Bank.js / bank.js：Bytecode 跟 ABI 都是從 Remix 上複製下來。
web3.min.js：沒有更改。

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;

	// 儲存定存金額跟期數
	mapping (address => uint256) private CD_balance;
	mapping (address => uint256) private CD_period;

	// 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);

	// 購買定存、合約期滿、提前解約事件
	event CD_DepositEvent(address indexed from, uint256 value, uint256
timestamp);
	event CD_WithdrawEvent(address indexed from, uint256 value, uint256
timestamp);
	event CD_all_WithdrawEvent(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);
    }

    /* CD 定存存錢
    CD_period_為呼叫參數，這個值決定定存期數，msg.value 為定存金額。
    設置 require 判斷當銀行的錢不夠付定存、定存期數小於 1 或大於 12 都
    要拋出異常，並設置 CD_DepositEvent 監聽這個 function。當按過一次定
    存，還沒解約再按第二次的時候，會把前一次的定存蓋掉(前一次的錢先
    還回來再扣除第二次新的定存金額)，以新的金額跟期數為主。
    */
    function CD_deposit(uint256 CD_period_) public payable {
        require(balance[msg.sender] >= msg.value, "your balances are not
enough");
        require(CD_period_ >= 1, "your period should >= 1");
        require(CD_period_ <= 12, "your period should <= 12");
        balance[msg.sender] += CD_balance[msg.sender];
        balance[msg.sender] -= msg.value;
        CD_balance[msg.sender] = msg.value;
        CD_period[msg.sender] = CD_period_;
        emit CD_DepositEvent(msg.sender, msg.value, now);
    }

    /* CD all 提錢 (合約期滿)
    weiValue 為合約期滿拿回來的定存金額，定存解除之後把暫時儲存的定
    存金額跟期數都歸零，並設置 CD_all_WithdrawEvent 監聽這個
    function。*/
    function CD_all_withdraw() public {
        uint256 weiValue = (CD_balance[msg.sender] + CD_balance[msg.sender]
* CD_period[msg.sender] / 100);
```

```
          balance[msg.sender] += weiValue;
          msg.sender.transfer(weiValue);
          CD_balance[msg.sender] = 0;
          CD_period[msg.sender] = 0;
          emit CD_all_WithdrawEvent(msg.sender, weiValue, now);
     }

     // CD  提錢  (提前解約)
     finished_period 為呼叫參數，這個值決定提前解約期數。weiValue 為提前
     解約拿回來的定存金額，定存解除之後把暫時儲存的定存金額跟期數都
     歸零，設置 require 判斷輸入的期數要大於等於 1 小於等於，否則拋出異
     常，當初設定的定存期數並設置 CD_WithdrawEvent 監聽這個
     function。*/
     function CD_withdraw(uint256 finished_period) public {
          require(finished_period >= 1, "your period should >= 1");
          require(finished_period <= CD_period[msg.sender] , "your period should
<= your CD period");
          uint256 weiValue = (CD_balance[msg.sender] + CD_balance[msg.sender]
* CD_period[msg.sender] / 100 * finished_period / CD_period[msg.sender]);
          balance[msg.sender] += weiValue;
          msg.sender.transfer(weiValue);
          CD_balance[msg.sender] = 0;
          CD_period[msg.sender] = 0;
          emit CD_WithdrawEvent(msg.sender, weiValue, 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);
	}

	// 檢查銀行帳戶餘額
	function getBankBalance() public view returns (uint256) {
		return balance[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");

// 抓取 HTML 的欄位、按鈕 id
let CD_deposit_text = $("#CD_deposit_text");
let CD_period_text = $("#CD_period_text");

let CD_deposit = $("#CD_deposit");
let CD_period = $("#CD_period");
let CD_depositButton = $("#CD_depositButton");

let CD_finished_period = $("#CD_finished_period");
let CD_withdrawButton = $("#CD_withdrawButton");

let CD_all_withdrawButton = $("#CD_all_withdrawButton");

let withdraw = $("#withdraw");
let withdrawButton = $("#withdrawButton");

let transferEtherTo = $("#transferEtherTo");
let transferEtherValue = $("#transferEtherValue");
let transferEtherButton = $("#transferEtherButton");

