# Solutions-JS Advanced: Exam 15 July 2018

# Problem 1. Sticky Notes (Simple DOM Interaction)

function addSticker() {

let $titleInput = $('input[class="title"]');

//let $titleInput = $('.title');

let $textInput = $('input[class="content"]');

//let $textInput = $('.content');

let $ul = $('#sticker-list');

if ($titleInput.val() && $textInput.val()) {

let $li = $('<li>');

$li.addClass('note-content');

let $a = $('<a>');

$a.addClass('button');

$a.text('x');

$a.on('click', () => $li.remove());

let $h2 = $('<h2>');

$h2.text($titleInput.val());

let $hr = $('<hr>');

let $p = $('<p>');

$p.text($textInput.val());

$li.append($a, $h2, $hr, $p);

// let li = $.parseHTML(`<li class="note-content"><a class="button">x</a><h2>${$titleInput.val()}</h2><hr/><p>${$textInput.val()}</p></li>`);

// let $li = $(li);

// $li.find('.button').on('click', () => $li.remove());

$ul.append($li);

$titleInput.val('');

$textInput.val('');

}

}

function addSticker() {

let $title = $('.title');

let $content = $('.content');

let $list = $('#sticker-list');

if ($title.val() && $content.val()) {

createNote();

reset();

}

// function createNote() {

// let $li = $('<li>');

// $li.addClass('note-content');

// let $a = $('<a>');

// $a.addClass('button');

// $a.text('x');

// $a.on('click', () => $li.remove());

// let $h = $('<h2>');

// $h.text($title.val());

// let $p = $('<p>');

// $p.text($content.val());

// $li.append($a);

// $li.append($h);

// $li.append($('<hr>'));

// $li.append($p);

// $list.append($li);

// }

function createNote() {

let parsedHtml = $.parseHTML(`<li class="note-content"><a class="button">x</a><h2>${$title.val()}</h2><hr><p>${$content.val()}</p></li>`);

let $el = $(parsedHtml);

$el.find('.button').on('click', () => $el.remove());

$list.append($el);

}

function reset() {

$title.val('');

$content.val('');

}

}

# Problem 2. Calculator (Unit Testing)

class Calculator {

constructor() {

this.expenses = [];

}

add(data) {

this.expenses.push(data);

}

divideNums() {

let divide;

for (let i = 0; i < this.expenses.length; i++) {

if (typeof (this.expenses[i]) === 'number') {

if (i === 0 || divide===undefined) {

divide = this.expenses[i];

} else {

if (this.expenses[i] === 0) {

return 'Cannot divide by zero';

}

divide /= this.expenses[i];

}

}

}

if (divide !== undefined) {

this.expenses = [divide];

return divide;

} else {

throw new Error('There are no numbers in the array!')

}

}

toString() {

if (this.expenses.length > 0)

return this.expenses.join(" -> ");

else return 'empty array';

}

orderBy() {

if (this.expenses.length > 0) {

let isNumber = true;

for (let data of this.expenses) {

if (typeof data !== 'number')

isNumber = false;

}

if (isNumber) {

return this.expenses.sort((a, b) => a - b).join(', ');

}

else {

return this.expenses.sort().join(', ');

}

}

else return 'empty';

}

}

module.exports = {Calculator};

let expect = require('chai').expect;

let Calculator = require('../02Calculator').Calculator;

//In Judge must be paste without this above

describe('Calculator', function(){

describe('constructor', function(){

it('should have property expenses', function(){

let calculator = new Calculator();

expect(calculator).to.haveOwnProperty('expenses');

});

it('should have property expenses empty array', function(){

let calculator = new Calculator();

expect(JSON.stringify(calculator.expenses)).to.be.equal('[]');

});

});

describe('add(data)', function(){

let calculator;

beforeEach(function(){

calculator = new Calculator();

});

it('should add strings to expenses', function(){

calculator.add('abc');

calculator.add('cdf');

expect(JSON.stringify(calculator.expenses)).to.be.equal('["abc","cdf"]');

