
19     }
20
21     function renounceOwnership() public override onlyOwner {
22         revert("can't renounceOwnership here"); //not possible with this smart contract
23     }
24
25     receive() external payable {
26         emit MoneyReceived(msg.sender, msg.value);
27     }
28 }
```





LAB : 03

Supply Chain Project

akan mempelajari cara sebuah Manager yang dapat menambah, membayar, dan memindahkan sebuah item atau wallet ke dalam Supply Chain yang akan memicu pengiriman melewati solusi rantai

ItemManager.sol

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "./Item.sol";
6
7 contract ItemManager {
8
9     struct S_Item {
10         Item _item;
11         ItemManager.SupplyChainSteps _step;
12         string _identifier;
13     }
14
15     mapping(uint => S_Item) public items;
16     uint index;
17
18     enum SupplyChainSteps {Created, Paid, Delivered}
19     event SupplyChainStep(uint _itemIndex, uint _step, address _address);
20
21     function createItem(string memory _identifier, uint _priceInWei) public {
22         Item item = new Item(this, _priceInWei, index);
23         items[index]._item = item;
24         items[index]._step = SupplyChainSteps.Created;
25         items[index]._identifier = _identifier;
26         emit SupplyChainStep(index, uint(items[index]._step), address(item));
27         index++;
28     }
29
30     function triggerPayment(uint _index) public payable {
31         Item item = items[_index]._item;
32         require(address(item) == msg.sender, "Only items are allowed to update themselves");
33         require(item.priceInWei() == msg.value, "Not fully paid yet");
34         require(items[_index]._step == SupplyChainSteps.Created, "Item is further in the supply chain");
35         items[_index]._step = SupplyChainSteps.Paid;
36         emit SupplyChainStep(_index, uint(items[_index]._step), address(item));
37     }
38
39     function triggerDelivery(uint _index) public {
40         require(items[_index]._step == SupplyChainSteps.Paid, "Item is further in the supply chain");
41         items[_index]._step = SupplyChainSteps.Delivered;
42         emit SupplyChainStep(_index, uint(items[_index]._step), address(items[_index]._item));
43     }
44 }
```

- Buat file remix sederhana seperti berikut dan beri nama ItemManager.sol
- setelah menulis code diatas lakukan penulisan kembali untuk menulis code berikut dan beri nama dengan "Item.sol"

Item.sol


```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "../ItemManager.sol";
6
7 contract Item {
8     uint public priceInWei;
9     uint public paidWei;
10    uint public index;
11
12    ItemManager parentContract;
13    constructor(ItemManager _parentContract, uint _priceInWei, uint _index) public {
14        priceInWei = _priceInWei;
15        index = _index;
16        parentContract = _parentContract;
17    }
18
19    receive() external payable {
20        require(msg.value == priceInWei, "We don't support partial payments");
21        require(paidWei == 0, "Item is already paid!");
22        paidWei += msg.value;
23        (bool success, ) = address(parentContract).call{value:msg.value}(abi.encodeWithSignature("triggerPayment(uint256)", index));
24        require(success, "Delivery did not work");
25    }
26
27    fallback () external {
28    }
29
30 }
```



Sekarang dengan ini, hanya perlu memberi nama pelanggan dengan alamat Kontrak Cerdas (Smart Contract) Item yang dibuat dengan nama "createItem" dan dia akan dapat membayar langsung dengan mengirimkan X Wei ke Kontrak Cerdas (Smart Kontrak). Tetapi kontrak pintar (Smart Kontrak) yang sudah dibuat belum terlalu aman.



