**#此次新增部分的註解已用藍色特別標註**

**Bank.sol**

pragma solidity ^0.4.23;

contract Bank {

// 此合約的擁有者

address private owner;

// 儲存所有會員的餘額

mapping (address => uint256) private balance;

/\*宣告

cdmoney：存定存本金

cdtime：存定存期數

\*/

mapping (address => uint256) private cdmoney;

mapping (address => uint256) private cdtime;

// 事件們，用於通知前端 web3.js

event DepositEvent(address indexed from, uint256 value, uint256 timestamp);

event WithdrawEvent(address indexed from, uint256 value, uint256 timestamp);

event TransferEvent(address indexed from, address indexed to, uint256 value, uint256 timestamp);

/\*事件

CDEvent：存定存的事件

CDcompleteEvent：定存期滿解約的事件

CDcancelEvent：定存未到期解約的事件

\*/

event CDEvent(address indexed from, uint256 value,uint256 times,uint256 timestamp);

event CDcompleteEvent(address indexed from, uint256 value, uint256 timestamp);

event CDcancelEvent(address indexed from, uint256 value, uint256 timestamp);

modifier isOwner() {

require(owner == msg.sender, "you are not owner");

\_;

}

// 建構子

constructor() public payable {

owner = msg.sender;

}

// 存錢

function deposit() public payable {

balance[msg.sender] += msg.value;

emit DepositEvent(msg.sender, msg.value, now);

}

// 提錢

function withdraw(uint256 etherValue) public {

uint256 weiValue = etherValue \* 1 ether;

require(balance[msg.sender] >= weiValue, "your balances are not enough");

msg.sender.transfer(weiValue);

balance[msg.sender] -= weiValue;

emit WithdrawEvent(msg.sender, etherValue, now);

}

// 轉帳

function transfer(address to, uint256 etherValue) public {

uint256 weiValue = etherValue \* 1 ether;

require(balance[msg.sender] >= weiValue, "your balances are not enough");

balance[msg.sender] -= weiValue;

balance[to] += weiValue;

emit TransferEvent(msg.sender, to, etherValue, now);

}

/\*定存購買功能

time為呼叫的參數，依照這個值可以決定定存期數

cdmoney是用來存定存金額的陣列

cdtime是用來存期數的陣列

設置require判斷定存期數是否介於最少1期和最多12期之間的規定，若不符，則拋出異常

並設置CDEvent監聽這個function

\*/

//定存購買

function CDbuy(uint256 time) public payable{

cdmoney[msg.sender] += msg.value;

cdtime[msg.sender] += time;

require(cdtime[msg.sender] <= 12 && cdtime[msg.sender] > =0, "期數最多12期，最少0期!");

emit CDEvent(msg.sender, msg.value,time,now);

}

/\*定存期滿功能

合約期滿可拿回所有本金加上利息，所以從cdmoney中抓取到的即為本金

透過cdtime[msg.sender]可以抓取到定存的期數，乘上一百分之一即為利息

並設置CDcompleteEvent監聽這個function

\*/

//定存期滿

function CDcomplete() public payable{

uint256 weiValue = cdmoney[msg.sender];

msg.sender.transfer(weiValue+weiValue\*cdtime[msg.sender]\*1/100);

emit CDcompleteEvent(msg.sender,weiValue+weiValue\*cdtime[msg.sender]\*1/100, now);

cdmoney[msg.sender] = 0;

cdtime[msg.sender] = 0;

}

/\*定存解約功能

releasetime為呼叫的參數，依照這個值可以決定已完成的期數

設置兩個require，一個require是判斷已完成的期數是否不超過購買定存的期數，若超過則拋出異常，另一個require是判斷已完成的期數是否符合期數最少0期且最多12期的規定，若不符，則拋出異常。

若皆符合兩個require，提前解約則可拿回所有本金加上已完成期數的利息

透過weiValue\*releasetime\*1/100可已得出提前解約應得的利息

並設置CDcancelEvent監聽這個function

\*/

//定存解約

function CDcancel(uint256 releasetime) public payable{

uint256 weiValue = cdmoney[msg.sender];

require(releasetime <= cdtime[msg.sender], "期數超過購買定存之期數");

require(releasetime <= 12 && releasetime > =0, "期數最多12期，最少0期!");

msg.sender.transfer(weiValue+weiValue\*releasetime\*1/100);

emit CDcancelEvent(msg.sender,weiValue+weiValue\*releasetime\*1/100, now);

cdmoney[msg.sender] = 0;

cdtime[msg.sender] = 0;

