Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт

по лабораторной работе №3

«*JavaScript*»

по дисциплине «Технологии интернет-программирования»

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент гр. 820602 | Н. В. Хаджинова |
| Шведов А.Р. |  |
|  |  |

Минск 2021

Содержание

[Введение 3](#_Toc88489842)

[1 Цель работы 4](#_Toc88489843)

[2 Теоретическая часть 5](#_Toc88489844)

[2.1 Введение в *DHTML* 5](#_Toc88489845)

[2.2 Основы языка *Javascript*, используемые в работе 6](#_Toc88489846)

[2.3 *HTML*-элемент *canvas* 11](#_Toc88489847)

[3 Практическая часть 13](#_Toc88489848)

[3.1 Обновление структуры проекта 13](#_Toc88489849)

[3.2 Руководство пользователя 14](#_Toc88489850)

[Заключение 27](#_Toc88489851)

[Список использованных источников 28](#_Toc88489852)

Введение

Для придания веб-страницам динамизма (например, выпадающие меню, анимации) используются языки написания скриптов. Стандартным скриптовым языком во всемирной паутине является *JavaScript*. Он обычно используется как встраиваемый язык для программного доступа к объектам приложений, а наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

*JavaScript* – это легковесный, интерпретируемый язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и во множестве других программных продуктов. Это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный стили программирования.

Средствами языка *JavaScript* можно добавлять различные эффекты анимации, реагировать на события (обрабатывать перемещения указателя мыши, нажатие клавиш с клавиатуры и прочее), запрашивать у пользователя данные, осуществлять проверку ввода данных в поля формы до отправки на сервер, создавать и считывать *cookie*, определять браузер, изменять содержимое *HTML*-страницы (в том числе добавлять новые теги и изменять стили элементов).

1. Цель работы

Закрепить знания языка программирования *JavaScript* и познакомиться с возможностями его применения на веб-страницах для изменения их содержимого и создания анимации. Создать новые страницы сайта, на которых продемонстрировать работу с операторами, функциями, событиями, объектами, их методами и свойствами, а также с формами в *JavaScript*. Познакомиться с *DHTML* и создать анимацию, предложенную в варианте задания.

1. Теоретическая часть
   1. Введение в *DHTML*

Веб-страницы бывают статическими и динамическими, последние отличаются тем, что в них используются сценарии (программы) на языке *JavaScript*.

Для того чтобы изменять уже нарисованный браузером экран или создавать новые окна, нужно проникнуть в иерархию объектов браузера, получить доступ к его объектам. В сценариях *JavaScript* браузер предоставляет веб-разработчику множество «готовых» объектов, с помощью которых он может взаимодействовать с элементами веб-страницы и самим браузером. В совокупности же все эти объекты составляют объектную модель браузера (*BOM* – *Browser Object Model*).

Схема объектной модели браузера приведена на рисунке 1.

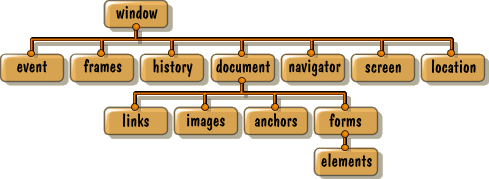


Рисунок 1 – Схема объектной модели браузера

На самом верху этой модели находится глобальный объект *window*. Он представляет собой одно из окон или вкладку браузера с его панелями инструментов, меню, строкой состояния, *HTML*-страницей и другими объектами. Доступ к этим различным объектам окна браузера осуществляется с помощью следующих основных объектов: *navigator*, *history*, *location*, *screen*, *document* и некоторых других. Так как данные объекты являются дочерними по отношению к объекту *window*, то обращение к ним происходит как к свойствам объекта *window*. Кроме того, объект *window* определяет множество методов для работы с браузером, например, *alert*(), *open*(), *setTimeout*().

Из всех этих объектов наибольший интерес и значимость для разработчика представляет объект *document*, который является корнем объектной модели документа (*DOM* – *Document Object Model*). Данная модель в отличие от объектной модели браузера стандартизована в спецификации и поддерживается всеми браузерами. Объект *document* представляет собой *HTML*-документ, загруженный в окно (вкладку) браузера. С помощью свойств и методов данного объекта можно получить доступ к содержимому *HTML*-документа, а также изменить его содержимое, структуру и оформление. Такой подход к созданию интерактивного веб-сайта, использующий сочетание статичного языка разметки *HTML*, каскадных таблиц стилей *CSS*, встраиваемого (и выполняемого на стороне клиента) скриптового языка *JavaScript*, а также объектной модели документа *DOM*, получил название *DHTML* (*Dynamic HTML*).

* 1. Основы языка *JavaScript*, используемые в работе

* + 1. Объекты – это динамические структуры, состоящие из пар ключ-значение. Значения могут иметь примитивные типы данных, могут быть объектами или функциями.

Объекты используются для хранения коллекций различных значений и более сложных сущностей. Объект может быть создан с помощью фигурных скобок с необязательным списком свойств. Свойство – это пара «ключ: значение», где ключ – это строка (также называемая «именем свойства»), а значение может быть примитивом, объектом или функцией.

Пример создания объекта:

*let obj = {*

*message : "A message",*

*doSomething : function() {}*

*}*

Свойства объекта можно, в любое время, читать, добавлять, редактировать и удалять. Обращение к свойству объекта выполняется «через точку»: *object*.*attribute*. Для обращения к свойствам, название которых состоит из нескольких слов, существует альтернативный способ доступа через квадратные скобки: *object*.[“*complex attribute*”]. Можно использовать квадратные скобки в литеральной нотации для создания вычисляемого свойства:

*let fruit = prompt("Какой фрукт купить?", "apple");*

*let bag = {*

*[fruit]: 10,*

*};*

Для задания свойства, равного значению переменной и имеющего такое же название, допустимо в список свойств объекта внести название такой переменной.

В отличие от многих других языков, особенность *JavaScript*-объектов в том, что можно получить доступ к любому свойству. Даже если свойства не существует, ошибки не возникает. При обращении к свойству, которого нет, возвращается *undefined*.

Для проверки существования свойства в объекте присутствует специальный оператор *in*, возвращающий логическое значение.

Перебор всех свойств объекта в цикле можно реализовать следующим образом: *for* (*key in object*) {}. Также можно воспользоваться командой *Object*.*keys*().

Для удаления свойства можно использовать оператор *delete*.

Объекты в языке реализованы в виде хэш-таблиц. Простую хэш-таблицу можно создать, используя команду *Object*.*create*(*null*).

Если объект нужно сделать неизменяемым, можно воспользоваться командой *Object*.*freeze*().

Помимо простых объектов в *JavaScript* есть много других типов объектов, например, таких как *Array* (для хранения упорядоченных коллекций данных), *Date* (для хранения информации о дате и времени), *Error* (для хранения информации об ошибке).

* + 1. Функции в *JavaScript* являются объектами. Функции можно назначать переменным, хранить в объектах или массивах, передавать в виде аргументов другим функциям и возвращать из других функций.

Существует три способа объявления функций:

– классическое объявление функции (*Function Declaration* *или Function Statement*);

– использование функциональных выражений (*Function Expression*), которые ещё называют функциональными литералами (*Function Literal*);

– использование синтаксиса стрелочных функций (*Arrow Function*), имеющих более короткий синтаксис.

Примеры объявления функций такими способами:

*function doSomething(){}*

*let doSomething = function() {}*

*let doSomething = () = > {};*

Кроме того, функции можно вызывать различными способами: обычным, в виде метода объекта, в виде конструктора, в виде функции с использованием метода *apply*(), в виде функции с использованием метода *bind*().

Функции можно вызывать с большим или меньшим количеством аргументов, чем то количество параметров, которое было задано при их объявлении. В ходе работы функции «лишние» аргументы будут просто проигнорированы (хотя у функции будет доступ к ним), отсутствующие параметры получат значение *undefined*.

У функций есть два псевдо-параметра: *this* и *arguments*.

Ключевое слово *this* представляет собой контекст функции. Значение, на которое оно указывает, зависит от того, как была вызвана функция.

Ключевое слово *arguments* – это псевдопараметр, который даёт доступ ко всем аргументам, использованным при вызове функции. Он похож на массив, но массивом не является. В частности, у него нет методов массива.

* + 1. Событие – это сигнал от браузера о том, что что-то произошло. Все *DOM*-узлы могут подавать такие сигналы (хотя события бывают и не только в *DOM*).

Событию можно назначить обработчик, то есть функцию, которая сработает, как только событие произошло. Именно благодаря обработчикам *JavaScript*-код может реагировать на действия пользователя.

Есть несколько способов назначить событию обработчик:

– использование атрибута элемента *HTML* (например, после задания атрибута *onclick* при щелчке мыши по данному элементу будет выполнен код, указанный в этом атрибуте);

– использование в скрипте *JavaScript* свойства *DOM*-объекта, соответствующее некоторому событию;

– использование метода *addEventListener*(*event*, *handler*[, *phase*]).

Последний способ является более гибким, он позволяет добавлять несколько обработчиков на одно и то же событие. Для удаления обработчика следует использовать метод *removeEventListener*()с аналогичными параметрами.