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 });
        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();
            // 刪除合約的時候順便把定存資訊清空
            CD_deposit_text.text("金額：  ");
            CD_period_text.text("期數：  ");
        })
        .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");
    });
});

// 當按下購買定存按鍵時
CD_depositButton.on("click", async function() {
    if (bankAddress == "") {
        return;
    }

    // 解鎖
    let unlock = await unlockAccount();
    if (!unlock) {
        return;
    }
    // 更新介面
    waitTransactionStatus();
    // 存款
    /* CD_period.val()抓取網頁上輸入的定存期數，CD_deposit.val()為網頁上輸
    入的定存金額，將參數傳給合約紀錄期數跟金額 */
    bank.methods
        .CD_deposit(parseInt(CD_period.val(), 10))
        .send({
            from: nowAccount,
            gas: 3400000,
            value: web3.utils.toWei(CD_deposit.val(), "ether")
        })
        .on("receipt", function(receipt) {
            log(receipt.events.CD_DepositEvent.returnValues, "CD 存款成功 (購買定
存)");

            // 觸發更新帳戶資料
            update.trigger("click");

            // 更新介面
            doneTransactionStatus();
            // 定存成功更新頁面上定存資訊
            CD_deposit_text.text("金額：  " + CD_deposit.val());
            CD_period_text.text("期數：  " + CD_period.val());
        })
        .on("error", function(error) {
            log(error.toString());
            // 更新介面
            doneTransactionStatus();
        });
    bank.events.CD_DepositEvent().on("data", function(event) {
        log(event, "CD_DepositEvent");
    });
});

// 當按下合約期滿按鍵時
CD_all_withdrawButton.on("click", async function() {
    if (bankAddress == "") {
        return;
    }

    // 解鎖
    let unlock = await unlockAccount();
    if (!unlock) {
        return;
    }
    /* 智能合約在定存時有儲存定存金額跟期數所以這邊讓合約那邊計算取回
    款項就好，不用傳參數。 */
    // 更新介面
    waitTransactionStatus();
    // 提款
    bank.methods
        .CD_all_withdraw()
        .send({
            from: nowAccount,
            gas: 3400000
        })
        .on("receipt", function(receipt) {
            log(receipt.events.CD_all_WithdrawEvent.returnValues, "CD all 提款成功
(合約期滿)");

            // 觸發更新帳戶資料
            update.trigger("click");

            // 更新介面
            doneTransactionStatus();
            // 解除定存清空頁面上定存資訊
            CD_deposit_text.text("金額：  ");
            CD_period_text.text("期數：  ");
        })
        .on("error", function(error) {
            log(error.toString());
            // 更新介面
            doneTransactionStatus();
        });
```

```
});

// 當按下提前解約按鍵時
CD_withdrawButton.on("click", async function() {
    if (bankAddress == "") {
        return;
    }

    // 解鎖
    let unlock = await unlockAccount();
    if (!unlock) {
        return;
    }

    // 更新介面
    waitTransactionStatus();
    // 提款
    /* CD_finished_period.val()取得提前解約期數並轉數字，將此參數傳給智能
    合約去計算領回的錢。*/
    bank.methods
        .CD_withdraw(parseInt(CD_finished_period.val(), 10))
        .send({
            from: nowAccount,
            gas: 3400000
        })
        .on("receipt", function(receipt) {
            log(receipt.events.CD_WithdrawEvent.returnValues, "CD 提款成功 (提前
解約)");

            // 觸發更新帳戶資料
            update.trigger("click");