}

// 檢查銀行帳戶餘額

function getBankBalance() public view returns (uint256) {

return balance[msg.sender];

}

function getCertificateDeposit() public view returns (unit256){

return cd money[msg.sender];

}

function kill() public isOwner {

selfdestruct(owner);

}

}

**Index.js**

"use strict";  
  
let contractAddress = $("#contractAddress");  
let deployedContractAddressInput = $("#deployedContractAddressInput");  
let loadDeployedContractButton = $("#loadDeployedContractButton");  
let deployNewContractButton = $("#deployNewContractButton");  
  
let killContractButton = $("#killContractButton");  
  
let whoami = $("#whoami");  
let whoamiButton = $("#whoamiButton");  
let copyButton = $("#copyButton");  
  
let update = $("#update");  
  
let logger = $("#logger");  
  
let deposit = $("#deposit");  
let depositButton = $("#depositButton");  
  
let withdraw = $("#withdraw");  
let withdrawButton = $("#withdrawButton");  
  
let transferEtherTo = $("#transferEtherTo");  
let transferEtherValue = $("#transferEtherValue");  
let transferEtherButton = $("#transferEtherButton");

//宣告  
let CDmoney = $('#CDmoney'); //定存金額  
let CDyear = $('#CDyear'); //定存期數  
let CDbuyButton = $('#CDbuyButton'); //定存的購買按鈕  
let CDcompleteButton = $('#CDcompleteButton'); //定存的合約期滿按鈕  
let CDcompleteyear = $('#CDcompleteyear'); //已完成期數  
let CDreleaseButton = $('#CDreleaseButton'); //定存的提前解約按鈕  
  
let bankAddress = "";  
let nowAccount = "";  
  
// let web3 = new Web3('http://localhost:8545');  
let web3 = new Web3(  
 new Web3.providers.WebsocketProvider("ws://localhost:8545")  
);  
  
let bank = new web3.eth.Contract(bankAbi);  
  
function log(...inputs) {  
 for (let input of inputs) {  
 if (typeof input === "object") {  
 input = JSON.stringify(input, null, 2);  
 }  
 logger.html(input + "\n" + logger.html());  
 }  
}  
  
init();  
  
async function init() {  
 let accounts = await web3.eth.getAccounts();  
  
 for (let account of accounts) {  
 whoami.append(`<option value="${account}">${account}</option>`);  
 }  
 nowAccount = whoami.val();  
  
 update.trigger("click");  
  
 log(accounts, "以太帳戶");  
}  
  
  
  
// 當按下購買定存按鍵時  
CDbuyButton.on('click', async function () {  
  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面   
 waitTransactionStatus()  
 var a = document.getElementById("CDmoney").value ; //取得CDmoney欄位的輸入值  
 var b = document.getElementById("CDyear").value ; //取得CDyear欄位的輸入值  
 $("#money").text("金額: " + a); //更新money欄位資料 後面加上方取得的CDmoney  
 $("#year").text("期數: " + b); //更新year欄位資料 後面加上方取得的CDyear

// 購買定存  
 bank.methods.CDbuy(parseInt(CDyear.val(), 10)).send({  
 from: nowAccount,  
 gas: 3400000,  
 value: web3.utils.toWei(CDmoney.val(), 'ether')  
 })  
 .on('receipt', function (receipt) {  
 log(receipt.events.CDEvent.returnValues, '購買成功') //如果接收到值代表交易成功，顯示購買成功  
  
 // 觸發更新帳戶資料  
 update.trigger('click')  
  
 // 更新介面   
 doneTransactionStatus()  
 })  
 .on('error', function (error) {  
 log(error.toString())  
 // 更新介面   
 doneTransactionStatus()  
 })  
})  
  
  
  
// 當按下合約期滿按鍵時  
CDcompleteButton.on('click', async function () {  
  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus()  
 $("#money").text("金額: "); //更新money欄位資料 後面加上方取得的CDmoney  
 $("#year").text("期數: "); //更新year欄位資料 後面加上方取得的CDyear  
 // 合約期滿  
 bank.methods.CDcomplete().send({  
 from: nowAccount,  
 gas: 3400000  
 })  
 .on('receipt', function (receipt) {  
 log(receipt.events.CDcompleteEvent.returnValues, '期滿領回成功')  
  
