// BUDGET CONTROLLER

var budgetController = (function() {

var Expense = function(id, description, value, date, percentage) {

this.id = id;

this.description = description;

this.value = value;

this.date = date;

this.percentage = -1;

};

Expense.prototype.calcPercentage = function(totalIncome) {

if (totalIncome > 0) {

this.percentage = ((this.value / totalIncome) \* 100);

} else {

this.percentage = -1;

}

};

Expense.prototype.getPercentage = function() {

return this.percentage;

};

var Income = function(id, description, value, date) {

this.id = id;

this.description = description;

this.value = value;

this.date = date;

};

var calculateTotal = function(type) {

var sum = 0;

data.allItems[type].forEach(function(cur) {

sum += cur.value;

});

data.totals[type] = sum;

};

var adjustDate = function(date) {

// function to lost the GMT correction that the system automatically does

// date is the Date object that came out of new Date(datepicker)

var dateString = date;

// re-create new date and convert to UTC String

dateString = new Date(dateString).toUTCString();

// split puts all the separate parts in an array

// slice drops (in this case up to but not including

// the element[4])

// join returns it all to a string

dateString = dateString.split(' ').slice(0, 4).join(' ');

return dateString;

}

var totalExpenses = 0;

var data = {

allItems: {

exp: [],

inc: []

},

totals: {

exp: 0,

inc: 0

},

budget: 0,

percentage: -1 // -1 to indicate something is non-existent

};

return {

addItem: function(type, des, val, date) { // different names cause less confusion

var newItem, ID;

date = adjustDate(date);

// Create new ID

if (data.allItems[type].length > 0) {

ID = data.allItems[type][data.allItems[type].length - 1].id + 1;

} else {

ID = 0;

}

// Create new items based on inc or exp type

if (type === 'exp') {

newItem = new Expense(ID, des, val, date);

} else if (type === 'inc') {

newItem = new Income(ID, des, val, date);

}

// push it into your data structure

data.allItems[type].push(newItem);

// return the new element

return newItem;

},

deleteItem: function(type, id) {

var ids, index;

// need to create an array with all the elements

// and then find the one with that id

// in other words we can't just use id as the index in the array

ids = data.allItems[type].map(function(current) { // cb has access to the current index, and the entire array

// map returns a new array

return current.id;

});

index = ids.indexOf(id);

if (index !== -1) {

data.allItems[type].splice(index, 1);

}

},

calculateBudget: function() {

// calculate total income and expenses

calculateTotal('exp');

calculateTotal('inc');

// calculate the budget: income - expenses

data.budget = data.totals.inc - data.totals.exp;

// calculate the percentage of income that we spent

if (data.totals.inc > 0) {

data.percentage = Math.round((data.totals.exp / data.totals.inc) \* 100);

} else {

data.percentage = -1;

}

},

calculatePercentages: function() {

// a = 20, b = 10, c = 40

// income = 100;

// a = 20 / 100 = 20%

// b = 10 / 100 = 10%

// c would be 40%

data.allItems.exp.forEach(function(cur) {

cur.calcPercentage(data.totals.inc);

});

},

getPercentages: function() {

var allPerc = data.allItems.exp.map(function(cur) {

return cur.getPercentage();

});

return allPerc;

},

getBudget: function() {

return {

budget: data.budget,

totalInc: data.totals.inc,

totalExp: data.totals.exp,

percentage: data.percentage

}

},

};

})();

// UI CONTROLLER

var UIController = (function() {

var DOMstrings = {

inputType: '.add\_\_type',

inputDescription: '.add\_\_description',

inputValue: '.add\_\_value',

inputDate: '.add\_\_date',

inputBtn: '.add\_\_btn',

incomeContainer: '.income\_\_list',

expensesContainer: '.expenses\_\_list',

budgetLabel: '.budget\_\_value',

incomeLabel: '.budget\_\_income--value',

expensesLabel: '.budget\_\_expenses--value',

percentageLabel: '.budget\_\_expenses--percentage',

container: '.container',

expensesPercLabel: '.item\_\_percentage',

dateLabel: '.budget\_\_title--month'

};

var formatNumber = function(num, type) {

var numSplit, int, dec;

