Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования «Тульский государственный университет»

Институт прикладной математики и компьютерных наук

Кафедра «Информационная безопасность»

Отчет по лабораторным работам 7-8

*по курсу*

СЕТЕВОЕ ПРОГРАММИРОВАНИЕ

Выполнил: студенты группы 221061-ПБ Булычева Е.А. \_\_\_\_\_\_\_\_\_\_\_

Проверил: асс. каф. ИБ Поляничко К.В. \_\_\_\_\_\_\_\_\_\_\_

Тула, 2019 г.

**ХОД РАБОТЫ**

1. Написать функцию, преобразующую число в объект. Передавая на вход число от 0 до 999, надо получить на выходе объект, в котором в соответствующих свойствах описаны единицы, десятки и сотни. Например, для числа 245 надо получить следующий объект: {‘единицы’: 5, ‘десятки’: 4, ‘сотни’: 2}. Если число превышает 999, необходимо выдать соответствующее сообщение с помощью console.log и вернуть пустой объект.

function getNum( num ) {

var obj = {};

if ( num < 1 || num > 999 || isNaN( num )) {

console.log( 'Ошибка. Введите число от 1 до 999' );

return obj;

}

else {

obj[ 'едининцы' ] = num % 10;

obj[ 'десятки' ] = ( num - obj[ 'едининцы' ]) % 100 / 10;

obj[ 'сотни' ] = ( num - obj[ 'едининцы' ] - obj[ 'десятки' ] \* 10 ) % 1000 / 100;

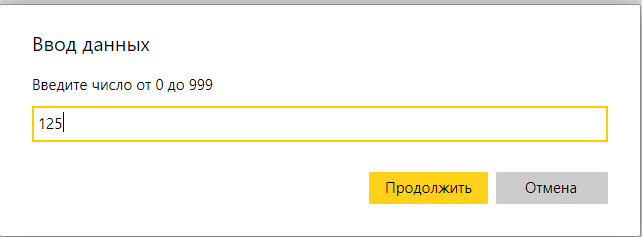
return obj;

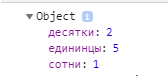
}

}

var a = prompt('Введите число от 0 до 999');

console.log( getNum(a));





1. Продолжить работу с интернет-магазином, а именно, перенести функционал подсчета корзины на объектно-ориентированную базу:

• создать объект Basket с полем goodList и методами countTotalPrice, countTotalNumber и putProduct (методы должны считать общую стоимость того, что в goodList, общее количество того, что в goodList и добавлять товар в корзину)

• создать массив товаров с полями name и price

• после этого пройтись по нему в цикле и положить в корзину по одному товару на четной позиции и по 2 товара на нечетной позиции

• внутри корзины в массиве goodList у каждого товара должно быть поле

count – количество товаров этого вида в корзине (например, 5 красных шаров)

• вывести общую стоимость корзины и общее количество товаров в корзине

let basket = {

goodList: [],

countTotalPrice() {

let totalPrice = 0;

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

let goodsItem = this.goodList[i];

totalPrice += goodsItem.price \* goodsItem.count;

}

return totalPrice;

},

countTotalNumber() {

let totalCount = 0;

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

let goodsItem = this.goodList[i];

totalCount += goodsItem.count;

}

return totalCount;

},

putProduct(product, count) {

let goodsItem = {

product,

price: product.price,

count

};

this.goodList.push(goodsItem);

}

};

let products = [];

for (let i = 1; i < (parseInt(Math.random() \* 10) + 2); i++) {

let product = {

name: 'name' + i,

price: parseInt(Math.random() \* 10000) / 100

};

products.push(product);

}

console.log(products);

for (let i = 1; i <= products.length; i++) {

// положить в корзину по одному товару на четной позиции и по 2 товара на нечетной позиции

let count = i % 2 ? 2 : 1;

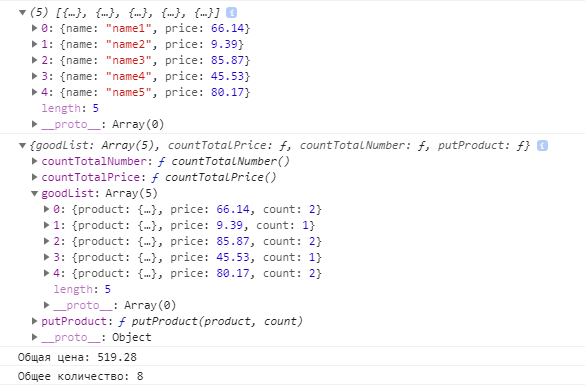
basket.putProduct(products[i - 1], count);

