Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Распределенные системы обработки информации

Контрольная работа №1

по теме

«Объекты JavaScript»

|  |  |
| --- | --- |
| Выполнил студент 5-го курса  группы 694051 специальности  «Электронный маркетинг» | Кузьмич Павел  Валерьевич |
| Проверил старший преподаватель  кафедры экономической  информатики | Атрощенко Натэлла  Александровна |

Минск, 2020

Содержание

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 3](#__RefHeading___Toc233_1502732707)

[1.1. Понятие Cookie. Создание cookies в JavaScript Dynamic Invocation Interface 3](#__RefHeading___Toc787_2586772390)

[1.2. Возможности динамической подгрузки данных на веб-страницу в JavaScript 4](#__RefHeading___Toc789_2586772390)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 5](#__RefHeading___Toc791_2586772390)

[2.1. Описание HTML кода страницы 5](#__RefHeading___Toc793_2586772390)

[2.2. Описание CSS кода каскадных таблиц стилей 7](#__RefHeading___Toc235_1502732707)

[2.3. Описание функциональной части 9](#__RefHeading___Toc795_2586772390)

[ВЫВОДЫ 17](#__RefHeading___Toc797_2586772390)

# 

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1. Понятие Cookie. Создание cookies в JavaScript Dynamic Invocation Interface

Куки (cookie) — это небольшие файлы или фрагменты данных, которые отправляются веб-сервером и хранятся на компьютере пользователя. Веб-клиентом обычно выступает веб-браузер. При необходимости открыть соответствующую страницу какого-либо сайта он пересылает куки веб-серверу в виде HTTP-запроса.

Как правило, куки используются для аутентификации пользователей, отслеживания сеансов доступа пользователей, хранения индивидуальных предпочтений и настроек, ведения статистики о клиентах.

Они служат для того, чтобы «узнавать» (аутентифицировать) пользователя, который уже посещал раньше сайт и выводить персонализированные страницы, что соответствуют его предпочтениям. В куки можно найти текстовую информацию, которая необходима для правильной работы веб-ресурса (к примеру, логин и пароль клиента). В таком случае при переходе со страницы на страницу сайт не будет требовать ввести ваши данные повторно.

Куки (cookie) активно используются в интернет-магазинах. Благодаря им, если пользователь закрыл страницу сайта или браузер, после следующего входа его корзина и просмотренные товары сохранятся.

Некоторые типы куки-файлов используются для того, чтобы пользователь мог осуществлять быстрый просмотр содержимого сайта или использовать некоторые его функции. Например, получить доступ к некоторым разделам можно только после регистрации. На сегодняшний день выделяют следующие типы cookie:

* сессионные (временные) — они могут существовать только во временной памяти;
* постоянные — удаляются не сразу после закрытия окна браузера, а через определённое время (они передаются на сервер каждый раз, когда пользователь переходит на сайт);
* защищённые — передаются только по специальным шифрованным соединениям (HTTPS);
* HttpOnly — к этому файлу нельзя обращаться через API.

JavaScript может создавать, читать и удалять файлы cookie с помощью свойства *Document.cookie* . С помощью JavaScript можно создать файл cookie так: *document.cookie = "username=John Doe"*.

Можно также добавить дату окончания срока действия (в формате UTC). По умолчанию файл cookie удаляется при закрытии обозревателя: *document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC"*.

С помощью параметра path можно указать обозревателю, к какому пути принадлежит файл cookie. По умолчанию файл cookie принадлежит текущей странице. *document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/"*.

С помощью JavaScript файлы cookie можно читать следующим образом:

*var x = document.cookie*. Document. cookie будет возвращать все куки в одной строке так же, как: кукие1 = value; кукие2 = значение; кукие3 = значение.

С помощью JavaScript можно изменить файл cookie так же, как и его создание: *document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/"*. Старый файл cookie перезаписывается.