Когда происходит событие, браузер создаёт объект события, записывает в него детали и передаёт его в качестве аргумента функции-обработчику. В объекте события содержатся такие свойства, как *type* (тип события, например, *click*), *currentTarget* (элемент, на котором сработал обработчик), *clientX* и *clientY* (координаты курсора в момент клика относительно окна, для событий мыши) и некоторые другие.

Можно назначить обработчиком не только функцию, но и объект при помощи *addEventListener*(). В этом случае, когда происходит событие, вызывается метод этого объекта *handleEvent*(), в котором необходимо разместить логику обработки события.

* + 1. Для осуществления условного ветвления в алгоритмах в языке *JavaScript* присутствуют оператор *if* и тернарный оператор «?». Для задания ветвления с большим количеством известных возможных направлений используется конструкция *switch*. Примеры использования операторов условного ветвления:

*let accessAllowed;*

*let age = prompt('Сколько вам лет?', '');*

*if (age > 16) {*

*accessAllowed = true;*

*} else {*

*accessAllowed = false;*

*}*

*let accessAllowed = (age > 16) ? true : false;*

*let a = prompt('Сколько будет 5+5?', '');*

*switch (a) {*

*case 10:*

*alert( 'В точку!' );*

*break;*

*case 8:*

*case 9:*

*alert( 'Мало' );*

*break;*

*case 11:*

*case 12:*

*alert( 'Много' );*

*break;*

*default:*

*alert( "Ответ странный" );*

*}*

* + 1. Для повторения однотипных действий в языке *JavaScript* присутствуют три типа циклов: *while*, *do while* и *for*.

Цикл *while* проверяет условие выполнения перед каждой итерацией, *do* *while* – после каждой итерации. Эти циклы используют, как правило, при неизвестном количестве итераций. Второй гарантирует, что тело цикла будет выполнено как минимум один раз.

Цикл *for*, как и *while*, проверяет условие выполнения перед каждой итерацией, но имеет более сложную структуру: он состоит из начала, условия, шага и тела. Этот цикл обычно используется при известном количестве итераций.

Для принудительного прерывания цикла используется оператор *break*, для перехода к следующей итерации – *continue*. В обоих операторах можно указывать метку, которая соответствует целевому циклу (можно использовать для воздействия на внешний цикл из внутреннего).

Примеры использования циклов:

*let i = 0;*

*while (i < 5) {*

*alert( i );*

*i++;*

*}*

*i = 0;*

*do {*

*alert( i );*

*i++;*

*} while (i < 5);*

*for (i = 0; i < 5; i++) {*

*alert(i);*

*}*

* + 1. Один из способов взаимодействия с пользователями представляют *HTML*-формы. Например, если нужно получить от пользователя некоторую информацию, можно определить на веб-странице форму, которая будет содержать текстовое поле для ввода информации и кнопку для отправки.

В *JavaScript* форма представлена объектом *HtmlFormElement*. Формы в документе входят в специальную коллекцию *document*.*forms*. Это так называемая «именованная» коллекция: для получения формы можно использовать как её имя (атрибут name), так и её порядковый номер в документе (например, *document*.*forms*.*form*1 и *document*.*forms*[0]). Любой элемент формы доступен в «именованной» коллекции *form*.*elements*.

Среди методов формы надо отметить метод *submit*(), который отправляет данные формы на сервер, и метод *reset*(), который очищает поля формы.

Форма может содержать различные элементы ввода *HTML*: *input*, *textarea*, *button*, *select* и другие. Но все они имеют ряд общих свойств и методов. Также, как и форма, элементы форм имеют свойство *name*, с помощью которого можно получить значение атрибута *name*. Другим важным свойством является свойство *value*, которое позволяет получить или изменить значение поля (*checked* для чекбоксов и переключателей). С помощью свойства *form* можно получить родительский объект формы. Свойство *type* позволяет получить тип поля ввода. Это либо название тега элемента (например, *textarea*), либо значение атрибута *type* у элементов *input*. Из методов можно также выделить методы *focus*() (устанавливает фокус на элемент) и *blur*() (убирает фокус с элемента).

* + 1. Для работы с датой и временем в языке *JavaScript* используются объекты *Date*. При создании объекта без указания параметров он инициализируется текущей датой и временем.

Объекты *Date* имеют множество методов для получения и установки даты и времени, их преобразования к различным единицам измерения времени, вычисления разности дат и некоторых других операций. Также присутствует механизм авто-исправления некорректных параметров конструктора для даты, например, когда 32-й день месяца преобразуется к первому дню следующего.

* 1. *HTML*-элемент *canvas*

*Canvas* – элемент *HTML*, предназначенный для создания растрового двухмерного изображения при помощи скриптов, обычно на языке *JavaScript*. Начало отсчёта блока находится слева сверху. От него и строится каждый элемент блока. Размер пространства координат не обязательно отражает размер фактической отображаемой площади. По умолчанию его ширина равна 300 пикселям, а высота – 150.

Элемент *canvas* создает область, в которой при помощи *JavaScript* можно рисовать разные объекты, выводить изображения, трансформировать их и менять свойства. На практике, он используется, как правило, для отрисовки графиков для статей и игрового поля в некоторых браузерных играх, но также может использоваться для встраивания видео в страницу и создания полноценного плеера.

Для работы используется объект, содержащий в себе так называемый контекст выполнения, который получают вызовом метода *getContext*() элемента *canvas*. Всё рисование происходит с помощью методов этого объекта.

Некоторые методы объекта контекста:

– *moveTo* (перемещает начальную точку нового фрагмента контура в точку с заданными координатами);

– *lineTo* (соединяет последнюю точку фрагмента контура и точку с заданными координатами прямой линией);

– *beginPath* (создаёт новый пустой контур, обнуляя список фрагментов контура);

– *closePath* (перемещает точку рисования назад к началу текущего фрагмента контура.);

– *stroke* (выполняет обводку фрагментов контура обводкой текущего стиля);

– *fill* (заливает фрагменты контура текущей заливкой);

– *strokeRect* (рисует на холсте прямоугольник с начальной точкой в заданной позиции, имеющий заданные ширину и высоту, используя текущий стиль обводки);

– *fillRect* (рисует залитый прямоугольник в заданной позиции, размер которого определяется аргументами ширины и высоты, используя текущий стиль);

– *clearRect* (устанавливает прозрачный чёрный цвет для всех пикселей, расположенных внутри прямоугольника с заданным положением, шириной и высотой, таким образом стирая любое ранее нарисованное содержимое);

– *arc* (добавляет к контуру дугу, определяющуюся заданными координатами центра, радиусом, углами начала и конца, рисующуюся в заданном направлении относительно движения часовой стрелки).

1. Практическая часть
   1. Обновление структуры проекта

Для выполнения лабораторной работы были добавлены пять новых страниц: *LoopsAndConditions*, *FunctionsAndEvents*, *ObjectsMethodsAndProps*, *DHTML*, *BallsAnimation*. В соответствующих им директориях расположены файлы *index.html*, *index.js*, *styles.css*. В некоторых директориях также расположены вспомогательные файлы и *HTML*-страницы, необходимые для корректной работы основных добавленных страниц. Полученная структура проекта приведена на рисунке 2.

Graphical user interface, text

Description automatically generated

Рисунок 2 – Обновлённая структура проекта

Связи между *HTML*-документами и соответствующими им локальными *JavaScript*-файлами устанавливались через элемент *script* следующим образом:

*<script type="text/javascript" src="./index.js"></script>* .

* 1. Руководство пользователя
     1. После доработки сайта приведём описание новых страниц для пользователя.

На сайт были добавлены страницы со следующими названиями:

– «Примеры использования циклов и условий»;

– «Примеры использования функций и событий»;

– «Примеры использования объектов, методов и свойств в *JavaScript*»;

– «Примеры использования *DHTML*»;

– «Анимация шаров в коробке».

Осуществить переход на данные страницы можно по ссылкам из «Домашней страницы».

На новых страницах сайта также присутствует панель навигации (расположена в шапке страниц), а для возврата на «Домашнюю страницу» в подвале этих страниц присутствует соответствующая ссылка.

* + 1. «Домашняя страница» содержит список ссылок на все информативные страницы сайта. Это первая страница, которая отображается пользователю при открытии сайта. Для перехода на другие страницы достаточно выбрать одну из ссылок в списке навигации. Ссылки сгруппированы по разделам.

«Домашняя страница» сайта представлена на рисунке 3.