Ownable.sol



```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.6.0;
4
5 import "./Ownable.sol";
6 import "./Item.sol";
7
8 contract ItemManager is Ownable {
9
10     //...
11
12     function createItem(string memory _identifier, uint _priceInWei) public onlyOwner {
13         //...
14     }
15
16     function triggerPayment(uint _index) public payable {
17         //...
18     }
19
20     function triggerDelivery(uint _index) public onlyOwner {
21         //...
22     }
23 }
```

Maka langkah selanjutnya butuh semacam fungsi pemilik, dan biasanya code akan ditambahkan Kontrak Cerdas (Smart Kontrak) OpenZeppelin dengan Fungsionalitas yang Dapat Dimiliki. Tetapi dikarenakan dokumen mereka belum diperbarui ke solidity 0.6 maka fungsi Ownable dibuat sendiri yang sangat mirip dengan satu dari OpenZeppelin dan beri nama file tersebut dengan "Ownable.sol" lalu Dengan ubah ItemManager sehingga semua fungsi, yang seharusnya dapat dieksekusi oleh "pemilik saja" memiliki pengubah yang benar :

truffle

```
PS C:\Users\Fajri> npm install -g truffle
npm WARN deprecated mkdirp-promise@5.0.1: This package is broken and no longer maintained. 'mkdirp' itself supports promises now, please switch to that.
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated ipfs-http-client@0.10.0: This module has been superseded by the multiformats module
npm WARN deprecated circular-json@0.5.9: CircularJSON is in maintenance only, flattened is its successor.
npm WARN deprecated cids@1.1.9: This module has been superseded by the multiformats module
npm WARN deprecated ipfs-dag-cbor@0.17.1: This module has been superseded by @ipld/dag-cbor and multiformats
npm WARN deprecated multicodec@1.0.4: This module has been superseded by the multiformats module
npm WARN deprecated @nodefactory/filspn-adaptec@0.2.2: Package is deprecated in favour of @chainsafe/filspn-adaptec
```

● Setelah menulis semua code berikut lakukan install Truffle dengan mebuca terminal (Mac/Linux) atau PowerShell (Windows 10/11) Setelah menulis semua code berikut lakukan install Truffle dengan mebuca terminal (Mac/Linux) atau PowerShell (Windows 10/11)

● Jika sudah bisa cek truffle version dengan cara menulis code berikut

```
PS C:\Users\Fajri> truffle version
Truffle v5.4.0 (core: 5.4.0)
Solidity v0.5.16 (solc-js)
Node v16.13.0
Web3.js v1.4.0
```

×

×

×

truffle

Jika sudah silahkan membuat Kemudian buat folder kosong, dengan nama "s06-eventtrigger" dan jika sudah bisa menggunakan fungsi cd s06-eventtrigger untuk berpindah folder

```
ebd> mkdir s06-eventtrigger
```

```
ebd> cd .\s06-eventtrigger\  
s06-eventtrigger> ls_  
s06-eventtrigger>
```

Dan mengetikkan ls untuk melihat isi folder

Jika sudah berpindah folder ketikan truffle unbox react

```
x  
x x  
x  
s06-eventtrigger> truffle unbox react  
✓ Preparing to download box  
✓ Downloading  
✓ cleaning up temporary files  
✓ Setting up box  
s06-eventtrigger> ls
```

truffle



Lalu ls dan akan keluar seperti ini

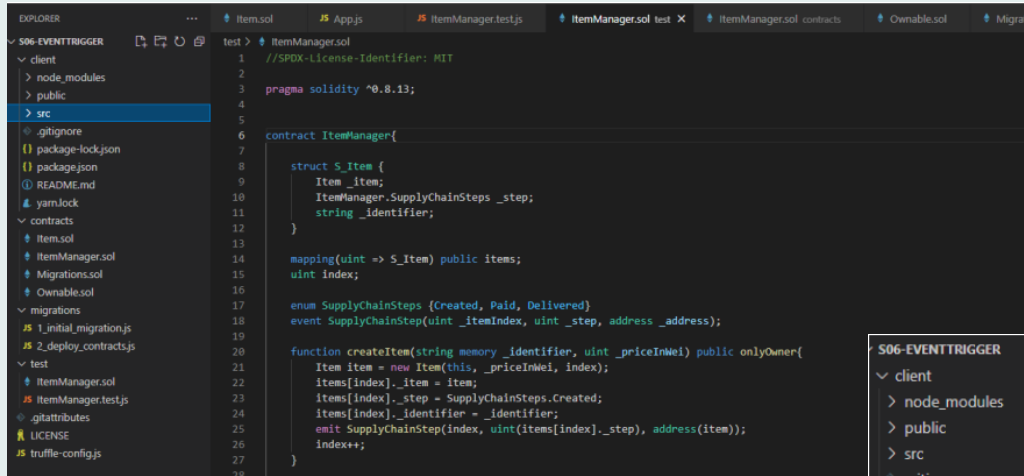
```
PS C:\Users\Fajri\s06-eventtrigger> ls
```

```
Directory: C:\Users\Fajri\s06-eventtrigger
```