## 1.2. Возможности динамической подгрузки данных на веб-страницу в JavaScript

Любое веб-приложение можно логически поделить на две составляющие — на клиентскую часть и серверную часть. К клиентской части относятся сам браузер и скрипты, которые он выполняет, к серверной — набор скриптов, которые генерируют ответ на любой запрос пользователя.  
 Жизнь любой страницы начинается с запроса от клиента к серверу. Ответом будет код страницы, содержащий, помимо структуры и стилей, логику клиентской части.

После получения страницы с сервера, браузер отображает её и запускает на выполнение приложенные к ней скрипты.  
Клиентская часть реагирует на различные события — например, на клик по некоторому элементу, перемещение мыши или на истечение таймера. Для того, чтобы получить какие-то данные с сервера (или отправить что-то на него), используются дополнительные, обычно асинхронные, запросы.  
 Для того, чтобы обновить структуру страницы, скрипту клиента необходимо знать, что необходимо убрать, что и куда необходимо добавить, что на что заменить. Существуют разные варианты, как организовать такие обновления.

Дублирование — логику отображения знает и клиентская, и серверная часть. В таком случае, ответы на регулярные запросы со стороны клиента могут содержать исключительно данные — изменения в модели. При получении такого ответа клиентская часть «оборачивает» данные в HTML-теги, добавляет необходимые тексты и обновляет структуру страницы. Серверу же знания об отображении нужны только для того, чтобы сгенерировать изначальную версию страницы.

Плюсы: малый объем трафика. К минусам можно отнести необходимость дублировать код.

Другой вариант — сервер может не отправлять весь html компонента, а присылать только «дельту» — изменения, которые необходимо внести. Теперь клиент определяет элемент, который будет изменять, и то, как он его будет изменять, непосредственно из ответа сервера.

Плюсы: отсутствие дублирования кода. К минусам можно отнести достаточно большой объем сетевого трафика, сложность вычисления и записи дельты в случае нетривиальных изменений, общее усложнение клиентской и серверной части.

Следующий вариант — можно переложить всю ответственность за генерацию HTML на клиента с использованием JavaScript. В таком случае сервер будет только предоставлять данные, необходимые для отображения. Ответы, как и в первом варианте, будут содержать только данные. Главное отличие от предыдущих вариантов заключается в том, что сервер не выполняет первоначальную генерацию страницы, её сборка осуществляется уже браузером клиента. Он может пригодиться, если необходимо уменьшить нагрузку на сервер.

Плюсы подхода: малый объем трафика, уменьшение нагрузки на сервер. К минусам можно отнести высокую нагрузку на компьютер пользователя, возможную избыточность — часть знаний клиентской части об отображении может так и остаться невостребованной, если какие-то события не наступят.

Каждый из рассмотренных методов имеет право на жизнь, и может быть использован в проектах разной сложности.

# 2 ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Описание HTML кода страницы

Основным элементов страницы является разбитая на секции (тег <section>) форма ввода данных и вывода результатов их обработки. Форма состоит из нескольких полей ввода данных о компании (названия, адреса, телефона, продукта), которые заданы тегами типа <input type=«text»>, а также содержит кнопки для добавления, обновления и вывода данных (<input type=«button»>). После тега страницы задан тег <script src="script.js"> для инициализации функционала из JavaScript файла. Листинг кода приведён ниже.

Листинг 1.1 Структура HTML страницы.

<html>

<head>

<meta charset="utf-8">

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

<title>JS objects</title>

</head>

<body>

<div class="box">

<h1>company info</h1>

Листинг 1.1 Продолжение

<section>

<div class="input-box" style="margin-right: 20px;">

<label for="name">Name</label>

<input id="name" type="text" value="" name="name" placeholder="enter..." onclick="this.value=''">

</div>

<div class="input-box">

<label for="country">Country</label>

<input id="country" type="text" value="" name="country" placeholder="enter..." onclick="this.value=''">

</div>

<br style="clear:both;" />

</section>

<section>

<div class="input-box" style="margin-right: 20px;">

<label for="address">Address</label>

<input id="address" type="text" value="" name="address" placeholder="enter..." onclick="this.value=''">

<label for="phone" style="padding-top: 8px;">Phone</label>

<input id="phone" type="text" value="" name="phone" placeholder="enter..." onclick="this.value=''">