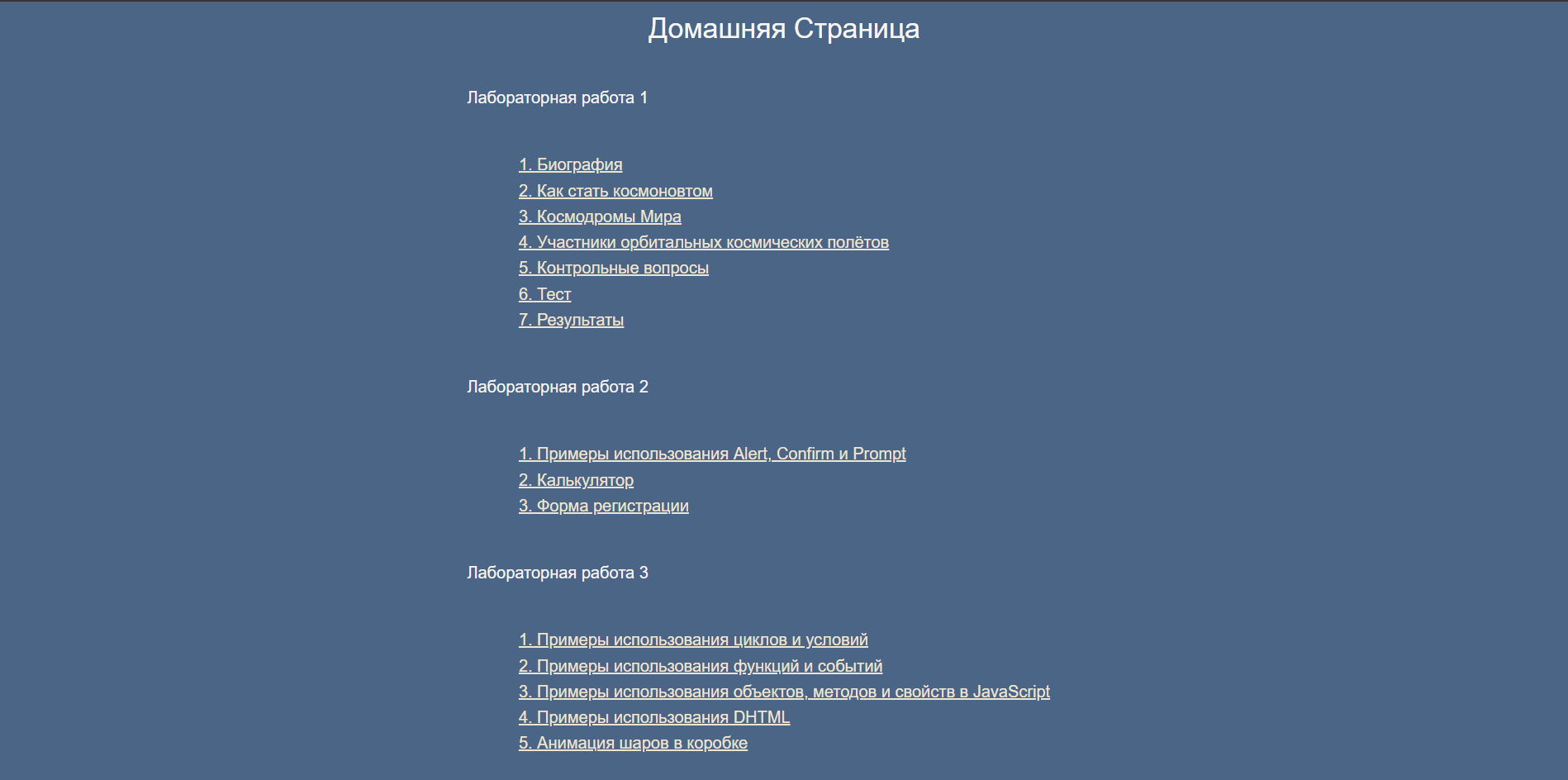


Рисунок 3 – «Домашняя страница»

* + 1. Страница «Примеры использования циклов и условий» содержит десять кнопок для демонстрации работы условного ветвления и циклов в языке *JavaScript*.

В шапке страницы расположена панель навигации для дальнейших переходов по страницам сайта. В подвале страницы – ссылка для возврата на «Домашнюю страницу».

Страница «Примеры использования циклов и условий» приведена на рисунке 4.

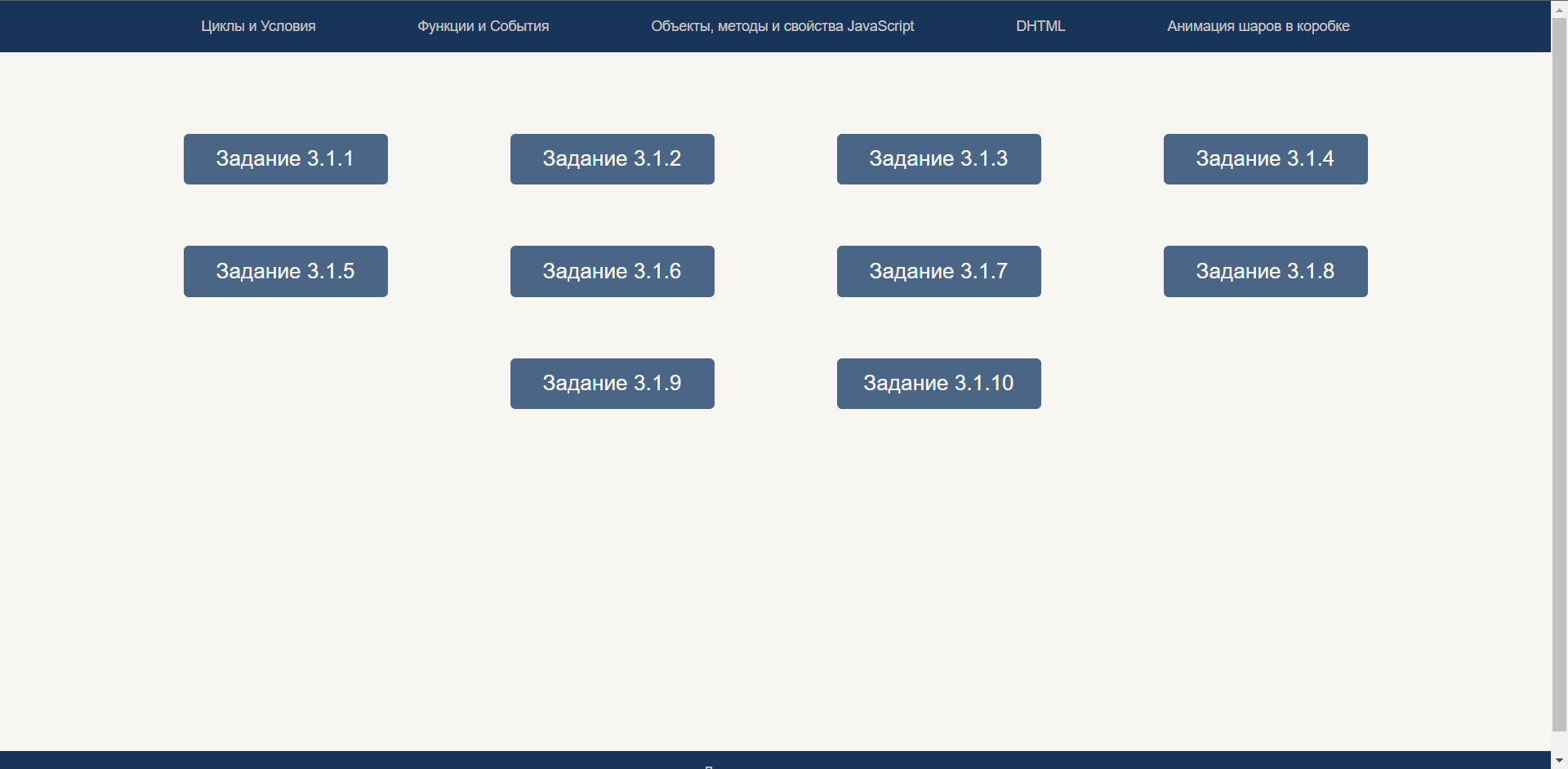


Рисунок 4 – Страница «Примеры использования циклов и условий»

Каждая кнопка соответствует некоторому примеру использования условного ветвления и циклов в языке *JavaScript*. Ниже приведено описание работы некоторых из них.

При нажатии на кнопку «Задание 3.1.1» вверху страницы отображается модальное окно с запросом подтверждения действия и кнопками «ОК» и «Отмена». Внешний вид этогомодального окна представлен на рисунке 5. При утвердительном ответе последовательно отображаются два сообщения, при отрицательном – только одно.