});

it('should add numbers to expenses', function(){

calculator.add(8);

calculator.add(-10);

calculator.add(7.9);

expect(JSON.stringify(calculator.expenses)).to.be.equal("[8,-10,7.9]");

});

it('should add mixed to expenses', function(){

calculator.add('abc');

calculator.add(10);

calculator.add(['cdf', 8]);

calculator.add({name: 'Poli'});

expect(JSON.stringify(calculator.expenses)).to.be.equal('["abc",10,["cdf",8],{"name":"Poli"}]');

});

});

describe('divideNums()', function(){

let calculator;

beforeEach(function(){

calculator = new Calculator();

});

it('should throw for empty expenses', function(){

expect(() => calculator.divideNums()).to.throw('There are no numbers in the array!');

});

it('should throw for expenses without numbers', function(){

calculator.add('abc');

calculator.add('def');

expect(() => calculator.divideNums()).to.throw('There are no numbers in the array!');

});

it('should return message for expenses with numbers and zero', function(){

calculator.add(80);

calculator.add('abc');

calculator.add(10);

calculator.add('100');

calculator.add(0);

expect(calculator.divideNums()).to.be.equal('Cannot divide by zero');

});

it('should return correct value for expenses with numbers', function(){

calculator.add(80);

calculator.add('abc');

calculator.add(10);

calculator.add('100');

expect(calculator.divideNums()).to.be.equal(8);

});

it('should return correct value for expenses with floting point numbers', function(){

calculator.add(0.8);

calculator.add('abc');

calculator.add(10);

calculator.add('100');

expect(calculator.divideNums()).to.be.closeTo(0.08, 0.01);

});

it('should return correct value for expenses with negative numbers', function(){

calculator.add(80);

calculator.add('abc');

calculator.add(-10);

calculator.add('100');

expect(calculator.divideNums()).to.be.equal(-8);

});

});

describe('toString()', function(){

let calculator;

beforeEach(function(){

calculator = new Calculator();

});

it('should return "empty array" for empty expenses', function(){

expect(calculator.toString()).to.be.equal("empty array")

});

it('should return correct value for one member', function(){

calculator.add('abc');

expect(calculator.toString()).to.be.equal('abc');

});

it('should return correct value for more members', function(){

calculator.add('abc');

calculator.add('100');

calculator.add(10);

calculator.add(0.7);

calculator.add(-18);

//calculator.add([]);//the second zero test don't past with this

//calculator.add({});//the second zero test don't past with this

expect(calculator.toString()).to.be.equal('abc -> 100 -> 10 -> 0.7 -> -18');

//expect(calculator.toString()).to.be.equal('abc -> 100 -> 10 -> 0.7 -> -18 -> [] -> {}');

});//the second zero test don't past with this

});

describe('orderBy()', function(){

let calculator;

beforeEach(function(){

calculator = new Calculator();

});

it('should return correct values for numbers', function(){

calculator.add(6);

calculator.add(-10);

calculator.add(0.9);

calculator.add(190);

calculator.add(-9.7);

calculator.add(6);

expect(calculator.orderBy()).to.be.equal('-10, -9.7, 0.9, 6, 6, 190');

});

it('should return correct values for not numbers', function(){

calculator.add('abc');

calculator.add('def');

calculator.add('xyz');

expect(calculator.orderBy()).to.be.equal('abc, def, xyz');

});

it('should return correct values for numbers and non numbers', function(){

calculator.add(6);

calculator.add(-10);

calculator.add(0.9);

calculator.add('abc');

calculator.add(190);

calculator.add('xyz');

calculator.add(-9.7);

calculator.add('def');

calculator.add(6);

expect(calculator.orderBy()).to.be.equal('-10, -9.7, 0.9, 190, 6, 6, abc, def, xyz');

});

});

});

const Calculator = require('../02Calculator');

const assert = require('chai').assert;

describe('Calucator', function() {

let calculator;

beforeEach(function(){

calculator = new Calculator();