Mode		LastWriteTime	Length	Name
----		-----	-----	----
d-----		4/21/2022 8:15 PM		client
d-----		4/21/2022 8:20 PM		contracts
d-----		4/21/2022 8:15 PM		migrations
d-----		4/21/2022 8:15 PM		test
-a----		4/21/2022 8:15 PM	33	.gitattributes
-a----		4/21/2022 8:15 PM	1075	LICENSE
-a----		4/21/2022 8:28 PM	357	truffle-config.js

x
x x
x

Modify Code in folder

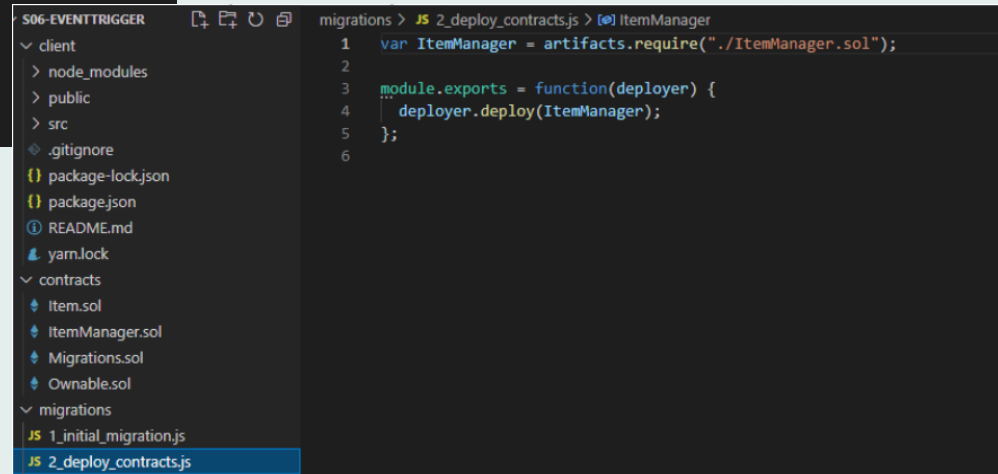


The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing the file structure of a project. The 'src' folder is selected. In the main editor, the 'ItemManager.sol' file is open, displaying Solidity code. The code includes a pragma statement for Solidity version 0.8.13, a contract definition for 'ItemManager', a struct for 'S_Item', a mapping for items, an enum for 'SupplyChainSteps', an event for 'SupplyChainStep', and a function 'createItem'.

```
1 //SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.13;
4
5
6 contract ItemManager{
7
8     struct S_Item {
9         Item _item;
10        ItemManager.SupplyChainSteps _step;
11        string _identifier;
12    }
13
14    mapping(uint => S_Item) public items;
15    uint index;
16
17    enum SupplyChainSteps {Created, Paid, Delivered}
18    event SupplyChainStep(uint _itemIndex, uint _step, address _address);
19
20    function createItem(string memory _identifier, uint _priceInWei) public onlyOwner{
21        Item item = new Item(this, _priceInWei, index);
22        items[index]._item = item;
23        items[index]._step = SupplyChainSteps.Created;
24        items[index]._identifier = _identifier;
25        emit SupplyChainStep(index, uint(items[index]._step), address(item));
26        index++;
27    }
28 }
```

Jika sudah buka folder s06-eventtrigger tersebut dan cek dibagian contract jika ada file "Migrations.sol" silahkan hapus dan replace dengan file yang sudah dibuat sebelumnya dan akan menampilkan seperti ini

Kemudian ubah file "migrasi" di folder migrasi/:



The screenshot shows the VS Code interface. On the left, the Explorer sidebar is open, showing the file structure of a project. The 'migrations' folder is selected. In the main editor, the '2_deploy_contracts.js' file is open, displaying JavaScript code. The code includes a require statement for 'ItemManager.sol' and a function 'deploy' that uses 'deployer.deploy' to deploy the 'ItemManager' contract.

```
1 var ItemManager = artifacts.require("./ItemManager.sol");
2
3 module.exports = function(deployer) {
4     deployer.deploy(ItemManager);
5 };
6
```



Modify truffle-config.js



Ubah file truffle-config.js untuk mengunci versi kompiler tertentu:

```
JS truffle-config.js > [?] <unknown>
1  const path = require("path");
2
3  module.exports = {
4    // See <http://truffleframework.com/docs/advanced/configuration>
5    // to customize your Truffle configuration!
6    contracts_build_directory: path.join(__dirname, "client/src/contracts"),
7    networks: {
8      develop: {
9        port: 8545
10      }
11    },
12    compilers: {
13      solc: {
14        version: "^0.8.0"
15      }
16    }
17  };
18
```

×

×

×

Migrate

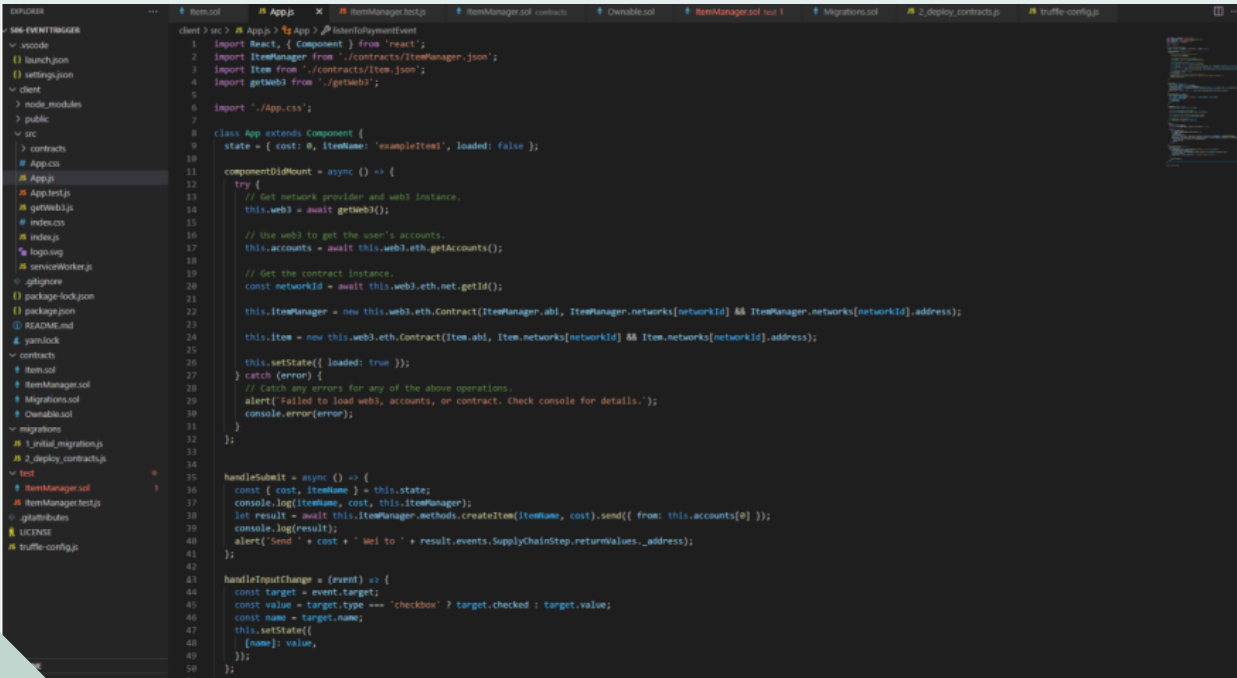
Jika sudah Kembali ke power shell lalu ketikan truffle develop dan ketikan migrate dan akan menampilkan seperti ini :

```
truffle(develop)> migrate

Compiling your contracts...
=====
> Compiling .\contracts\Item.sol
> Compiling .\contracts\ItemManager.sol
> Compiling .\contracts\Ownable.sol
> Artifacts written to C:\Users\Fajri\s06-eventtrigger\client\src\contracts
x Compiled successfully using:
x - solc: 0.8.13+commit.abaa5c0e.Emscripten.clang
x Network up to date.
  Fetching solc version list from solc-bin. Attempt #1
  Fetching solc version list from solc-bin. Attempt #1
```