}

console.log(basket);

console.log('Общая цена: ' + basket.countTotalPrice().toFixed(2));

console.log('Общее количество: ' + basket.countTotalNumber());



1. Создать функцию, генерирующую шахматную доску. Можно использовать любые html-теги. Доска должна быть верно разлинована на черные и белые ячейки. Строки должны нумероваться числами от 1 до 8, столбцы — латинскими буквами A, B, C, D, E, F, G, H.

function chessboard() {

var newTable = document.createElement( 'table' ),

lets = [ '','A','B','C','D','E','F','G','H','' ],

blackFigs1 = [ '8','','','','','','','','','8' ],

whiteFigs1 = [ '1','','','','','','','','','1' ],

blackFigs2 = [ '7','','','','','','','','','7' ],

whiteFigs2 = [ '2','','','','','','','','','2' ];

for ( var i = 0, a = 9; i < 10, a >= 0; i++, a-- ) {

var newTr = newTable.insertRow(i);

for ( var j = 0; j < 10; j++ ) {

var newTd = newTr.insertCell( j );

switch (i) {

case 0:

newTd.innerText = lets[ j ];

break;

case 1:

newTd.innerHTML = blackFigs1[ j ];

break;

case 2:

newTd.innerHTML = blackFigs2[ j ];

break;

case 7:

newTd.innerHTML = whiteFigs2[ j ];

break;

case 8:

newTd.innerHTML = whiteFigs1[ j ];

break;

case 9:

newTd.innerText = lets[ j ];

break;

default:

if ( j === 0 || j === 9 ) {

newTd.innerHTML = a;

}

break;

}

}

}

var elem = document.getElementsByClassName( 'chess' );

document.body.appendChild( newTable );

};

chessboard();



1. Сделать генерацию корзины динамической: верстка корзины не должна находиться в HTML-структуре. Там должен быть только div, в который будет вставляться корзина, сгенерированная на базе JS:

a. Пустая корзина должна выводить строку «Корзина пуста»;

b. Наполненная должна выводить «В корзине: n товаров на сумму m рублей».

Продолжаем реализовывать модуль корзины:

• Добавлять в объект корзины выбранные товары по клику на кнопке «Купить» без перезагрузки страницы;

• Привязать к событию покупки товара пересчет корзины и обновление ее внешнего вида.

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Супер магазин 123</title>

<link rel="stylesheet" href="styleshop.css">

</head>

<body>

<header class="header">

<img src="zowie2.jpg" align="left">

<h1>ZOWIE</h1>

</header>

<div class="container">

<h2>ВЫБЕРИТЕ ВАШ ПРОДУКТ</h2>

<div class="products">

<div class="product">

<div>

<h3>Мышь EC1-A</h3>

<img src="https://zowie.benq.com/content/dam/game/en/product/mouse/ec1-a-white/gallery/ec1-a-white-top.png" />

<h4>в белом цвете</h4>

</div>

<div>

<d class="p11">Эргономичный дизайн специально для правшей. Он предлагает пользователям больше места для движения запястья. Правая сторона мыши имеет округлую форму. Это позволяет перемещать мышь, удерживая её кончиками пальцев. Глянцевые версии предназначены для геймеров, которым нравится ощущение такого покрытия. Серия EC поставляется в двух размерах для удовлетворения различных потребностей.</d>

</div>

<div>

<p data-price="4490"><b>Цена: 4 490руб</b></p>

<button class="add-to-basket-button">Добавить в корзину</button>

</div>

</div>

<div class="product">

<div>

<h3>Мышь FK1+</h3>

<img src="https://zowie.benq.com/content/dam/game/en/product/mouse/fk1plus-white/gallery/zowie-fk1+-white-top.png" />

<h4>в белом цвете</h4>

</div>

<div>

<d class="p11">Вне зависимости от того, правша вы или левша, серии FK созданы для вашего комфорта. Низкопрофильная конструкция обеспечивает наиточнейший контроль для пользователей как с ладонным, так и с когтевым хватом. Глянцевые версии предназначены для геймеров, которым нравится ощущение такого покрытия. Серия FK представлена в трех разных размерах.</d>