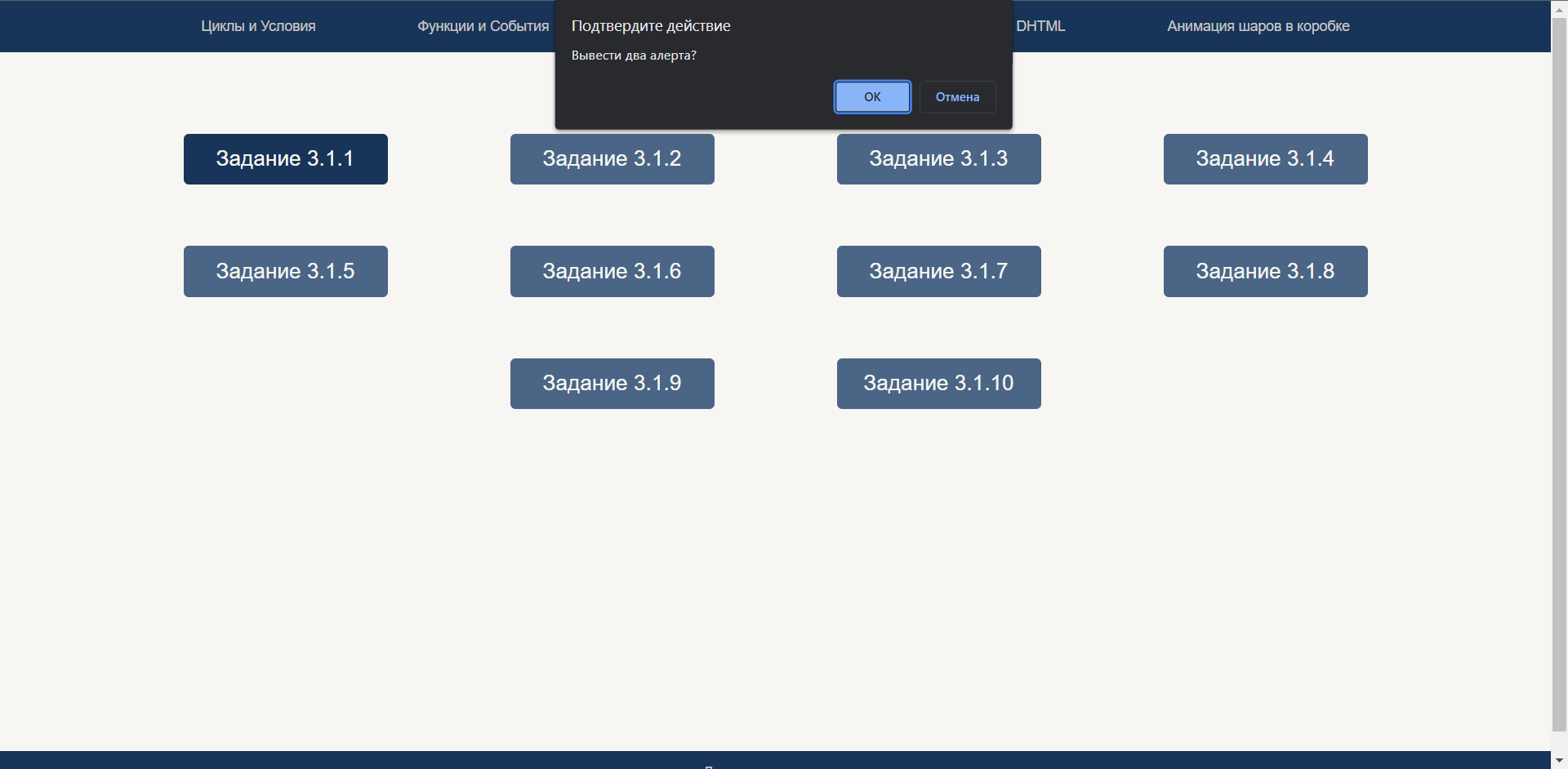


Рисунок 5 – Модальное окно *confirm*() для запроса подтверждения действия от пользователя после нажатия на кнопку «Задание 3.1.1»

При нажатии на кнопку «Задание 3.1.10» вверху страницы отображается модальное окно с запросом ввода пароля и кнопками «ОК» и «Отмена». Его внешний вид представлен на рисунке 6.

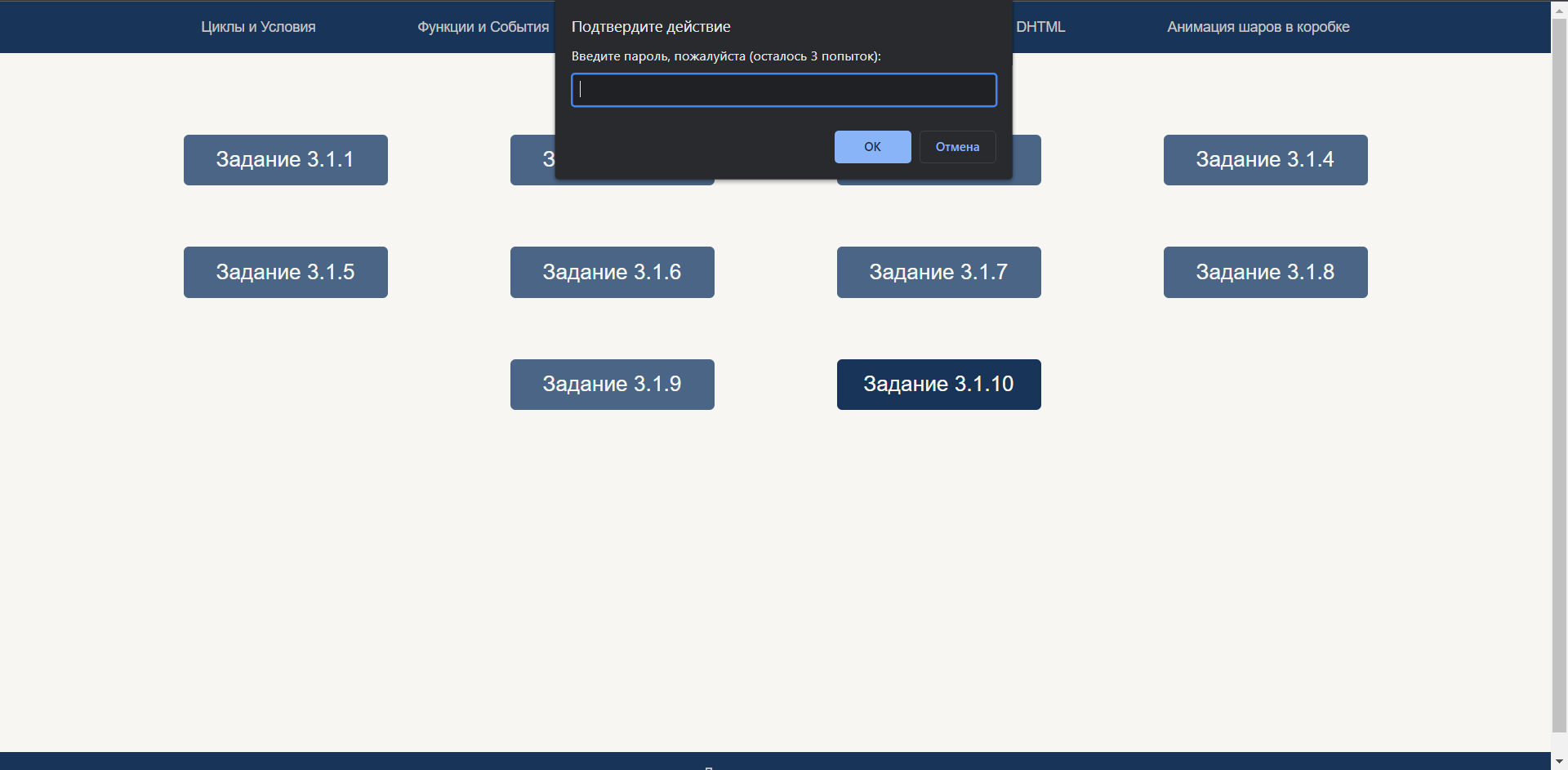


Рисунок 6 – Модальное окно *prompt*()для запроса ввода пароля пользователем после нажатия на кнопку «Задание 3.1.10»

На ввод пароля пользователю предоставляется три попытки. При вводе верного пароля вверху отображается сообщение о успешном вводе пароля, изображённое на рисунке 7.