});

it('Contains a property expenses that is initialized to an empty array. ', function() {

assert.isArray(calculator.expenses);

assert.isEmpty(calculator.expenses);

});

describe('Function add(data) – adds the passed in item (of any type) to the expenses.', function() {

it('Add primitive', function() {

calculator.add(5);

calculator.add('text');

calculator.add(1.5);

calculator.add(true);

assert.deepEqual(calculator.expenses, [5, 'text', 1.5, true])

});

it('Add reference', function() {

calculator.add({key: 'value'});

calculator.add([1]);

// calculator.add(function() {}); !!!!!!!

assert.deepEqual(calculator.expenses, [{ key: 'value' }, [ 1 ]]);

});

});

describe('Function divideNums()', function() {

it('standard 2', function() {

calculator.add(100);

calculator.add(2);

assert.equal(calculator.divideNums(), 50);

});

it('standard 3', function() {

calculator.add(100);

calculator.add(2);

calculator.add(5);

assert.equal(calculator.divideNums(), 10);

});

it('no input', function() {

assert.throw(() => calculator.divideNums(), 'There are no numbers in the array!');

});

it('no number input', function() {

calculator.add('pesho')

calculator.add({})

calculator.add('gosho')

// Number as string

assert.throw(() => calculator.divideNums(), 'There are no numbers in the array!');

});

it('division with floats', function() {

calculator.add(10.5)

calculator.add(2)

assert.closeTo(calculator.divideNums(), 5.25, 0.01);

});

it('division with zero', function() {

calculator.add(10.5);

calculator.add(0);

assert.equal(calculator.divideNums(), 'Cannot divide by zero');

});

});

describe('Function toString()', function() {

it('standard', function() {

calculator.add(10);

calculator.add('Pesho');

calculator.add(5);

assert.equal(calculator.toString(), '10 -> Pesho -> 5');

});

it('one input', function() {

calculator.add(1);

// assert.strictEqual(calculator.toString(), '1');

assert.equal(calculator.toString(), '1');

});

it('no input', function() {

assert.equal(calculator.toString(), 'empty array');

});

});

describe('Function orderBy() ', function() {

it('standard', function() {

calculator.add(10);

calculator.add(-3);

calculator.add(30);

calculator.add(1);

// Floats !!

assert.equal(calculator.orderBy(), '-3, 1, 10, 30');

});

it('non numbers', function() {

calculator.add({});

calculator.add([1,2,3]);

calculator.add('pesho');

assert.equal(calculator.orderBy(), '1,2,3, [object Object], pesho');

});

it('mixed values', function() {

calculator.add({});

calculator.add([1,2,3]);

calculator.add(100);

calculator.add('pesho');

calculator.add(-100);

assert.equal(calculator.orderBy(), '-100, 1,2,3, 100, [object Object], pesho');

});

});

});

# Problem 3. Book Collection

class BookCollection {

constructor(shelfGenre, room, shelfCapacity) {

this.room = room;

this.shelfGenre = shelfGenre;

this.shelfCapacity = shelfCapacity;

this.shelf = [];

return this;

}

get room() { return this.\_room; }

set room(room) {

if (room !== 'livingRoom' && room !== 'bedRoom' && room != 'closed') {

throw `Cannot have book shelf in ${room}`;

}

this.\_room = room;

}

get shelfCondition() { return this.shelfCapacity - this.shelf.length; }

addBook(bookName, bookAuthor, genre) {

//if(this.shelf.length === this.shelfCapacity){

if (this.shelfCondition === 0) {

this.shelf.shift();

}

let book = { bookName, bookAuthor };

if (genre){

book.genre = genre;

}

this.shelf.push(book);

this.shelf.sort((a,b) => a.bookAuthor.localeCompare(b.bookAuthor));

return this;

}

throwAwayBook(bookName){

this.shelf = this.shelf.filter(b => b.bookName !== bookName);

return this;

}

showBooks(genre){

let genreBooks = this.shelf.filter(b => b.genre === genre).map(b => `\uD83D\uDCD6 ${b.bookAuthor} - "${b.bookName}"`);

return `Results for search "${genre}":\n` + genreBooks.join('\n');