// + or - b4 the number

// exactly 2 decimal points

// comman separating the thousands

num = Math.abs(num);

num = num.toFixed(2);

numSplit = num.split('.');

int = numSplit[0];

if (int.length > 3) {

int = int.substr(0, int.length - 3) + ',' + int.substr(int.length - 3, 3);

}

dec = numSplit[1];

return (type === 'exp' ? '-' : '+') + ' ' + int + '.' + dec;

};

var nodeListForEach = function(list, callback) {

for (var i = 0; i < list.length; i++) {

callback(list[i], i);

}

};

var getMonth = function(mnth) {

var months = ['January', 'February', 'March', 'April',

'May', 'June', 'July', 'August',

'September', 'October', 'November', 'December'];

mnth = months[mnth];

return mnth;

};

var displayDate = function(enteredDate) {

//console.log(Object.prototype.toString.call(enteredDate) === '[object Date]');

// above checks that an actual data object was created

var date, month, day;

date = new Date(enteredDate);

month = date.getMonth();

day = date.getDate();

month = getMonth(month);

return month + ' ' + day;

};

return {

getInput: function() {

return {

type: document.querySelector(DOMstrings.inputType).value, // inc or exp

description: document.querySelector(DOMstrings.inputDescription).value,

value: parseFloat(document.querySelector(DOMstrings.inputValue).value),

date: document.querySelector(DOMstrings.inputDate).value

};

},

addListItem: function(obj, type) {

var html, newHtml, element;

// create HTML string with placeholder text

if (type === 'inc') {

element = DOMstrings.incomeContainer;

html = '<div class="item clearfix" id="inc-%id%"><div class="item\_\_description">%description%</div><div class="item\_\_date">%date%</div><div class="right clearfix"><div class="item\_\_value">%value%</div><div class="item\_\_delete"><button class="item\_\_delete--btn"><i class="ion-ios-close-outline"></i></button></div></div></div>'

} else if (type === 'exp') {

element = DOMstrings.expensesContainer;

html = '<div class="item clearfix" id="exp-%id%"><div class="item\_\_description">%description%</div><div class="item\_\_date">%date%</div><div class="right clearfix"><div class="item\_\_value">%value%</div><div class="item\_\_percentage">21%</div><div class="item\_\_delete"><button class="item\_\_delete--btn"><i class="ion-ios-close-outline"></i></button></div></div></div>'

}

// replace the placeholder text with some actual data (data received from object)

newHtml = html.replace('%id%', obj.id);

newHtml = newHtml.replace('%description%', obj.description);

newHtml = newHtml.replace('%value%', formatNumber(obj.value));

newHtml = newHtml.replace('%date%', displayDate(obj.date));

//newHtml = newHtml.replace('%date%', formatNumber(obj.value));

// insert HTML into DOM

document.querySelector(element).insertAdjacentHTML('beforeend', newHtml);

},

deleteListItem: function(selectorID) {

var el = document.getElementById(selectorID);

el.parentNode.removeChild(el);

},

clearFields: function() {

var fields, fieldsArr;

fields = document.querySelectorAll(DOMstrings.inputDescription + ', ' + DOMstrings.inputValue); // selectors will be separated by a comma

// tricking the list items as an array

// call the slice method using 'call' and then passing the fields variable into it and then it becomes the 'this'

fieldsArr = Array.prototype.slice.call(fields); // setting the fields variable to the this

fieldsArr.forEach(function(current, index, array) {

current.value = "";

});

fieldsArr[0].focus();

},

displayBudget: function(obj) {

var type;

obj.budget > 0 ? type = 'inc' : type = 'exp';

document.querySelector(DOMstrings.budgetLabel).textContent = formatNumber(obj.budget, type);

document.querySelector(DOMstrings.incomeLabel).textContent = formatNumber(obj.totalInc, 'inc');

document.querySelector(DOMstrings.expensesLabel).textContent = formatNumber(obj.totalExp, 'exp');

if (obj.percentage > 0) {

document.querySelector(DOMstrings.percentageLabel).textContent = obj.percentage + '%';

} else {

document.querySelector(DOMstrings.percentageLabel).textContent = '---';