</div>

<div>

<p data-price="4990"><b>Цена: 4 990руб</b></p>

<button class="add-to-basket-button">Добавить в корзину</button>

</div>

</div>

<div class="product">

<div>

<h3>Мышь ZA11</h3>

<img src="https://zowie.benq.com/content/dam/game/en/product/mouse/za11-white/gallery/zowie-za11-white-top.png" />

<h4>в белом цвете</h4>

</div>

<div>

<d class="p11">Удобный дизайн как для правшей, так и для левшей. Удобная конструкция идеально ложится под ладонь пользователя. Глянцевые версии предназначены для геймеров, которым нравится ощущение такого покрытия. Серия ZA представлена в трех разных размерах.</d>

</div>

<div>

<p data-price="4860"><b>Цена: 4 860руб</b></p>

<button class="add-to-basket-button">Добавить в корзину</button>

</div>

</div>

<div class="product">

<div>

<h3>Мышь EC1-B CS: GO</h3>

<img src="https://zowie.benq.com/content/dam/game/en/product/mouse/ec1-b-csgo/gallery/ec1-b-csgo-01.png" />

</div>

<div>

<d class="p11">Эргономичный дизайн специально для правшей. Он предлагает пользователям больше места для движения запястья. Правая сторона мыши имеет округлую форму. Это позволяет перемещать мышь, удерживая её кончиками пальцев. Серия EC-B с сенсором 3360 поставляется в двух размерах для удовлетворения разных потребностей геймеров.</d>

</div>

<div>

<p data-price="5200"> <b>Цена: 5 200руб</b></p>

<button class="add-to-basket-button">Добавить в корзину</button>

</div>

</div>

</div>

</div>

<div id="basket"></div>

<script src = 'scriptshop.js'></script>

</body>

</html>

**script.js**

const basketElement = document.getElementById('basket')

function renderBasket(basek) {

const totalPrice = basket.countTotalPrice()

const totalCount = basket.countTotalCount()

return `

<h1>Ваша корзина</h1>

<p>Кол-во товаров: ${totalCount}</p>

<p>Общая цена: ${totalPrice}</p>

<h2>Товары:</h2>

${basket.products.map(product => {

return `<p>Товар: ${product.name}, кол-во: ${product.count}, цена: ${product.price \* product.count}</p>`

}).join('<br>')}

`

}

class Basket {

constructor(){

this.products = []

}

addToBasket(productToAdd) {

if (this.products.find(product => product.name === productToAdd.name)) {

this.products.find(product => product.name === productToAdd.name).count += productToAdd.count

} else {

this.products.push(productToAdd)

}

this.renderToElem()

}

countTotalPrice() {

let totalPrice = 0

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

let product = this.products[i]

totalPrice += product.price \* product.count

}

return totalPrice

}

countTotalCount() {

let totalCount = 0

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

let product = this.products[i]

totalCount += product.count

}

return totalCount

}

renderToElem() {

basketElement.innerHTML = renderBasket(this)

}

}

const basket = new Basket()

basket.renderToElem()

Array.from(document.querySelectorAll('.add-to-basket-button')).forEach(button => {

button.onclick = () => {

const product = button.parentElement.parentElement

const price = +product.querySelector("p").getAttribute("data-price")

const name = product.querySelector("h3").innerText

basket.addToBasket({

price,

count: 1,

name,

})

}

})

**style.css**

h1{

font-family: CURSIVE;

}

h2,h3,h4 {

font-family:"Montserrat","微軟正黑體",Arial, sans-serif, 'icomoon';

text-align: left;

}

/\*p{

font-weight:bold;

}\*/

d.p11{

font-family: CORBEL;

/\*text-align:justify; \*/

}

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

color: #212121;

font-family: sans-serif;

background-color: #FAFAFA;

}

.container {

padding: 8px;

width: 100%;

max-width: 1000px;

margin: 0 auto;

min-height: 800px;

}

#basket {

position: fixed;

bottom: 0;

left: 0;

background-color: #1b1b1bde;

width: 370px;

min-height: 100px;

color: white;

padding: 8px;

}

.header {

padding: 20px;

background-color: #080808;

color: white;

}

.products {

display: flex;

flex-flow: row nowrap;