</div>

<div class="input-box">

<label for="product-name">Product</label>

<input id="product-name" type="text" value="" name="product- name" placeholder="name" onclick="this.value=''">

<input id="product-color" type="text" value="" name="product- color" placeholder="color" onclick="this.value=''">

<input id="product-price" type="number" value="" name="product- price" placeholder="price" onclick="this.value=''">

</div>

</section>

<section>

<div class="input-box" style="margin-right: 15px;">

Select<br>company

</div>

<div class="input-box">

<select id="show-company-selector" class="select-box">

<option>-----None-----</option>

</select>

</div>

</section>

<section>

Листинг 1.1 Продолжение

<div style="float: left; margin-bottom: 20px;">

</div>

</section>

<section>

<div style="float: left; margin-right: 10px;">

<input type="button" id="add-button" class="button" value="Add">

</div>

<div style="float: left; margin-right: 10px;">

<input type="button" id="update-button" class="button" value="Update">

</div>

<div style="float: left; margin-right: 5px;">

<input type="button" id="show-button" class="button" value="Show">

</div>

<div style="float: left; margin-right: 10px;">

<select id="show-type-selector" class="select-box">

<option>as list</option>

<option>as text</option>

</select>

</div>

<div style="float: left;">

<input type="button" id="show-all-button" class="button" value="Show all">

</div>

</section>

<section>

<div class="result-box" id="result-box" style="margin-top: 30px;">

</div>

</section>

</div>

</body>

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

</html>

## 2.2. Описание CSS кода каскадных таблиц стилей

Для форматирования формы ввода, заголовков, а также полей ввода данных о компании используются каскадные таблицы стилей приведённые в листинге 2.1.

Листинг 2.1

body {

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

background-image: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);

}

h1 {

color: #ffffff;

font-family: 'Lato', sans-serif;

text-transform: uppercase;

font-size: 28px;

}

input,

label {

display: block;

}

.box {

display: flex;

flex-direction: column;

align-items: center;

padding: 40px 50px;

color: #fff;

background: rgba(0, 0, 0, 0.8);

border-radius: 10px;

box-shadow: 0 0.4px 0.4px rgba(128, 128, 128, 0.109), 0 12px 12px rgba(128, 128, 128, 0.35);

}

.input-box {

font-family: Verdana, Geneva, Tahoma, sans-serif;

color: tan;

float: left;

padding-bottom: 20px;

}

Для задания аттрибутов оформления кнопок и окна вывода результатов используются следующие CSS элементы, представленные в листинге 2.2.

Листинг 2.2

.result-box {

display: block;

width: 100%;

align-items: center;

Листинг 2.2. Продолжение

background: rgba(128, 128, 128, 0.109);

border-radius: 10px;

}

.button {

text-decoration: none;

float: left;

color: white;

font-size: 18px;

background: lightsteelblue;

padding: 10px 20px;

border-radius: 4px;

font-weight: normal;

text-shadow: 0.5px 1px black;

transition: all 0.2s ease-in-out;

}

.select-box {

position: relative;

display: block;

width: 100%;

height: 100%;

font-size: 18px;

color: #60666d;

}

## 2.3. Описание функциональной части

Функциональные особенности типа объекта Company содержаться в одноимённом классе. Данные класс содержит несколько конструкторов — один для инициализации объекта на основе списка параметров, другой — на основе JSON объекта. В любом случае задаются атрибуты объекта такие как companyName, country, address, phone, а также список продуктов, производимых компанией. Также в классе имеется сеттер метод setCompanyId для установления уникального идентификатора компании (далее используется как ключ для размещения в localStorage) на основе результата хэш-функции от имени. Дополнительно в классе прописаны get методы infoText() и infoList() для получения информации о компании в виде списка или общего текста соответсвенно. Метод addProduct(product) предназначен для добавления в список продуктов нового за счёт обновления аттрибута this.products. Описание данного класса представлено в листинге 3.1.