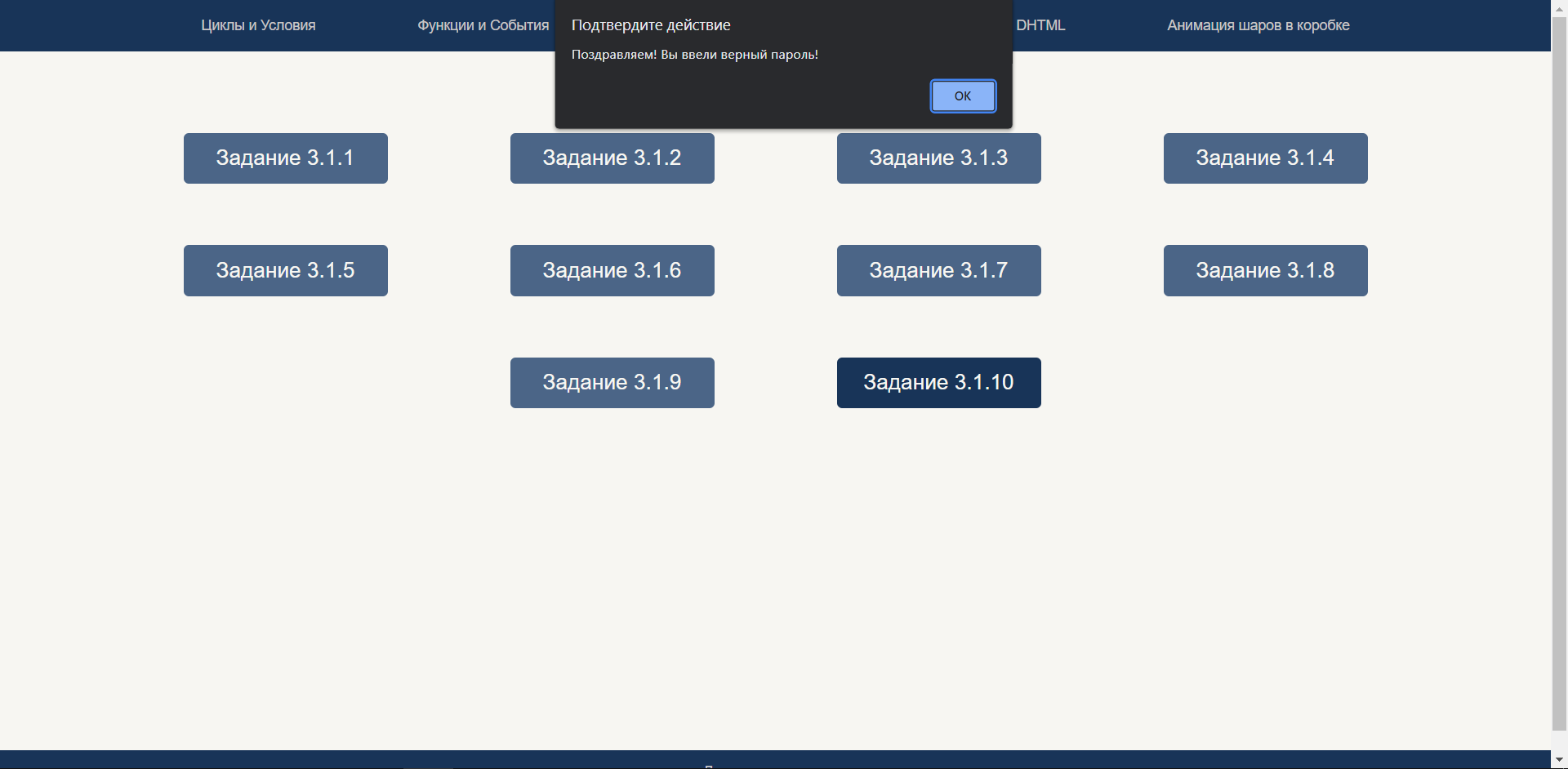


Рисунок 7 – Сообщение *alert*() о вводе пользователем верного пароля

При неверном вводе отображается модальное окно с запросом повторного ввода пароля и количеством оставшихся попыток. Если все попытки были использованы и верный пароль не был введён, то происходит перенаправление на страницу блокировки, изображённую на рисунке 8.

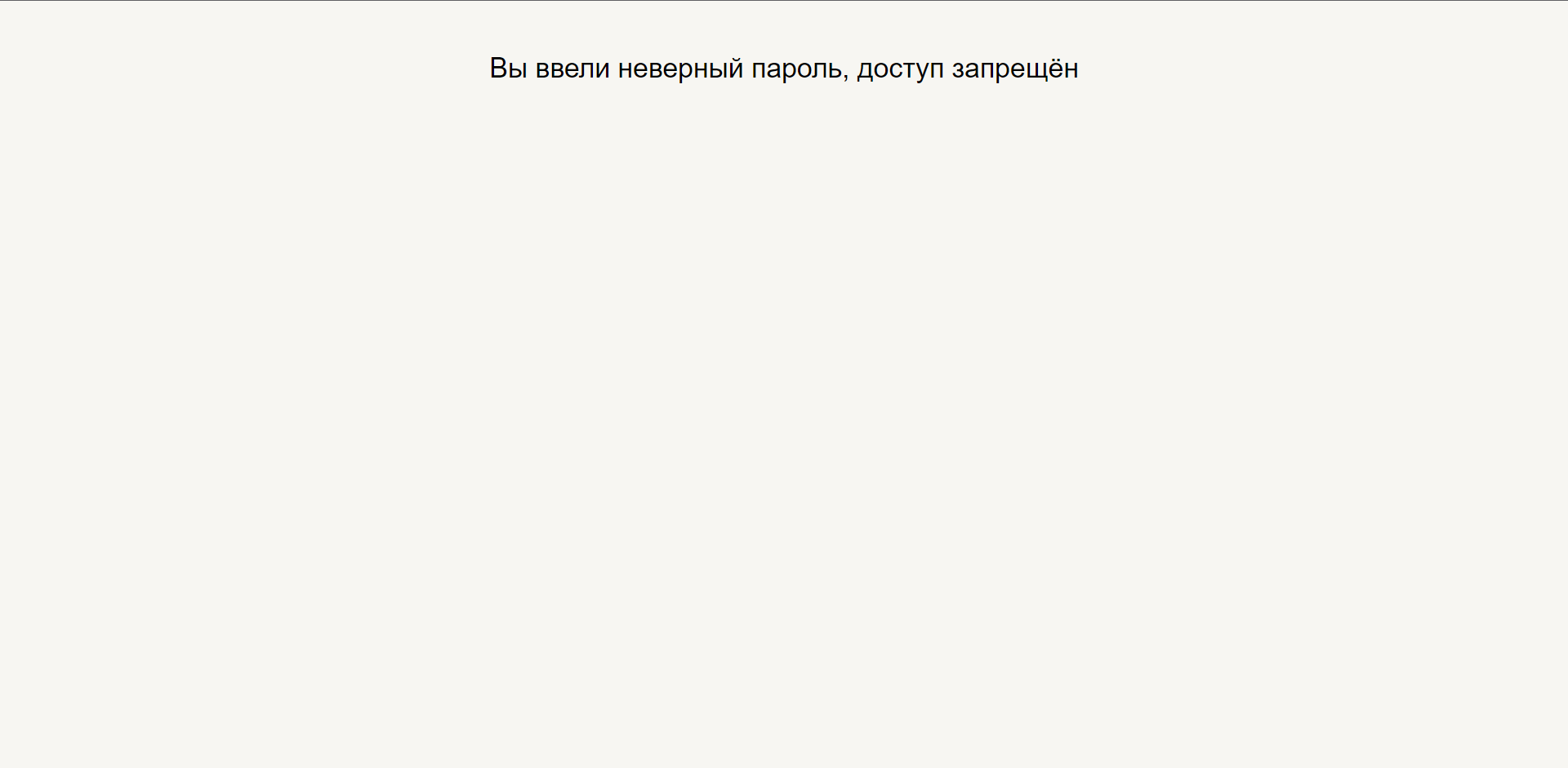


Рисунок 8 – Страница блокировки

* + 1. Страница «Примеры использования функций и событий» содержит шесть кнопок и дополнительную ссылку на «Домашнюю страницу» для демонстрации работы с функциями и событиями в языке *JavaScript*.

В шапке страницы расположена панель навигации для дальнейших переходов по страницам сайта. В подвале страницы – ссылка для возврата на «Домашнюю страницу».

Страница «Примеры использования функций и событий» приведена на рисунке 9.

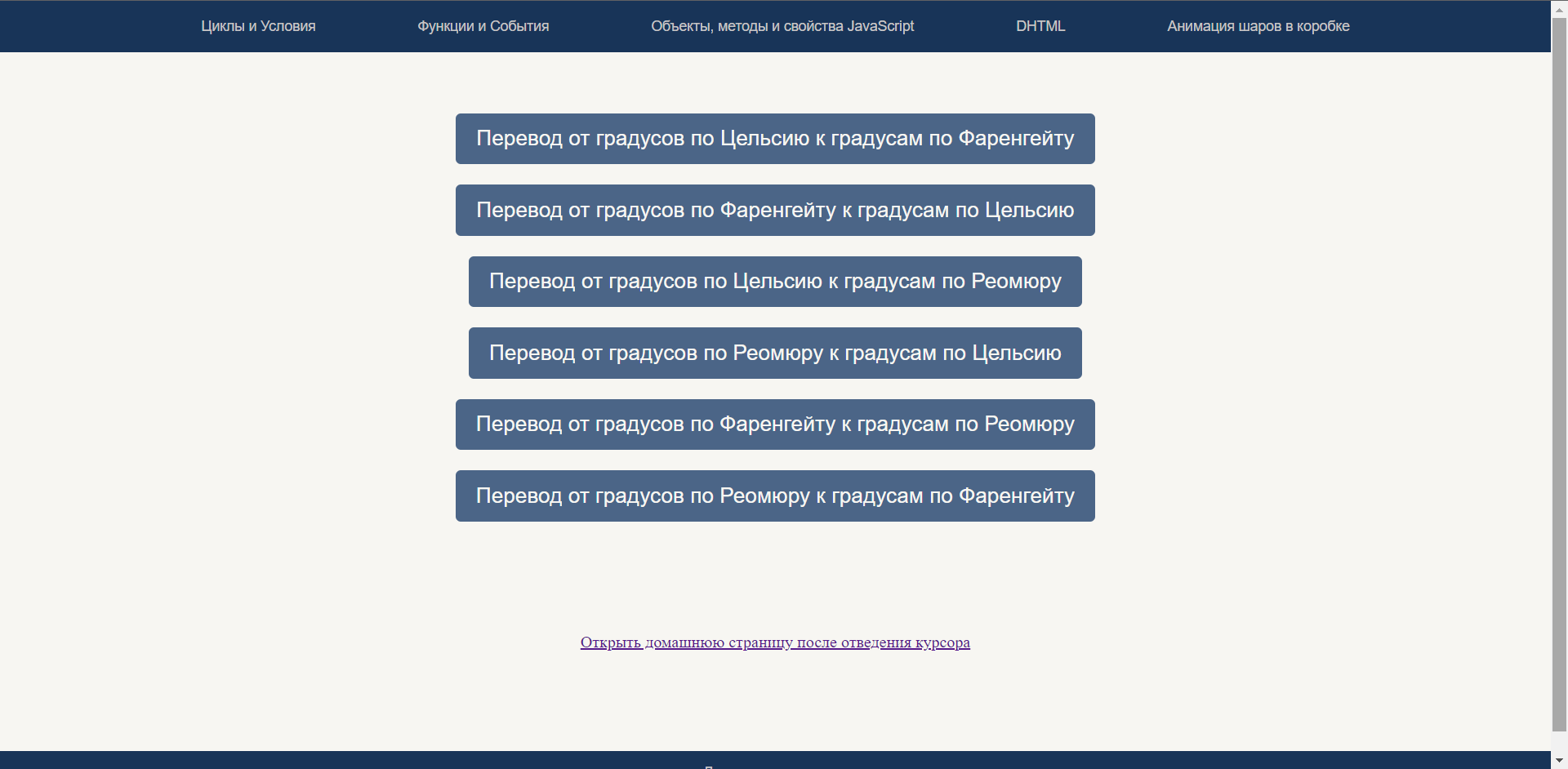


Рисунок 9 – Страница «Примеры использования функций и событий»