Modify HTML

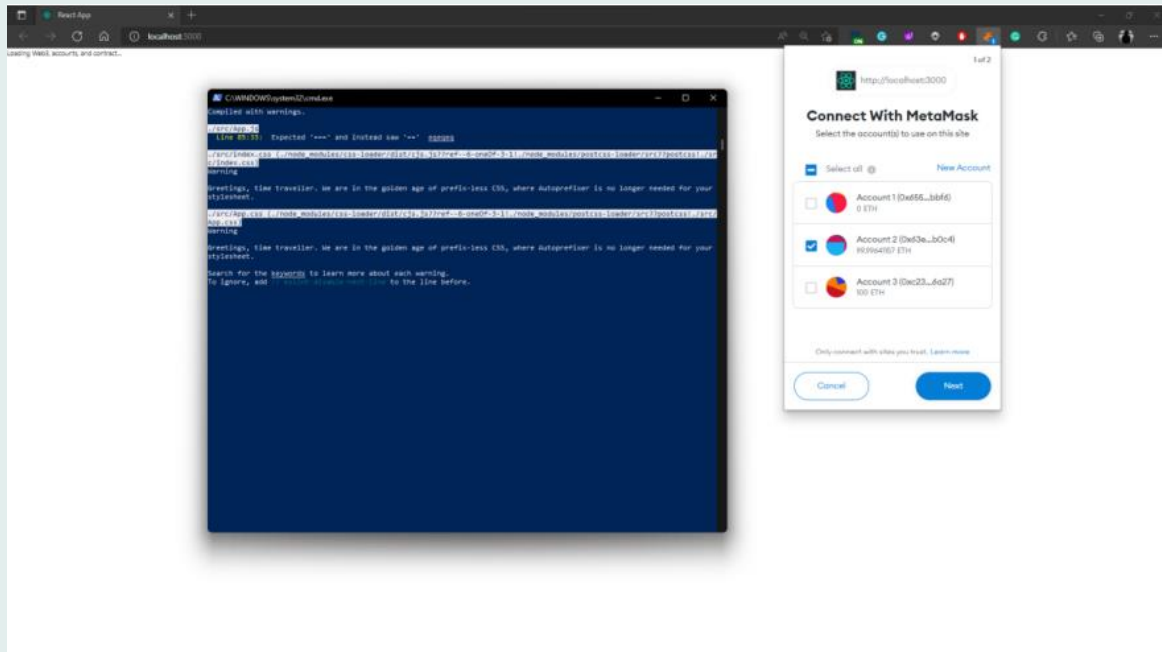
Jika sudah modif HTML menjadi seperti ini



```
client > cd > # App.js > # ItemManager.test.js
1 import React, { Component } from 'react';
2 import ItemManager from '../contracts/ItemManager.json';
3 import Item from '../contracts/Item.json';
4 import getWeb3 from './getWeb3';
5
6 import './App.css';
7
8 class App extends Component {
9   state = { cost: 0, itemName: 'exampleItem', loaded: false };
10
11   componentDidMount = async () => {
12     try {
13       // Get network provider and web3 instance.
14       this.web3 = await getWeb3();
15
16       // Use web3 to get the user's accounts.
17       this.accounts = await this.web3.eth.getAccounts();
18
19       // Get the contract instance.
20       const networkId = await this.web3.eth.net.getId();
21
22       this.itemManager = new this.web3.eth.Contract(ItemManager.abi, ItemManager.networks[networkId] && ItemManager.networks[networkId].address);
23
24       this.item = new this.web3.eth.Contract(Item.abi, Item.networks[networkId] && Item.networks[networkId].address);
25
26       this.setState({ loaded: true });
27     } catch (error) {
28       // Catch any errors for any of the above operations.
29       alert('Failed to load web3, accounts, or contract. Check console for details.');
```

Npm start

 Lalu buka baru power shell dan masuk ke folder client dan ketikan npm start



Npm start



Sehingga akan muncul seperti ini dan akan menampilkan

React App

localhost:3000

Simply Payment/Supply Chain Example!