}

toString(){

let booksInShelf = "It's an empty shelf";

if(this.shelf.length > 0){

let books = this.shelf.map(b => `\uD83D\uDCD6 "${b.bookName}" - ${b.bookAuthor}`);

booksInShelf = `"${this.shelfGenre}" shelf in ${this.room} contains:\n` + books.join('\n');

}

return booksInShelf;

}

}

let livingRoom = new BookCollection("Programming", "livingRoom", 5)

.addBook("Introduction to Programming with C#", "Svetlin Nakov")

.addBook("Introduction to Programming with Java", "Svetlin Nakov")

.addBook("Programming for .NET Framework", "Svetlin Nakov");

console.log(livingRoom.toString());

let bedRoom = new BookCollection('Mixed', 'bedRoom', 5);

bedRoom.addBook("John Adams", "David McCullough", "history");

bedRoom.addBook("The Guns of August", "Cuentos para pensar", "history");

bedRoom.addBook("Atlas of Remote Islands", "Judith Schalansky");

bedRoom.addBook("Paddle-to-the-Sea", "Holling Clancy Holling");

console.log("Shelf's capacity: " + bedRoom.shelfCondition);

console.log(bedRoom.showBooks("history"));

let garden = new BookCollection("Programming", "garden");

class BookCollection {

constructor(shelfGenre, room, shelfCapacity) {

this.shelfGenre = shelfGenre;

this.room = room;

this.shelfCapacity = shelfCapacity; // Number

this.shelf = [];

return this;

}

get room() {

return this.\_room;

}

set room(value) {

switch (value) {

case 'livingRoom':

case 'bedRoom':

case 'closet':

this.\_room = value;

break;

default:

throw `Cannot have book shelf in ${value}`;

}

}

get shelfCondition() {

return this.shelfCapacity - this.shelf.length;

}

addBook(bookName, bookAuthor, genre) {

let book = {bookName, bookAuthor, genre};

if (this.shelfCondition <= 0) {

this.shelf.shift();

}

this.shelf.push(book);

this.shelf.sort((a, b) => a.bookAuthor.localeCompare(b.bookAuthor));

return this;

}

throwAwayBook(bookName) {

this.shelf = this.shelf.filter(b => b.bookName !== bookName);

return this;

}

showBooks(genre) { // if no books

let result = `Results for search "${genre}":\n`;

result += this.shelf

.filter(b => b.genre === genre)

.map(b => `\uD83D\uDCD6 ${b.bookAuthor} - "${b.bookName}"`)

.join('\n');

return result.trim();

}

toString() {

if (this.shelf.length <= 0) {

return "It's an empty shelf";

}

let result = `"${this.shelfGenre}" shelf in ${this.room} contains:\n`;

result += this.shelf

.map(b => `\uD83D\uDCD6 "${b.bookName}" - ${b.bookAuthor}`)

.join('\n');

return result.trim();

}

}

# Problem 4. Online Shop (DOM)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Online Shop</title>

<style>

.wrapper {

margin: auto;

width: 53%;

border: 3px solid rgb(26, 26, 26);

background-color: rgb(250, 250, 250);

padding: 10px;

}

#header {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 300%;

font-weight: bold;

color: rgb(26, 26, 26);

text-align: center;

padding: 10px;

}

li {

list-style-type: none;

}

.block {

display: inline-block;

margin-left: 165px;

}

.field {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 25px;

color: rgb(26, 26, 26);

}

.custom-select {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

background-color: rgb(255, 255, 255);

color: rgb(26, 26, 26);

border: 1px solid rgb(26, 26, 26);

padding: 2px;

margin-top: 10px;

width: 200px;

}

.input1 {

width: 40px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

border: 1px solid rgb(26, 26, 26);

background-color: #ffffff;

color: rgb(26, 26, 26);

padding: 2px;

}

#capacity {

width: 40px;

font-size: 20px;

padding: 2px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

}

#sum {

width: 80px;

font-size: 20px;

padding: 2px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

margin: 10px 10px;