 // 觸發更新帳戶資料  
 update.trigger('click')  
  
 // 更新介面   
 doneTransactionStatus()  
 })  
 .on('error', function (error) {  
 log(error.toString())  
 // 更新介面   
 doneTransactionStatus()  
 })  
})  
  
  
// 當按下提前解約按鍵時  
CDreleaseButton.on('click', async function () {  
  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus()  
 $("#money").text("金額: "); //更新money欄位資料 後面加上方取得的CDmoney  
 $("#year").text("期數: "); //更新year欄位資料 後面加上方取得的CDyear

// 提前解約  
 bank.methods.CDcancel(parseInt(CDcompleteyear.val(), 10)).send({  
 from: nowAccount,  
 gas: 3400000  
 })  
 .on('receipt', function (receipt) {  
 log(receipt.events.CDcancelEvent.returnValues, '解約成功')  
  
 // 觸發更新帳戶資料  
 update.trigger('click')  
  
 // 更新介面   
 doneTransactionStatus()  
 })  
 .on('error', function (error) {  
 log(error.toString())  
 // 更新介面   
 doneTransactionStatus()  
 })  
})  
  
  
  
  
  
// 當按下載入既有合約位址時  
loadDeployedContractButton.on("click", function() {  
 loadBank(deployedContractAddressInput.val());  
});  
  
// 當按下部署合約時  
deployNewContractButton.on("click", function() {  
 newBank();  
});  
  
// 當按下登入按鍵時  
whoamiButton.on("click", function() {  
 nowAccount = whoami.val();  
  
 update.trigger("click");  
});  
  
// 當按下複製按鍵時  
copyButton.on("click", function() {  
 let textarea = $("<textarea />");  
 textarea  
 .val(whoami.val())  
 .css({  
 width: "0px",  
 height: "0px",  
 border: "none",  
 visibility: "none"  
 })  
 .prependTo("body");  
  
 textarea.focus().select();  
  
 try {  
 if (document.execCommand("copy")) {  
 textarea.remove();  
 return true;  
 }  
 } catch (e) {  
 console.log(e);  
 }  
 textarea.remove();  
 return false;  
});  
  
// 當按下更新按鍵時  
update.on("click", async function() {  
 if (bankAddress != "") {  
 let ethBalance = await web3.eth.getBalance(nowAccount);  
 let bankBalance = await bank.methods  
 .getBankBalance()  
 .call({ from: nowAccount });

/\*定存金額顯示於畫面

透過await把bamk.methods下的.getCertificateDeposit().call({ from: nowAccount })封裝成recall function

\*/

let CertificateDeposit = await bank.methods

.getCertificateDeposit()

.call({ from: nowAccount });  
  
 log({  
 address: bankAddress,  
 ethBalance: ethBalance,  
 bankBalance: bankBalance  
 });  
 log("更新帳戶資料");  
  
 $("#ethBalance").text("以太帳戶餘額 (wei): " + ethBalance);  
 $("#bankBalance").text("銀行ETH餘額 (wei): " + bankBalance);  
 } else {  
 let ethBalance = await web3.eth.getBalance(nowAccount);  
  
 $("#ethBalance").text("以太帳戶餘額 (wei): " + ethBalance);  
 $("#bankBalance").text("銀行ETH餘額 (wei): ");  
 }  
});  
  
// 當按下刪除合約按鈕時  
killContractButton.on("click", async function() {  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus();  
 // 刪除合約  
 bank.methods  
 .kill()  
 .send({  
 from: nowAccount,  
 gas: 3400000  
 })  
 .on("receipt", function(receipt) {  
 log(bankAddress, "成功刪除合約");  
  
 bankAddress = "";  
 contractAddress.text("合約位址:" + bankAddress);  
 deployedContractAddressInput.val(bankAddress);  
  
 // 觸發更新帳戶資料  
 update.trigger("click");  
  
 // 更新介面  
 doneTransactionStatus();  
 })  
 .on("error", function(error) {  
 log(error.toString());  
 // 更新介面  
 doneTransactionStatus();  
 });  
});  
  