Кнопки предназначены для перевода введённой пользователем температуры между различными шкалами её измерения: Цельсия, Реомюра и Фаренгейта. Все кнопки настроены так, что срабатывают только по двойному щелчку мыши по ним. Ниже приведено описание работы одной из кнопок.

При двойном нажатии на кнопку «Перевод от градусов по Цельсию к градусам по Фаренгейту» вверху страницы отображается модальное окно с запросом ввода температуры по шкале Цельсия и кнопками «ОК» и «Отмена». Его внешний вид представлен на рисунке 10. При корректном вводе значения температуры, оно переводится в значение по шкале Фаренгейта и отображается пользователю в модальном окне, изображённом на рисунке 11.

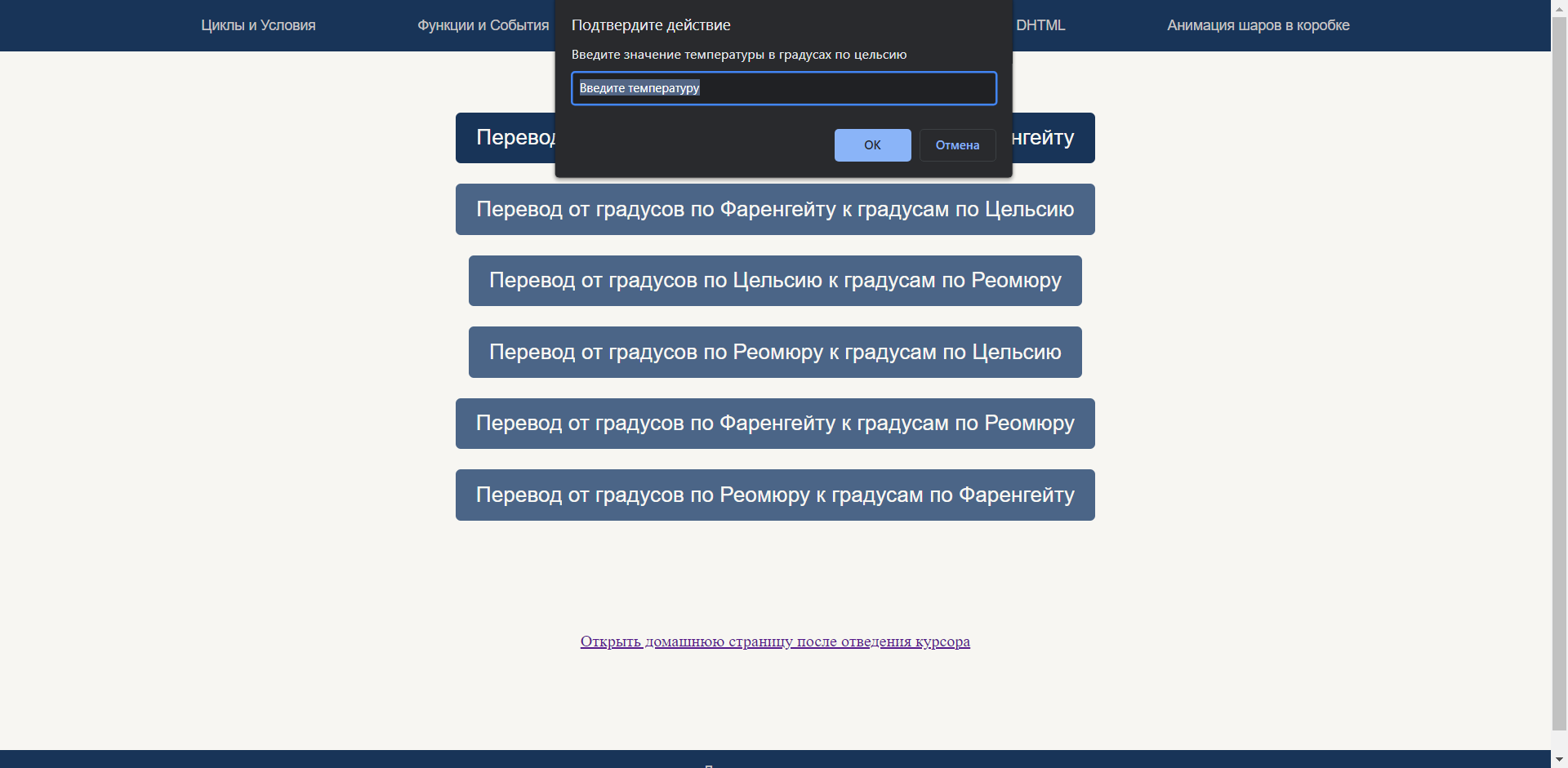


Рисунок 10 – Модальное окно *prompt*() для запроса ввода числового значения температуры по шкале Цельсия

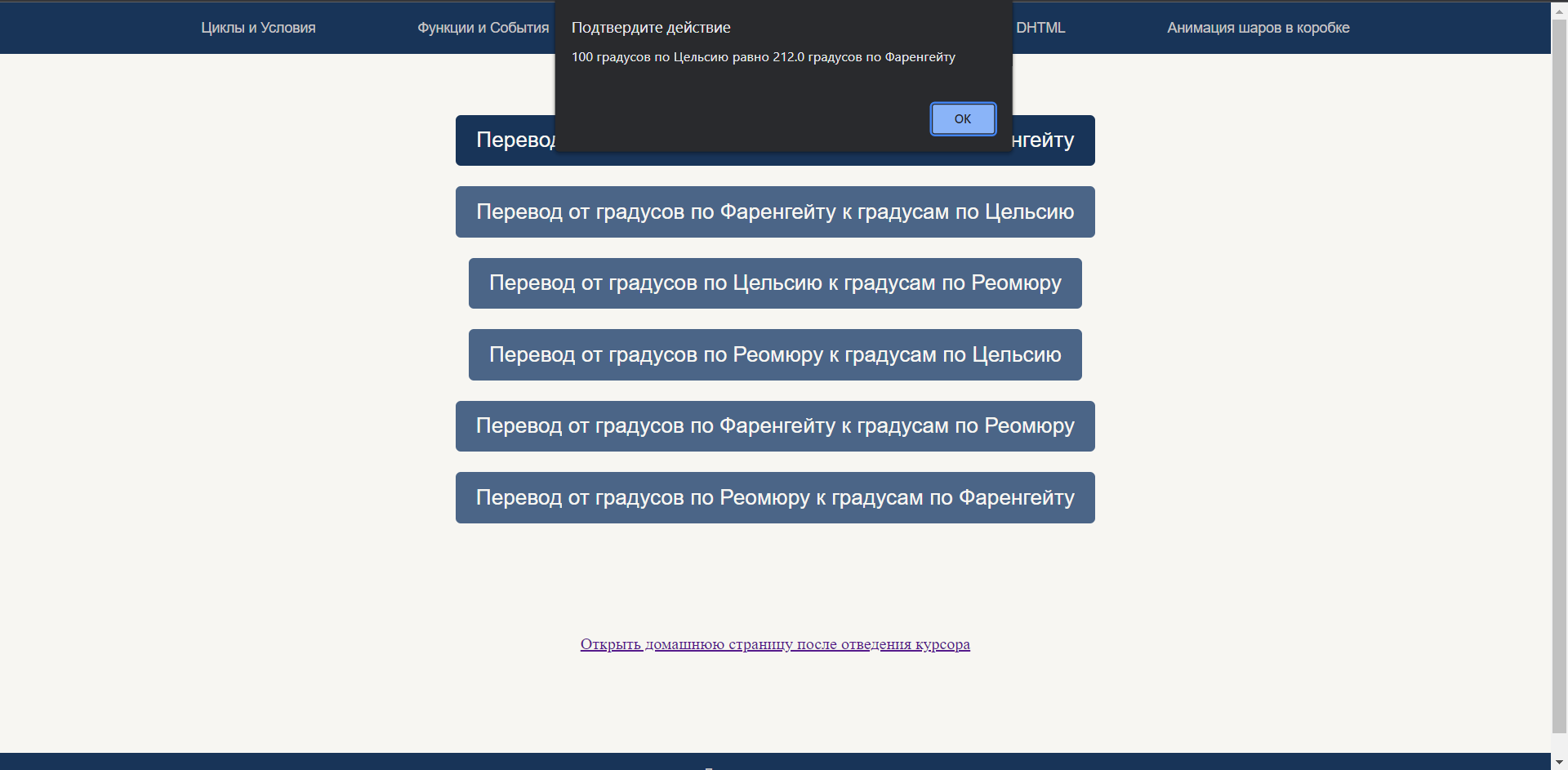


Рисунок 11 – Сообщение *alert*() с результатом перевода числового значения температуры по шкале Цельсия в значение по шкале Фаренгейта

После отведения курсора от дополнительной ссылки на «Домашнюю страницу», расположенной под кнопками, отображается модальное окно с запросом подтверждения перехода на «Домашнюю страницу». Это окно показано на рисунке 12.