}

.fullCapacity {

width: 40px;

font-size: 20px;

padding: 2px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

color: rgb(250, 0, 0);

border: 1px solid #f07777b2;

background-color: #f78989bb;

}

.display {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 15px;

border: 1px solid rgb(26, 26, 26);

padding: 2px;

background-color: #ffffff;

color: rgb(26, 26, 26);

margin-top: 5px;

width: 100%;

height: 300px;

}

.text {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

color: rgb(26, 26, 26);

margin-right: 10px;

}

.button:enabled {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

cursor: pointer;

border: 1px solid rgb(26, 26, 26);

padding: 5px 15px;

margin: 3px;

border-radius: 8px;

}

.button:disabled {

background: rgb(248, 248, 248);

cursor: pointer;

color: rgb(216, 216, 216);

font-size: 20px;

padding: 5px 15px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

margin: 3px;

border: 1px solid rgb(26, 26, 26);

border-radius: 8px;

}

</style>

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

</head>

<body>

<div class="wrapper">

</div>

<script src="04OnlineShop.js"></script>

<script>onlineShop('.wrapper')</script>

</body>

function onlineShop(selector) {

let form = `<div id="header">Online Shop Inventory</div>

<div class="block">

<label class="field">Product details:</label>

<br>

<input placeholder="Enter product" class="custom-select">

<input class="input1" id="price" type="number" min="1" max="999999" value="1"><label class="text">BGN</label>

<input class="input1" id="quantity" type="number" min="1" value="1"><label class="text">Qty.</label>

<button id="submit" class="button" disabled>Submit</button>

<br><br>

<label class="field">Inventory:</label>

<br>

<ul class="display">

</ul>

<br>

<label class="field">Capacity:</label><input id="capacity" readonly>

<label class="field">(maximum capacity is 150 items.)</label>

<br>

<label class="field">Price:</label><input id="sum" readonly>

<label class="field">BGN</label>

</div>`;

$(selector).html(form);

const maxCapacity = 150;

let $inputProduct = $('input[class="custom-select"]');

//let $inputProduct = $('.custom-select');

let $inputPrice = $('#price');

let $inputQuantity = $('#quantity');

let $buttonSubmit = $('#submit');

let $ulInventory = $('ul[class="display"]');

//let $ulInventory = $('.display');

let $outputCapacity = $('#capacity');

let $outputPrice = $('#sum');

// $inputProduct.on('input', () => {

// let isEmpty = $inputProduct.val() === '';

// $buttonSubmit.attr('disabled', isEmpty);

// });

$inputProduct.on('input', () => {

if ($inputProduct.val() === '') {

$buttonSubmit.attr('disabled', true);

}

else {

$buttonSubmit.attr('disabled', false);

}

});

$buttonSubmit.on('click', addProduct);

function addProduct() {

let product = $inputProduct.val();

let price = Number($inputPrice.val());

let quantity = Number($inputQuantity.val());

let oldCapacity = Number($outputCapacity.val());

let capacity = oldCapacity + quantity;

if (capacity < maxCapacity) {

let $li = $('<li>');

$li.text(`Product: ${product} Price: ${price} Quantity: ${quantity}`);

$ulInventory.append($li);

$outputCapacity.val(capacity);

let totalPrice = Number($outputPrice.val());

totalPrice += price;

$outputPrice.val(totalPrice);

}

else{

$outputCapacity.val('full');

$outputCapacity.addClass('fullCapacity');

disableInputFields();

}

resetInputFields();

}

function disableInputFields(){

$inputProduct.attr('disabled', true);

$inputPrice.attr('disabled', true);

$inputQuantity.attr('disabled', true);

$buttonSubmit.attr('disabled', true);

}

function resetInputFields() {

$inputProduct.val('');

$inputPrice.val(1);

$inputQuantity.val(1);

$buttonSubmit.attr('disabled', true);