}

.product {

display: flex;

flex-flow: column nowrap;

justify-content: space-between;

width: 30%;

padding: 8px;

height: 600px;

background-color: white;

margin-right: 15px;

box-shadow: 0 1px 3px rgba(0,0,0,0.12);

}

.product img {

width: 100%;

}

.add-to-basket-button {

width: 100%;

color: white;

padding: 10px;

background-color: #424242;

/\*#6A1B9A;\*/

border-radius: 8px;

border: none;

margin-top: 5px;

text-transform: uppercase;

box-shadow: 0 1px 3px rgba(0,0,0,0.62);

cursor: pointer;

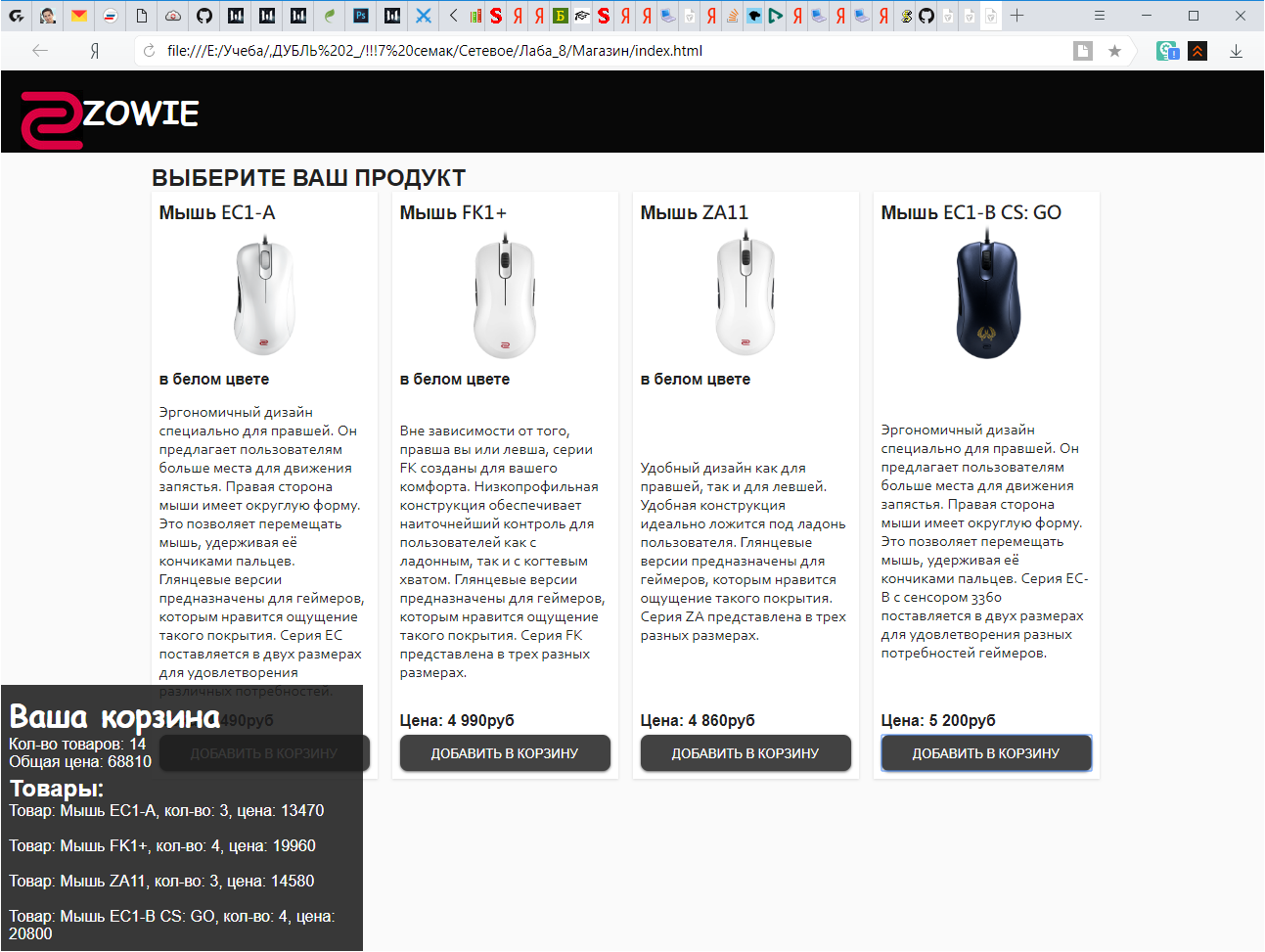
font-family:"Montserrat","微軟正黑體",Arial, sans-serif, 'icomoon';