Листинг 3.1 Определение класса Company

class Company {

*// стандартный коструктор*

constructor(name, country, address, phone, product) {

this.companyName = name;

this.country = country;

this.address = address;

this.phone = phone;

if (typeof product != "undefined") {

this.addProduct(product);

} else {

this.products = []

}

}

*// коструктор на основе json объекта описывающего компанию*

static jsonConstructor(jsonCompanyObject) {

var product;

var companyInstance = new Company(

jsonCompanyObject["companyName"],

jsonCompanyObject["country"],

jsonCompanyObject["address"],

jsonCompanyObject["phone"]

)

for (product of jsonCompanyObject["products"]) {

companyInstance.addProduct(product);

}

return companyInstance;

}

*// метод для добавления id компании на основе результата хэш- функции от имени*

setCompanyId() {

this.companyId = hashCode(this.companyName);

}

*// статический метод для формирования json объекта товара*

static createProduct(name, color, price) {

return {

"name": name,

"color": color,

"price": price

};

}

Листинг 3.1

*// метод для добавления товара в список товаров компании*

addProduct(product) {

if (!this.products) {

this.products = [product];

} else {

this.products.push(product)

}

}

*// получение информации о компании в текстовом виде*

get infoText() {

var productsString = "";

for (var product of this.products) {

productsString = productsString + `${product["name"]} `;

}

return `Company "${this.companyName}" is based in ${this.country}, ${this.address}.<br>` + `Available products: ${productsString}`;

}

*// получение информации о компании в виде списка*

get infoList() {

var productsString = "";

var companyDataString = "";

for (var companyData of [this.companyName, this.country, this.address, this.phone]){

companyDataString = companyDataString + `<li>${companyData}</li>`;

}

for (var product of this.products) {

productsString = productsString + `<li>${product["name"]}, $ {product["color"]}, ${product["price"]}</li>`

}

return `<ul>${companyDataString + productsString}</ul>`;

}

}

Получение информации о полях, кнопках и их значений осуществляется с помощью метода getElementById объекта document (листинг 3.2).

Листинг 3.2

*// поля ввода информации о компании*

var nameField = document.getElementById('name');

var countryField = document.getElementById('country');

var addressField = document.getElementById('address');

var phoneField = document.getElementById('phone');

*// поля ввода информации о продукте*

var productName = document.getElementById('product-name');

var productColor = document.getElementById('product-color');

var productPrice = document.getElementById('product-price');

*// выпадающие селекторы компании и виды вывода результатов*

var showTypeSelector = document.getElementById('show-type-selector');

var showCompanySelector = document.getElementById('show-company- selector');

*// кнопки*

var addButton = document.getElementById('add-button');

var showButton = document.getElementById('show-button');

var updateButton = document.getElementById('update-button');

var showAllButton = document.getElementById('show-all-button');

var resultBox = document.getElementById('result-box');

Для добавления нового объекта типа Company на кнопку «add» добавлен обработчик события «click», где прописана функция, которая инициализируется новый объект компании, логирует информацию о нём в консоль, добавляет его в localStorage, а затем ещё и к выпадающему окну со списком доступных компаний. В завершение выполнения функции в блоке finally очищаются все поля для ввода информации. Листинг кода приведён ниже.

Листинг 3.3

addButton.addEventListener('click', function() {

const defaultStub = "empty";

try {

if (typeof(Storage) !== "undefined") {

name = nameField.value ? nameField.value : defaultStub;

country = countryField.value ? countryField.value : defaultStub;

address = addressField.value ? addressField.value : defaultStub;

phone = phoneField.value ? phoneField.value : defaultStub;

if (productPrice.value && productName.value) {

product = Company.createProduct(productName.value, productColor.value, productPrice.value);

} else {

product = "undefined"

}

var companyInstance = new Company(name, country, address, phone, product);

console.log("New Company object created: ", companyInstance)

companyInstance.setCompanyId();

localStorage.setItem(companyInstance.companyId, JSON.stringify(companyInstance));

var companyOption = document.createElement("option");

companyOption.id = companyInstance.companyId;

companyOption.innerHTML = companyInstance.companyName;

showCompanySelector.appendChild(companyOption)

} else {

console.log("ERROR: WebStorage not supported");

}

}

catch (error) {

console.log(error)

}

finally {

clearInputFields([nameField, countryField, addressField, phoneField, productName, productColor, productPrice])

}

});

Для обновления обновления информации о компании предусмотрен обработчик для кнопки «update». С помощью условного оператора if/else в зависимости от того, какое поле заполнено, обновляется тот или иной аттрибут. Листинг приведён ниже.