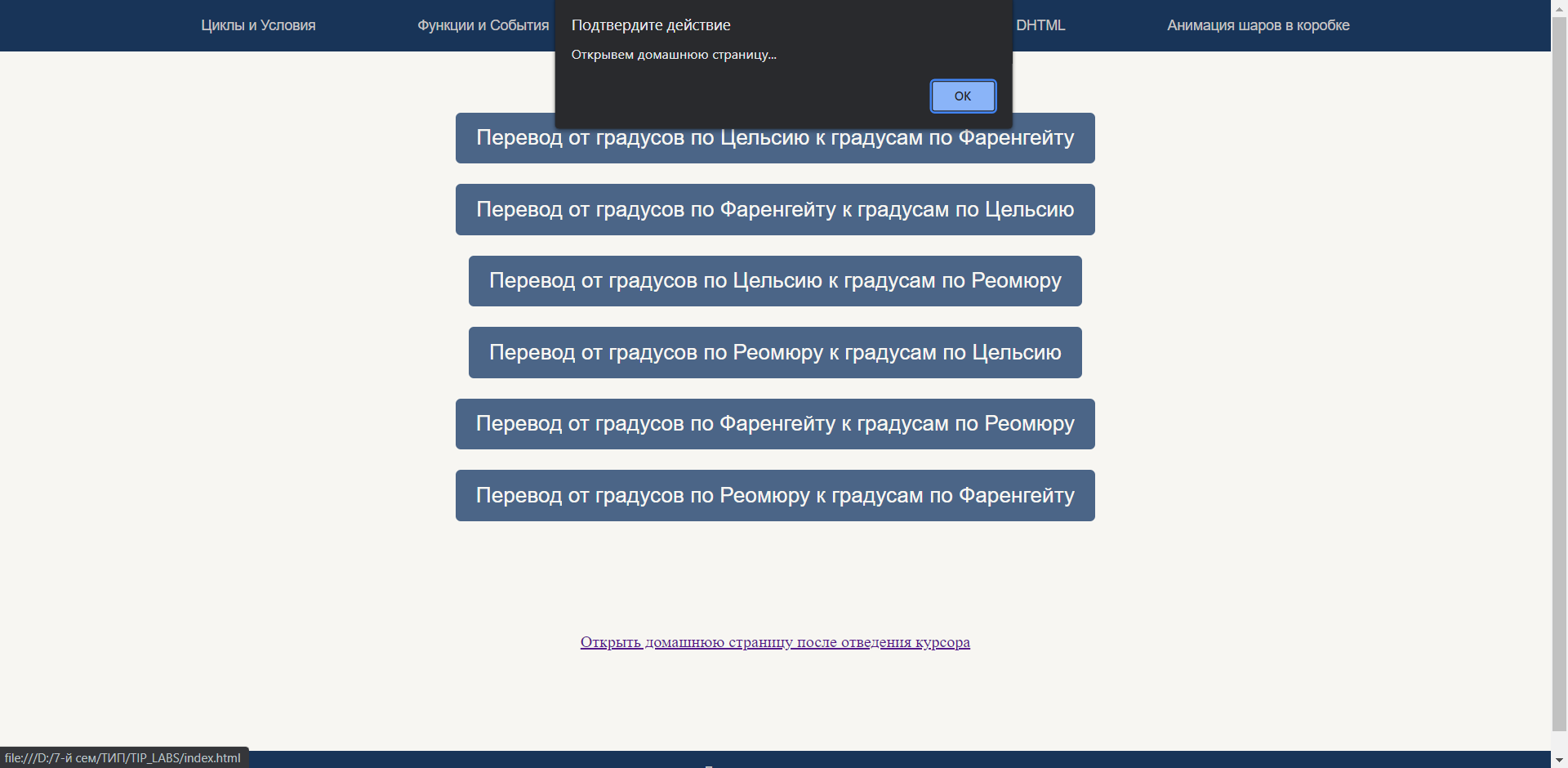


Рисунок 12 – Модальное окно *confirm*() для запроса подтверждения перехода на «Домашнюю страницу»

* + 1. Страница «Примеры использования объектов, методов и свойств в *JavaScript*» содержит несколько форм для ввода и отображения результата обработки введённых данных, а также две кнопки для демонстрации работы с объектами *Math* и *Date* в *JavaScript*. Данная страница показана на рисунке 13.

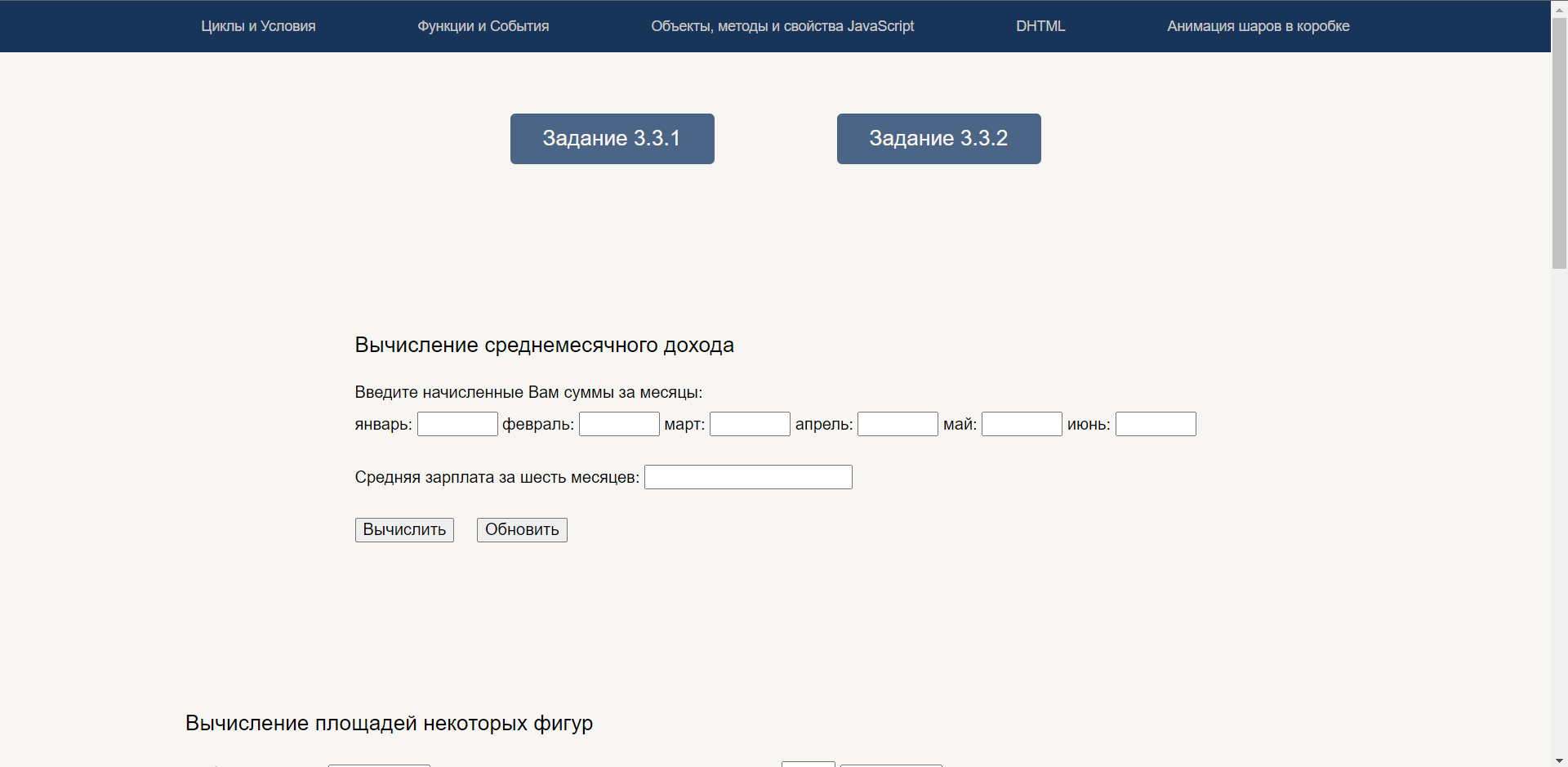


Рисунок 13 – Страница «Примеры использования объектов, методов и свойств в *JavaScript*»

При нажатии на кнопку «Задание 3.3.1» вверху страницы отображается модальное окно с сообщением, содержащим площадь круга с произвольным радиусом от нуля до ста. Его внешний вид представлен на рисунке 14.