Items

Add Element

Cost: Item Name: Create new Item



Npm start



Lalu masukan cost dan create item

Simply Payment/Supply Chain Example!

Items

Add Element

Cost: Item Name: Create new Item

MetaMask Notification

Localhost 8545

Account 2 → 0xDAA...3366

New address detected! Click here to add to your address book.

DETAILS DATA HEX

Estimated gas fee

0.00141726

0.001417 ETH

Site suggested

Max fee: 0.00141726 ETH

Total

0.00141726

0.00141726 ETH

Amount + gas fee

Max amount: 0.00141726 ETH

Reject

Confirm

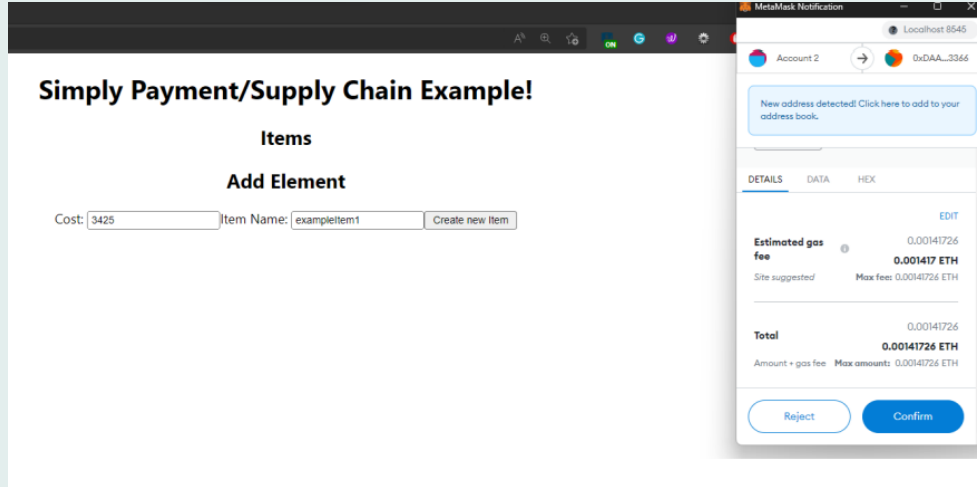
×

×

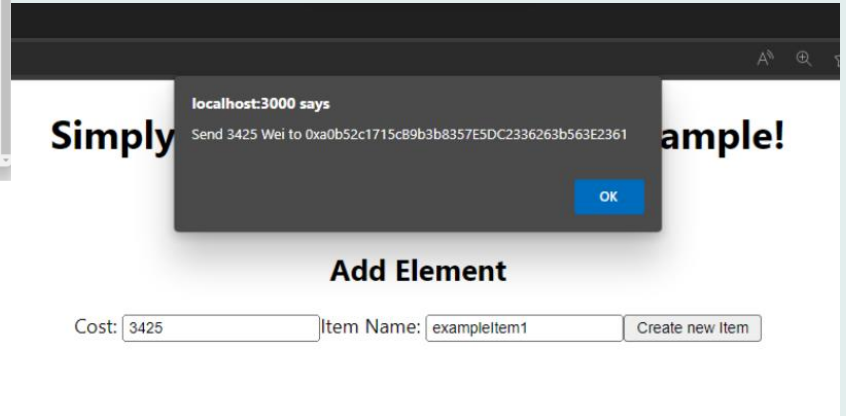
×

×

Npm start



● Lalu masukan cost dan create item



Metamask Inport Account

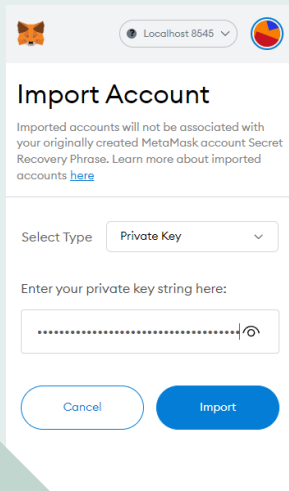
Accounts:

```
(0) 0x63eba8ba730d4ddd74cb7337d6a294f45622b0c4  
(1) 0xc23943bc1599d448605f1d8d4704fc09f9206a27
```