}

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Online Shop</title>

<style>

.wrapper {

margin: auto;

width: 53%;

border: 3px solid rgb(26, 26, 26);

background-color: rgb(250, 250, 250);

padding: 10px;

}

#header {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 300%;

font-weight: bold;

color: rgb(26, 26, 26);

text-align: center;

padding: 10px;

}

li {

list-style-type: none;

}

.block {

display: inline-block;

margin-left: 165px;

}

.field {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 25px;

color: rgb(26, 26, 26);

}

.custom-select {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

background-color: rgb(255, 255, 255);

color: rgb(26, 26, 26);

border: 1px solid rgb(26, 26, 26);

padding: 2px;

margin-top: 10px;

width: 200px;

}

.input1 {

width: 40px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

border: 1px solid rgb(26, 26, 26);

background-color: #ffffff;

color: rgb(26, 26, 26);

padding: 2px;

}

#capacity {

width: 40px;

font-size: 20px;

padding: 2px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

}

#sum {

width: 80px;

font-size: 20px;

padding: 2px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

margin: 10px 10px;

}

.fullCapacity {

width: 40px;

font-size: 20px;

padding: 2px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

color: rgb(250, 0, 0);

border: 1px solid #f07777b2;

background-color: #f78989bb;

}

.display {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 15px;

border: 1px solid rgb(26, 26, 26);

padding: 2px;

background-color: #ffffff;

color: rgb(26, 26, 26);

margin-top: 5px;

width: 100%;

height: 300px;

}

.text {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

color: rgb(26, 26, 26);

margin-right: 10px;

}

.button:enabled {

font-family: "Trebuchet MS", Helvetica, sans-serif;

font-size: 20px;

cursor: pointer;

border: 1px solid rgb(26, 26, 26);

padding: 5px 15px;

margin: 3px;

border-radius: 8px;

}

.button:disabled {

background: rgb(248, 248, 248);

cursor: pointer;

color: rgb(216, 216, 216);

font-size: 20px;

padding: 5px 15px;

font-family: "Trebuchet MS", Helvetica, sans-serif;

margin: 3px;

border: 1px solid rgb(26, 26, 26);

border-radius: 8px;

}

</style>

<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>

</head>

<body>

<div class="wrapper">

</div>

<script src="solution.js"></script>

<script>onlineShop('.wrapper')</script>

</body>

function onlineShop(selector) {

let form = `<div id="header">Online Shop Inventory</div>

<div class="block">

<label class="field">Product details:</label>

<br>

<input placeholder="Enter product" class="custom-select">

<input class="input1" id="price" type="number" min="1" max="999999" value="1"><label class="text">BGN</label>

<input class="input1" id="quantity" type="number" min="1" value="1"><label class="text">Qty.</label>

<button id="submit" class="button" disabled>Submit</button>

<br><br>

<label class="field">Inventory:</label>

<br>

<ul class="display">

</ul>

<br>

<label class="field">Capacity:</label><input id="capacity" readonly>

<label class="field">(maximum capacity is 150 items.)</label>

<br>

<label class="field">Price:</label><input id="sum" readonly>

<label class="field">BGN</label>

</div>`;

$(selector).html(form);

// Write your code here

let $product = $('.custom-select');

let $price = $('#price');

let $totalPrice = $('#sum');

let $quantity = $('#quantity');

let $button = $('#submit');

let $capacity = $('#capacity');

let $inventoryList = $('.display');

$product.on('input', () => {

let isEmpty = $product.val() === ''

$button.attr('disabled', isEmpty);

});

$button.on('click', onButtonClick);

function onButtonClick() {

let currentCapacity = Number($capacity.val());

let currentQuantity = Number($quantity.val());

let totalCapacity = currentCapacity + currentQuantity;

if (totalCapacity < 150) {

addProductToInventory();

$capacity.val(totalCapacity);

$totalPrice.val(

Number($totalPrice.val()) + Number($price.val()));

}

else {

disableInterface();

}

reset();

}

function addProductToInventory() {

$li = $(`<li>Product: ${$product.val()} Price: ${$price.val()} Quantity: ${$quantity.val()}</li>`);