Листинг 3.4

updateButton.addEventListener('click', function() {

var selectedCompany = showCompanySelector.options[showCompanySelector.

options.selectedIndex];

try{

if (selectedCompany.id != "") {

var companyInstance = Company.jsonConstructor(JSON.parse(

localStorage.getItem(selectedCompany.id)));

if (nameField.value != "") {

companyInstance.companyName = nameField.value;

companyInstance.setCompanyId();

localStorage.removeItem(companyInstance.companyId);

selectedCompany.id = companyInstance.companyId;

}

if (countryField.value != "") {

companyInstance.country = countryField.value;

}

if (addressField.value != "") {

companyInstance.address = addressField.value;

}

if (phoneField.value != "") {

companyInstance.phone = phoneField.value;

}

if (productName.value != "" && productColor.value != "" && productPrice.value != ""){

companyInstance.addProduct(Company.createProduct(productName.value, productColor.value, productPrice.value))

} else {

resultBox.innerHTML = "<p>Please enter all product attributes to add it!</p>";

}

console.log("LOG", companyInstance);

companyInstance.setCompanyId();

localStorage.setItem(companyInstance.companyId, JSON.stringify(companyInstance));

}

}

catch (error) {

console.log(error)

}

finally {

clearInputFields([nameField, countryField, addressField, phoneField, productName, productColor, productPrice])

}

});

Для отображения информации об отдельной компании или всех доступных компаниях используются методы «show» и «show all» соответственно, для которых добавлены обработчики событий. Первый в зависимости от выбранной компании в селекторе выводит в окно результатов результат. Информация вычитывается из localStorage по заданному уникальному идентификатору. Второй выводит все компании в виде таблиц в новом окне. Описание кода приведено в листинге 3.5.

Листинг 3.5

showButton.addEventListener('click', function() {

resultBox.value = "";

const as\_text = "as text";

const as\_list = "as list";

var selectedCompany = s showCompanySelector.options[showCompanySelector.

options.selectedIndex];

if (selectedCompany.id != "") {

var companyInstance = Company.jsonConstructor(

JSON.parse(localStorage.getItem(selectedCompany.id)));

if (showTypeSelector.value == as\_text){

resultBox.innerHTML = companyInstance.infoText;

} else if (showTypeSelector.value == as\_list){

resultBox.innerHTML = companyInstance.infoList;

}

}

});

showAllButton.addEventListener('click', function() {

tablesArray = [];

var tableHeaderHTML = "<tr><th>Key</th><th>Value</th></tr>";

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

var companyInstance = Company.jsonConstructor(

JSON.parse(localStorage.getItem(localStorage.key(i))));

var tableContent = `

<tr><td>Name</td><td>${companyInstance.companyName}</td></tr>

<tr><td>Country</td><td>${companyInstance.country}</td></tr>

<tr><td>Address</td><td>${companyInstance.address}</td></tr>

<tr><td>Phone</td><td>${companyInstance.phone}</td></tr>`;

tablesArray.push(`${tableHeaderHTML}${tableContent}`);

}

var allCompaniesData = "";

for (var item of tablesArray) {

allCompaniesData = allCompaniesData + `<table border="1" style="table-layout: fixed; width: 25%;">${item}</table>`;

}

console.log("Results: ", allCompaniesData);