Private Keys:

```
(0) 54c10b3bbb3b9a0692d2b47903b2d593f2ab26f75e414ffc0c35b9a5d19cfa9a  
(1) a7170ebdf82e0f284ff98fc0988b0d9b4f0ac83717f524097778457cf3f57f52
```

- Selain itu bisa mengimport account dengan menuliskan truffle develop dan akan muncul acc dan private key
- Setelah itu tinggal memasukan priveate key di import account
- Dan account akan otomatis terbuat



Import Account

Imported accounts will not be associated with your originally created MetaMask account Secret Recovery Phrase. Learn more about imported accounts [here](#)

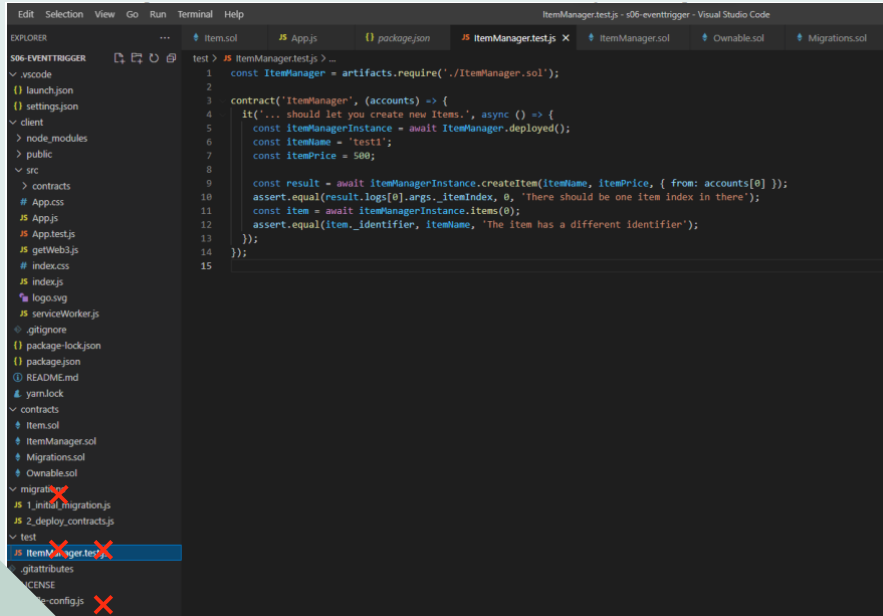
Select Type: Private Key

Enter your private key string here:

.....

Cancel Import

Test Unit



```
1 const ItemManager = artifacts.require('./ItemManager.sol');
2
3 contract('ItemManager', (accounts) => {
4   it('... should let you create new Items.', async () => {
5     const itemManagerInstance = await ItemManager.deployed();
6     const itemName = 'test1';
7     const itemPrice = 500;
8
9     const result = await itemManagerInstance.createItem(itemName, itemPrice, { from: accounts[0] });
10    assert.equal(result.logs[0].args._itemIndex, 0, 'There should be one item index in there');
11    const item = await itemManagerInstance.items(0);
12    assert.equal(item._identifier, itemName, 'The item has a different identifier');
13  });
14 });
15
```

Done semuanya selesai dan di truffle ada pengujian unit nah disini akan menerapkan unit test super sederhana dan lihat apakah bisa menguji item yang dibuat. Pertama-tama, hapus tes di folder "/test". Setelah it masukkan tes baru dengan code seperti berikut

Jika sudah buka powershell baru dan tuliskan truffle test dan akan menampilkan seperti in

```
PS C:\Users\Fajri\s06-eventtrigger\test> truffle test
Using network 'test'.

Compiling your contracts...
=====
✓ Fetching solc version list from solc-bin. Attempt #1
> Everything is up to date, there is nothing to compile.

Contract: ItemManager
  ✓ ... should let you create new Items. (726ms)

1 passing (976ms)

PS C:\Users\Fajri\s06-eventtrigger\test>
```



**THANKS
YOU**