$inventoryList.append($li);

}

function reset() {

$product.val('');

$price.val(1);

$quantity.val(1);

$button.attr('disabled', true);

}

function disableInterface() {

$capacity.val('full');

$capacity.addClass('fullCapacity');

$product.attr('disabled', true);

$price.attr('disabled', true);

$quantity.attr('disabled', true);

$button.attr('disabled', true);

}

}

**//ExamRetakeJSAdvanced30VIII2018-Vacantioner**

function makeReservation(selector) {

let Vacationer = user().User;

let btnManager = buttonManager($('button#submit'), $('button#edit'), $('button#continue'));

let container = $(selector);

let input = $('div.inputLabel > input');

let registration;

btnManager.submitBtn.on('click', function () {

if (input.eq(0).val().trim() && input.eq(1).val().trim()) {

registration = new Vacationer(input.toArray().map(i => i.value));

$('ul#infoPreview')

.append(`<li>Name: ${registration.fullName}</li>`)

.append(`<li>E-mail: ${registration.email}</li>`)

.append(`<li>Phone: ${registration.phoneNumber}</li>`)

.append(`<li>Address: ${registration.address}</li>`)

.append(`<li>Postal Code: ${registration.postalCode}</li>`);

input.val('');

btnManager.disableSubmit();

btnManager.enableEdit();

btnManager.enableContinue();

}

});

btnManager.editBtn.on('click', function () {

$('ul#infoPreview').empty();

registration.allProperties.forEach((p, index) => input.eq(index).val(p));

btnManager.enableSubmit();

btnManager.disableContinue();

btnManager.disableEdit();

});

btnManager.continueBtn.on('click', function () {

container

.append(`<h2>Payment details</h2>

<select id="paymentOptions" class="custom-select">

<option selected disabled hidden>Choose</option>

<option value="creditCard">Credit Card</option>

<option value="bankTransfer">Bank Transfer</option>

</select>

<div id="extraDetails"></div>`);

$('#paymentOptions').change('click', function () {

let extraDetails = $('div#extraDetails');

let choice = $('#paymentOptions option:selected');

extraDetails.empty();

if (choice.text() === 'Credit Card') {

extraDetails

.append(`<div class="inputLabel">Card Number<input></div><br>`)

.append(`<div class="inputLabel">Expiration Date<input></div><br>`)

.append(`<div class="inputLabel">Security Numbers<input></div><br>`)

.append(`<button id="checkOut">Check Out</button>`)

} else if (choice.text() === 'Bank Transfer') {

extraDetails

.append(`<p>You have 48 hours to transfer the amount to:<br>IBAN: GR96 0810 0010 0000 0123 4567 890</p>`)

.append(`<button id="checkOut">Check Out</button>`)

}

$('#checkOut').on('click', function () {

let wrapper = $('#wrapper');

wrapper.empty();

wrapper.append(`<h4>Thank you for your reservation!</h4>`)

});

});

btnManager.disableSubmit();

btnManager.disableContinue();

btnManager.disableEdit();

});

function user() {

class User {

constructor([fullName, email, phoneNumber, address, postalCode]) {

this.fullName = fullName;

this.email = email;

this.phoneNumber = phoneNumber;

this.address = address;

this.postalCode = postalCode;

}

get allProperties() {

return [this.fullName, this.email, this.phoneNumber, this.address, this.postalCode];

}

}

return {User};

}

function buttonManager(submitSelector, editSelector, continueSelector) {

let submitBtn = $(submitSelector);

let editBtn = $(editSelector);

let continueBtn = $(continueSelector);

return {

disableSubmit: () => submitBtn.attr('disabled', true),

enableSubmit: () => submitBtn.attr('disabled', false),

disableEdit: () => editBtn.attr('disabled', true),

enableEdit: () => editBtn.attr('disabled', false),

disableContinue: () => continueBtn.attr('disabled', true),

enableContinue: () => continueBtn.attr('disabled', false),

submitBtn,

editBtn,

continueBtn,

}

}

}