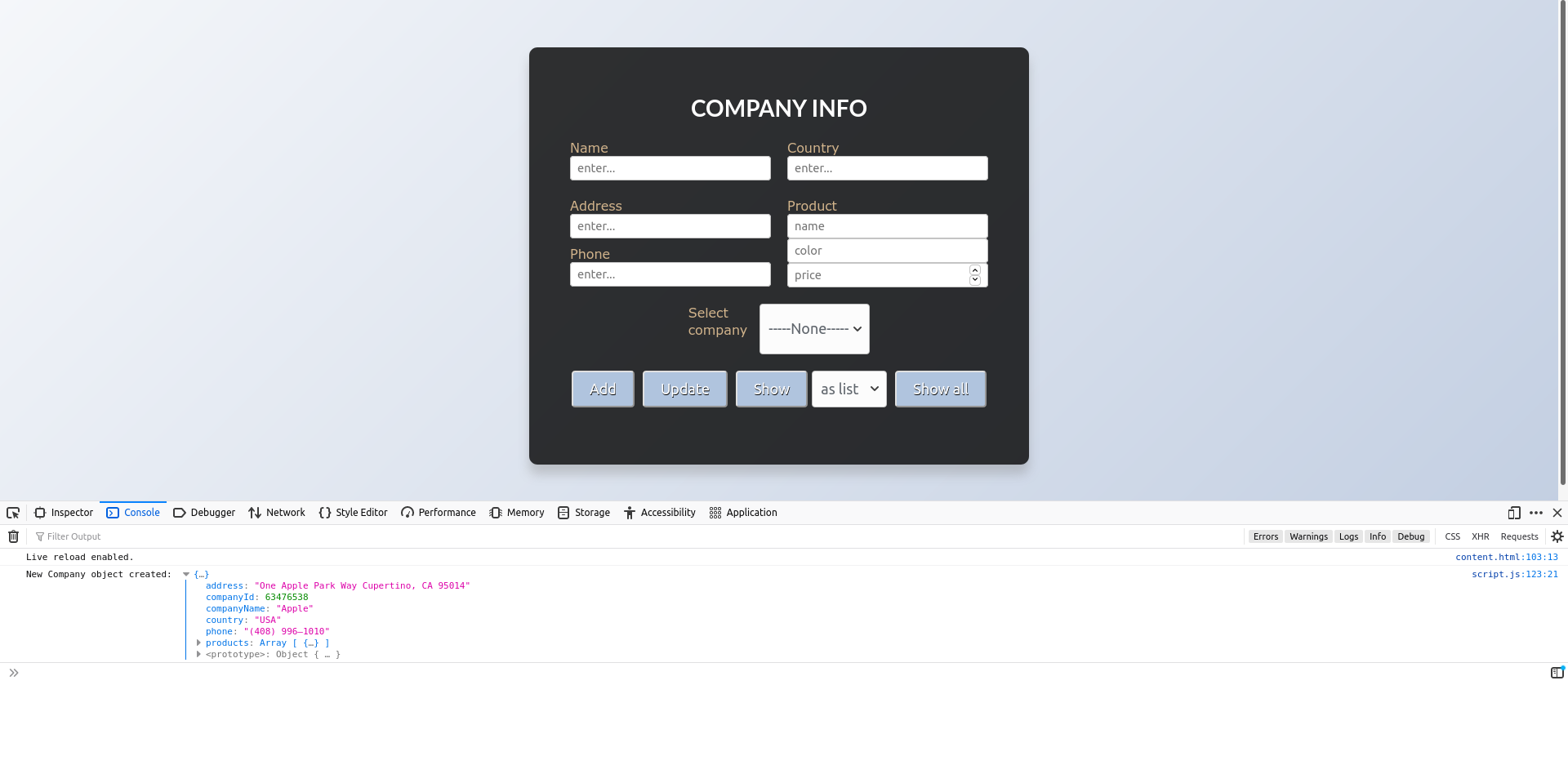
var newWindow = window.open("", "",);