// 當按下存款按鍵時  
depositButton.on("click", async function() {  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus();  
 // 存款  
 bank.methods  
 .deposit()  
 .send({  
 from: nowAccount,  
 gas: 3400000,  
 value: web3.utils.toWei(deposit.val(), "ether")  
 })  
 .on("receipt", function(receipt) {  
 log(receipt.events.DepositEvent.returnValues, "存款成功");  
  
 // 觸發更新帳戶資料  
 update.trigger("click");  
  
 // 更新介面  
 doneTransactionStatus();  
 })  
 .on("error", function(error) {  
 log(error.toString());  
 // 更新介面  
 doneTransactionStatus();  
 });  
 bank.events.DepositEvent().on("data", function(event) {  
 log(event, "DepositEvent");  
 });  
});  
  
// 當按下提款按鍵時  
withdrawButton.on("click", async function() {  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus();  
 // 提款  
 bank.methods  
 .withdraw(parseInt(withdraw.val(), 10))  
 .send({  
 from: nowAccount,  
 gas: 3400000  
 })  
 .on("receipt", function(receipt) {  
 log(receipt.events.WithdrawEvent.returnValues, "提款成功");  
  
 // 觸發更新帳戶資料  
 update.trigger("click");  
  
 // 更新介面  
 doneTransactionStatus();  
 })  
 .on("error", function(error) {  
 log(error.toString());  
 // 更新介面  
 doneTransactionStatus();  
 });  
});  
  
// 當按下轉帳按鍵時  
transferEtherButton.on("click", async function() {  
 if (bankAddress == "") {  
 return;  
 }  
  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus();  
 // 轉帳  
 bank.methods  
 .transfer(transferEtherTo.val(), parseInt(transferEtherValue.val(), 10))  
 .send({  
 from: nowAccount,  
 gas: 3400000  
 })  
 .on("receipt", function(receipt) {  
 log(receipt.events.TransferEvent.returnValues, "轉帳成功");  
  
 // 觸發更新帳戶資料  
 update.trigger("click");  
  
 // 更新介面  
 doneTransactionStatus();  
 })  
 .on("error", function(error) {  
 log(error.toString());  
 // 更新介面  
 doneTransactionStatus();  
 });  
});  
  
// 載入bank合約  
function loadBank(address) {  
 if (!(address === undefined || address === null || address === "")) {  
 let bank\_temp = new web3.eth.Contract(bankAbi);  
 bank\_temp.options.address = address;  
  
 if (bank\_temp != undefined) {  
 bankAddress = address;  
 bank.options.address = bankAddress;  
  
 contractAddress.text("合約位址:" + address);  
 log(bank, "載入合約");  
  
 update.trigger("click");  
 } else {  
 log(address, "載入失敗");  
 }  
 }  
}  
  
// 新增bank合約  
async function newBank() {  
 // 解鎖  
 let unlock = await unlockAccount();  
 if (!unlock) {  
 return;  
 }  
  
 // 更新介面  
 waitTransactionStatus();  
  
 bank  
 .deploy({  
 data: bankBytecode  
 })  
 .send({  
 from: nowAccount,  
 gas: 3400000  
 })  
 .on("receipt", function(receipt) {  
 log(receipt, "部署合約");  
  
 // 更新合約介面  
 bankAddress = receipt.contractAddress;  
 bank.options.address = bankAddress;  
 contractAddress.text("合約位址:" + receipt.contractAddress);  
 deployedContractAddressInput.val(receipt.contractAddress);  
  
 update.trigger("click");  
  
 // 更新介面  
 doneTransactionStatus();  
 });  
}  
  
function waitTransactionStatus() {  
 $("#accountStatus").html(  
 '帳戶狀態 <b style="color: blue">(等待交易驗證中...)</b>'  
 );  
}  
  
function doneTransactionStatus() {  
 $("#accountStatus").text("帳戶狀態");  
}  
  
async function unlockAccount() {  
 let password = prompt("請輸入你的密碼", "");  
 if (password == null) {  
 return false;  
 } else {  
 return web3.eth.personal  
 .unlockAccount(nowAccount, password, 60)  
 .then(function(result) {  
 return true;  
 })  
 .catch(function(err) {  
 alert("密碼錯誤");  
 return false;  
 });  
 }  
}