}

},

displayPercentages: function(percentages) {

var fields = document.querySelectorAll(DOMstrings.expensesPercLabel);

// node list = where all of our html elements are stored (dom tree)

// each element is called a node

// this is very powerful code

nodeListForEach(fields, function(current, index) {

if (percentages[index] > 0) {

current.textContent = percentages[index] + '%';

} else {

current.textContent = '---';

}

});

},

displayMonth: function() {

var now, year, month, months;

now = new Date();

year = now.getFullYear();

month = now.getMonth();

document.querySelector(DOMstrings.dateLabel).textContent = getMonth(month) + ' ' + year;

},

changedType: function() {

var fields = document.querySelectorAll(

DOMstrings.inputType +',' +

DOMstrings.inputDescription + ',' +

DOMstrings.inputValue + ',' +

DOMstrings.inputDate);

nodeListForEach(fields, function(cur) {

cur.classList.toggle('red-focus');

});

document.querySelector(DOMstrings.inputBtn).classList.toggle('red');

},

// remember for node list you can not use the forEach method

////// POWERFUL CODE

getDOMstrings: function() {

return DOMstrings;

}

};

})();

// separation of concerns - each part of the app should do only one thing independently.

// GLOBAL APP CONTROLLER

var controller = (function(budgetCtrl, UICtrl) {

var setupEventListeners = function() {

var DOM = UICtrl.getDOMstrings();

// seting up button listener here so we can decide how to delegate to other controllers

document.querySelector(DOM.inputBtn).addEventListener('click', ctrlAddItem);

// key press added as global event (can happen anywhere in the document)

document.addEventListener('keypress', function(event) {

if (event.keyCode === 13 || event.which === 13) {

ctrlAddItem();

}

});

// attaching the event handler that has a common parent element to both (income and expenses)

document.querySelector(DOM.container).addEventListener('click', ctrlDeleteItem);

document.querySelector(DOM.inputType).addEventListener('change', UICtrl.changedType);

};

var updateBudget = function() {

// 4. calculate budget

budgetCtrl.calculateBudget();

// return the budget

var budget = budgetCtrl.getBudget();

// 5. display budget on the UI

UICtrl.displayBudget(budget);

};

var updatePercentages = function() {

// 1. calculate percentages

budgetCtrl.calculatePercentages();

// 2. read percentages from budget controller

var percentages = budgetCtrl.getPercentages();

// 3. update the UI with the new percentages

UICtrl.displayPercentages(percentages);

};

var ctrlAddItem = function() {

var input, newItem;

// 1. get the field input data

var input = UICtrl.getInput();

if (input.description !== "" && !isNaN(input.value) && input.value > 0) {

// 2. add item to the budget controller

var newItem = budgetCtrl.addItem(input.type, input.description, input.value, new Date(input.date));

// 3. add item to the UI as well

UICtrl.addListItem(newItem, input.type);

// clearing fields

UICtrl.clearFields();

// calculate and update budget

updateBudget();

// calculate and update percentages

updatePercentages();

}

};

var ctrlDeleteItem = function(event) {

var itemID, splitID, type, ID;

// not the best way to do this / we are traversing the hard coded HMTL

itemID = event.target.parentNode.parentNode.parentNode.parentNode.id;

if (itemID) {

// inc-1 needs to be split up

// if you had 'income-1-type-number-range' if would return an array [income, 1, type, number, range]

splitID = itemID.split('-');

type = splitID[0];

ID = parseInt(splitID[1]);

// 1. delete item from data structure

budgetCtrl.deleteItem(type, ID);

// 2. delete item from UI

UICtrl.deleteListItem(itemID);

// 3. update and show new budget / totals

updateBudget();

// calculate and update percentages

updatePercentages();

}

};

// seting up the init call, since we want it

// to be public we need to return it in an object

return {

init: function() {

console.log('App has started');

UICtrl.displayMonth();

UICtrl.displayBudget({

budget: 0,

totalInc: 0,

totalExp: 0,

percentage: -1

});

setupEventListeners();

}

}

})(budgetController, UIController); // now this controller knows about the other two and can use their code.

// only line of code on the outside

controller.init();