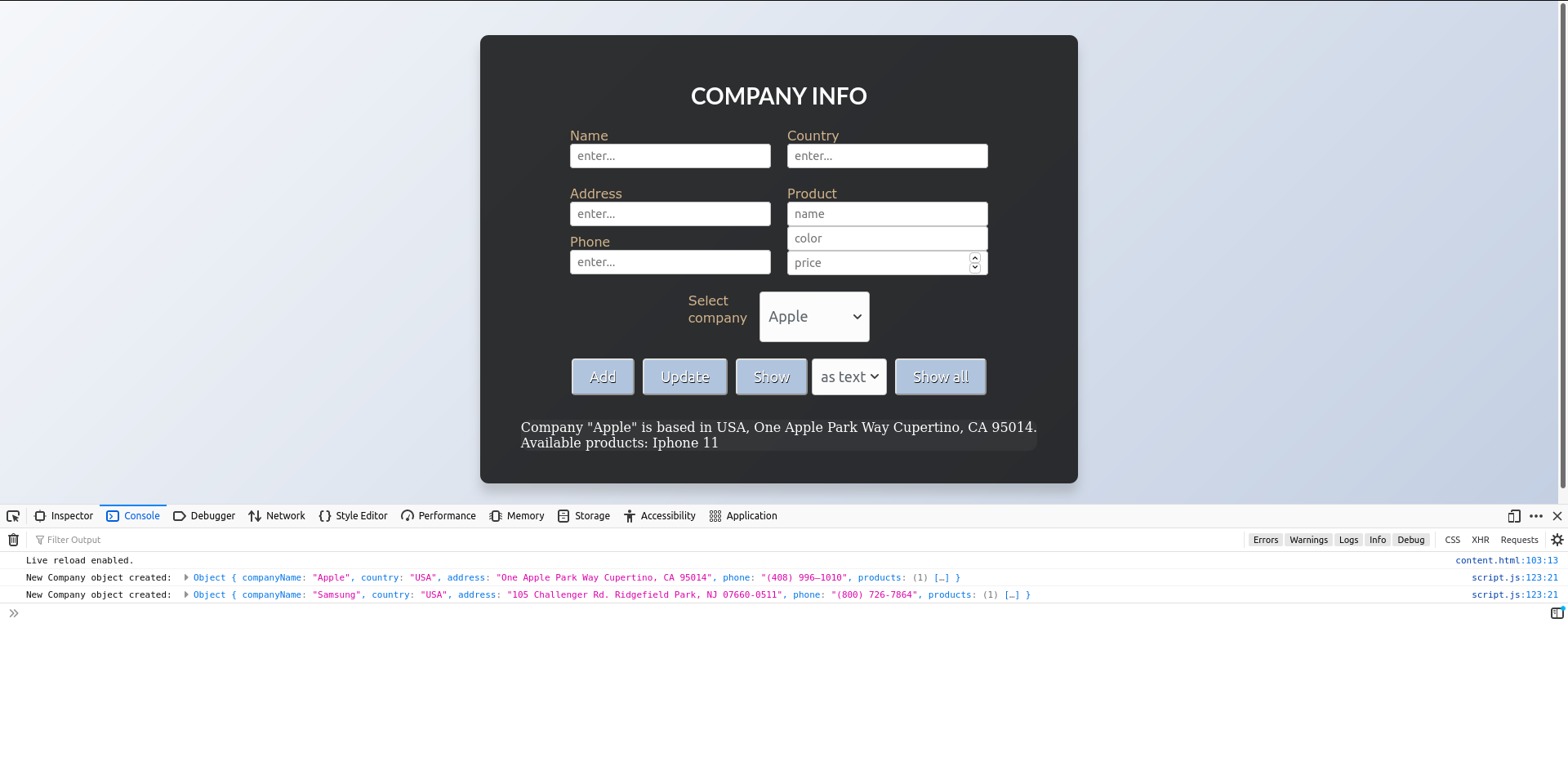
newWindow.document.write(allCompaniesData);

});

Скриншоты работы проекта приведены ниже на рисунках.

Рисунок 1 — Пустая форма ввода данных.

Рисунок 2 — Добавление объекта в localStorage.

Рисунок 3 — Отображение информации о компании.



# 

Рисунок 4 — Отображение всех данных.

# 

# ВЫВОДЫ

В результате выполнения контрольной работы была создана форма добавления, обновления и отображения информации о компания дистрибьюторах мобильной продукции. Хранение информации и объектов предусмотрена с использованием locaStorage. Динамические обновления контента обеспечено средствами JavaScript объектов.