            // 更新介面
            doneTransactionStatus();
            // 解除定存清空頁面上定存資訊
            CD_deposit_text.text("金額：  ");
            CD_period_text.text("期數：  ");
        })
        .on("error", function(error) {
            log(error.toString());
            // 更新介面
            doneTransactionStatus();
        });
});

// 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;
            });
    }
}
```

Index.html

```
<!DOCTYPE html>
<html>

<head>
        <meta charset="UTF-8">
        <title>DApp-example: Bank</title>
        <style>
                .flex {
                        display: flex;
                }

                .item {
                        flex: 1;
                        padding: 2rem;
                }
        </style>
</head>

<body>
        <div class="flex">
                <div class="item">
                        <h3>銀行合約</h3>
                        <h5 id="contractAddress">合約位址: </h5>
                        <input id="deployedContractAddressInput" type="text"
placeholder="請輸入既有合約位址" />
```

```
            <button id="loadDeployedContractButton">載入既有合約位址
</button>
            <button id="deployNewContractButton">部署新合約</button>

            <hr>
            <h3>刪除合約</h3>
            <button id="killContractButton">刪除目前合約</button>

            <hr>
            <h3>帳戶登入</h3>
            <select id="whoami"></select>
            <button id="whoamiButton">登入</button>
            <button id="copyButton">複製</button>

            <hr>
            <h3 id="accountStatus">帳戶狀態</h3>
            <h5 id="ethBalance">以太帳戶餘額 (wei): </h5>
            <h5 id="bankBalance">銀行 ETH 餘額 (wei): </h5>

            <hr>
            <h3 id="updateCoin">更新餘額</h3>
            <button id="update">執行</button>

            <hr>
            <h3>存款</h3>
            <input id="deposit" type="number" placeholder="請輸入存款額度
(ether)">
            <button id="depositButton">執行</button>

            <hr>
            <h3>提款</h3>
            <input id="withdraw" type="number" placeholder="請輸入提款額
度 (ether)">
            <button id="withdrawButton">執行</button>

            <hr>
            <h3>轉帳 ETH</h3>
            <input id="transferEtherTo" type="text" placeholder="請輸入轉帳
```

```
對象">
			<input id="transferEtherValue" type="number" placeholder="請輸
入轉帳額度 (ether)">
			<button id="transferEtherButton">執行</button>

			<!-- 網頁上輸入定存、解除定存的金額、期數欄位跟按紐、定
			存金額跟期數資訊 -->
			<hr>
			<h3>購買定存</h3>
			<input id="CD_deposit" type="number" placeholder="請輸入定存
額度 (ether)">
			<input id="CD_period" type="number" placeholder="請輸入定存
期數">
			<button id="CD_depositButton">購買</button>
			<p id="CD_deposit_text">金額：</p>
			<p id="CD_period_text">期數：</p><br><br>

			<button id="CD_all_withdrawButton">合約期滿</button><br>
			<input id="CD_finished_period" type="number" placeholder="已合
約期滿">
			<button id="CD_withdrawButton">提前解約</button>
			<p>期數 | 利率<br>
			01 | 0.01<br>
			02 | 0.02<br>
			03 | 0.03<br>
			04 | 0.04<br>
			05 | 0.05<br>
			06 | 0.06<br>
			07 | 0.07<br>
			08 | 0.08<br>
			09 | 0.09<br>
			10 | 0.10<br>
			11 | 0.11<br>
			12 | 0.12<br>
			</p>
		</div>
		<div class="item" style="width:500px;">
			<h3>活動紀錄</h3>
```

```
                    <pre id="logger"></pre>
                </div>
        </div>

        <script
src='http://cdnjs.cloudflare.com/ajax/libs/jquery/3.1.0/jquery.min.js'></script>
        <script src="js/web3.min.js"></script>
        <script src="js/bank.js"></script>
        <script src="js/index.js"></script>
</body>

</html>
```