}

.add-to-basket-button:hover {

background-color: #b71c1c;/\*#8E24AA;\*/

}



1. Создать форму в html со следующими полями:

• Логин - текстовое поле

• Пароль - поле типа password

• Повтор пароля - поле типа password

• Кнопка отправить форму

Необходимо реализовать валидацию этой формы по следующим правилам:

• Логин - должно содержать как минимум 1 символ, не более 50 символов.

• Пароль - минимум 5 символов, максимум 50.

• Повтор пароля - значение должно совпадать с полем пароль.

•Кнопка отправить форму - при нажатии на кнопку форма должна провериться, при прохождении проверки, форма отправляется, если проверка не была пройдена, то:

* + - Каждое поле, которое не прошло проверку должно выделяться красным цветом.
    - Под каждым полем, что не прошло проверку, должна писаться подсказка по какой причине проверка провалилась.

Пользоваться аттрибутами HTML5 запрещено, необходимо реализовать проверки с помощью javascript. Можно пользоваться сторонними стилевыми библиотеками (bulma, bootstrap).

**IndexPASS.html**

<meta charset='utf-8'>

<html>

Логин

<input type = "text" id="login">

Пароль

<input type = "password" id="password">

Повтор пароля

<input type = "password" id="password-confirm">

<button>Отправить</button>

<div id="errors" style="color: black"></div>

</html>

<script src = 'scriptPASS.js'></script>

**scriptPASS.js**

const loginInput = document.getElementById('login')

const passwordInput = document.getElementById('password')

const passwordConfirmInput = document.getElementById('password-confirm')

const errorsElement = document.getElementById('errors')

loginInput.addEventListener('input', validate)

passwordInput.addEventListener('input', validate)

passwordConfirmInput.addEventListener('input', validate)

loginInput.onchange = validate

passwordInput.onchange = validate

passwordConfirmInput.onchange = validate

function getErrors() {

if (loginInput.value.length === 0) {

loginInput.style.borderColor = 'red'

return 'Не заполнено поле login'

}

if (loginInput.value.length > 50) {

loginInput.style.borderColor = 'red'

return 'Логин не должен превышать 50 символов'

}

if (passwordInput.value.length < 5) {

passwordInput.style.borderColor = 'red'

return 'Пароль не может быть меньше 5 символов'

}

if (passwordInput.valuelength > 50) {

passwordInput.style.borderColor = 'red'

return 'Пароль не должен превышать 50 символов'

}

if(passwordInput.value !== passwordConfirmInput.value) {

passwordInput.style.borderColor = 'red'

passwordConfirmInput.style.borderColor = 'red'

return 'Поля "пароль" и "подтверждение пароля" не совпадают'

}

loginInput.style.borderColor = ''

passwordInput.style.borderColor = ''

passwordConfirmInput.style.borderColor = ''

return false

}

function validate() {

const errors = getErrors()

if (errors) {

errorsElement.innerText = errors

return

}

errorsElement.innerText = ''

}

**style.css**

table {

font-family: Impact, sans-serif;

font-size: 30px;

background-color: white;

/\* color: #968e8e;\*/

color: black;

}

td {

height: 50px;

width: 50px;

text-align: center;

vertical-align: middle;

}

/\* все четные номера строк, нечетные номера столбцов\*/

tr:nth-child(even) td:nth-child(odd) {

background-color: black;

}

tr:nth-child(odd) td:nth-child(even) {

background-color: black;

}

/\*последний элемент родителя, нечетные номера столбцов \*/

tr:last-child td:nth-child(odd),

/\*первый элемент родителя, четные номера столбцов \*/

tr:first-child td:nth-child(even){

background-color: white;

}

tr:nth-child(even) td:first-child{

background-color: white;

}

tr:nth-child(odd) td:last-child{

background-color: white;

}







# ВЫВОД

# При выполнении данной лабораторной работы были изучены навыки программирования на языке JavaScript, изучены операторы, циклы, массивы, а также написаны демонстрационные программы для закрепления материала.