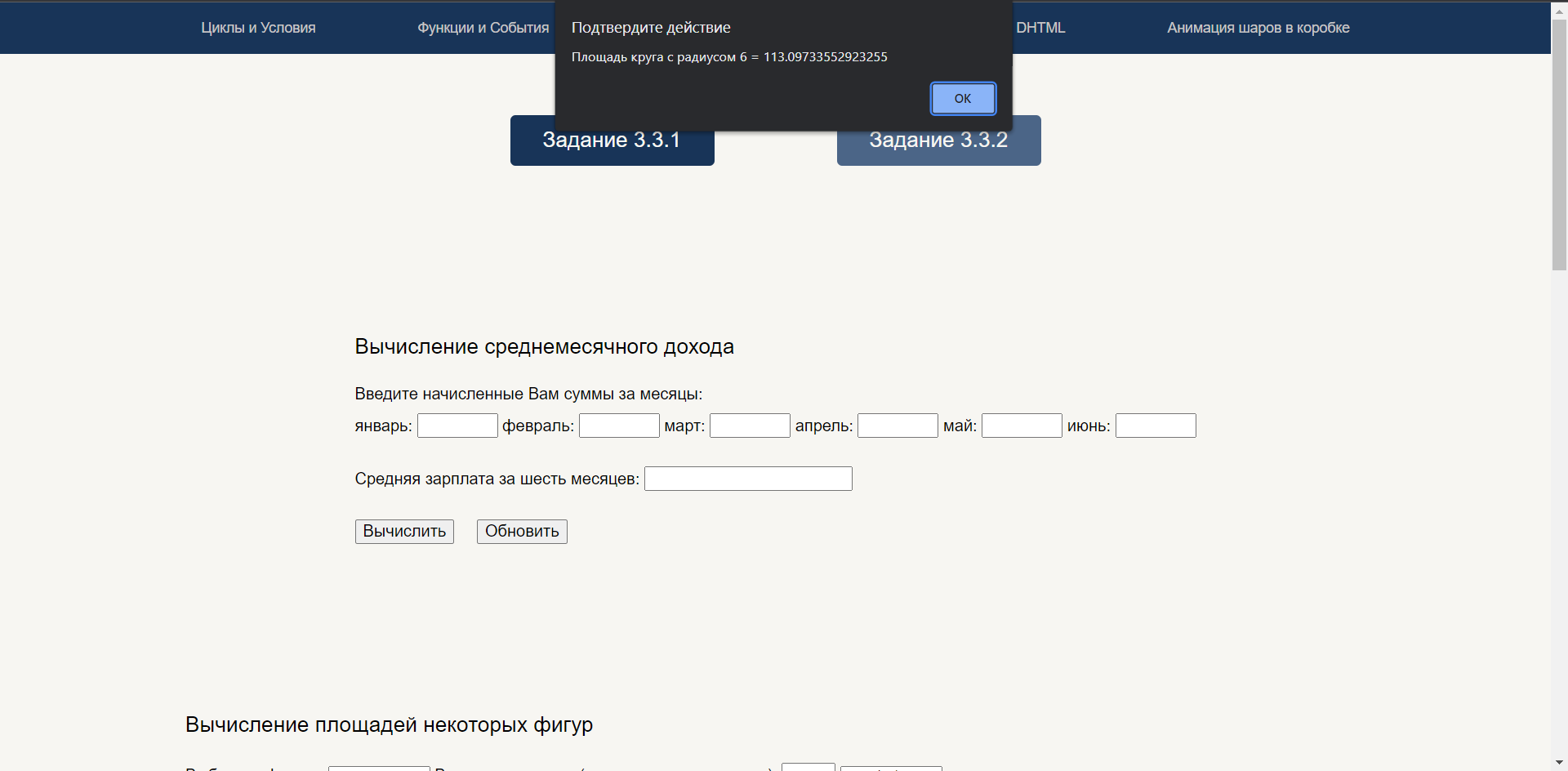


Рисунок 14 – Сообщение *alert*() с результатом вычисления площади круга

При нажатии на кнопку «Задание 3.3.2» вверху страницы отображается модальное окно с сообщением, содержащим дату через неделю от текущей. Его внешний вид представлен на рисунке 15.

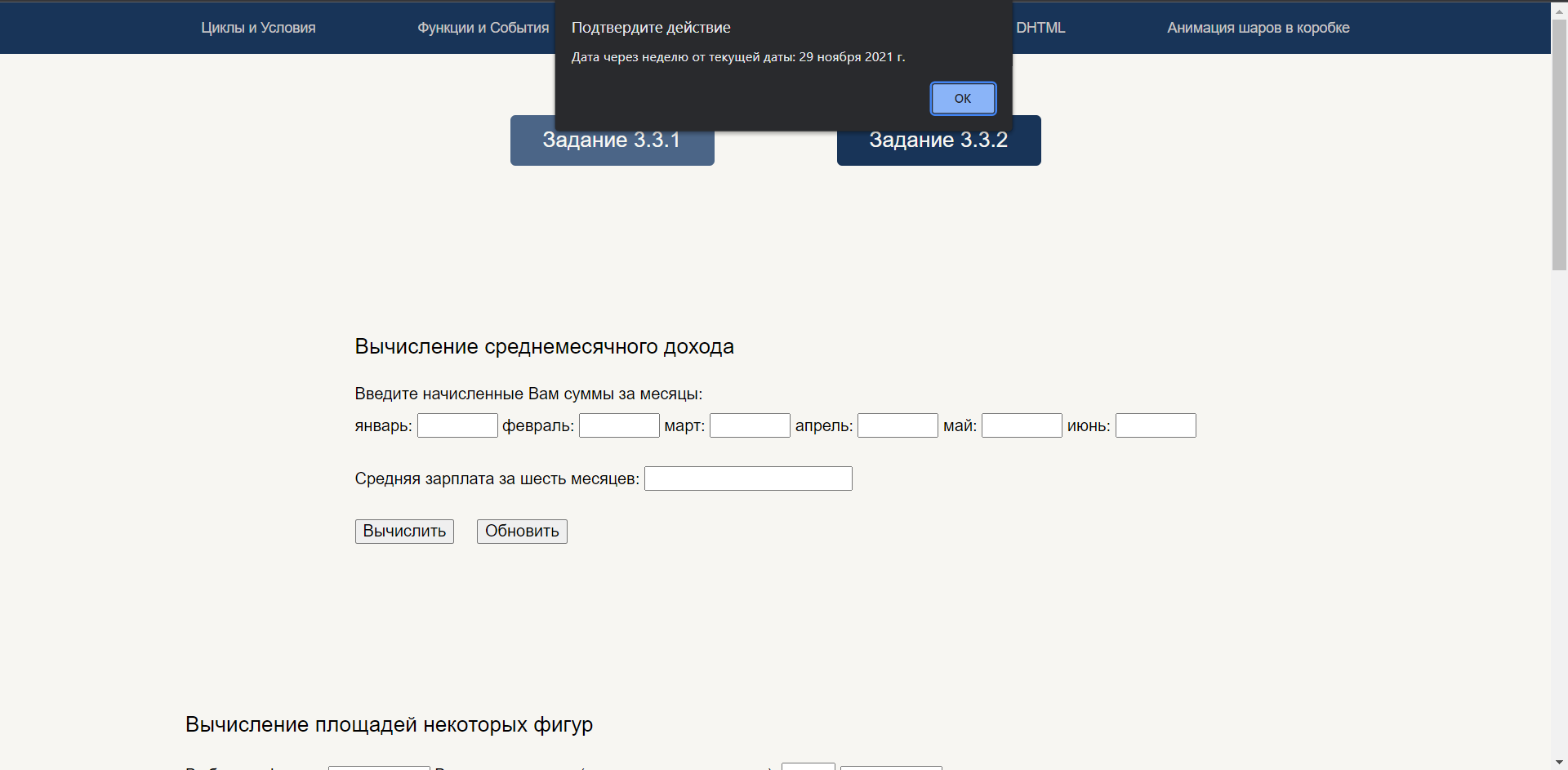


Рисунок 15 – Сообщение *alert*() с результатом вычисления даты через неделю

На сайте присутствуют три формы: для вычисления среднемесячного дохода за шесть месяцев, для вычисления площадей некоторых геометрических фигур и форма для вычисления максимального и минимального дохода за двенадцать месяцев. Ниже приведено описание формы для вычисления среднемесячного дохода за шесть месяцев.

Чтобы вычислить среднемесячный доход за шесть месяцев, необходимо ввести в соответствующую форму данные о доходах за шесть месяцев и нажать кнопку «Вычислить». В текстовом поле ниже отобразится значение среднемесячного дохода, вычисленное по введённым в форму данным. Пример такого расчёта представлен на рисунке 16.

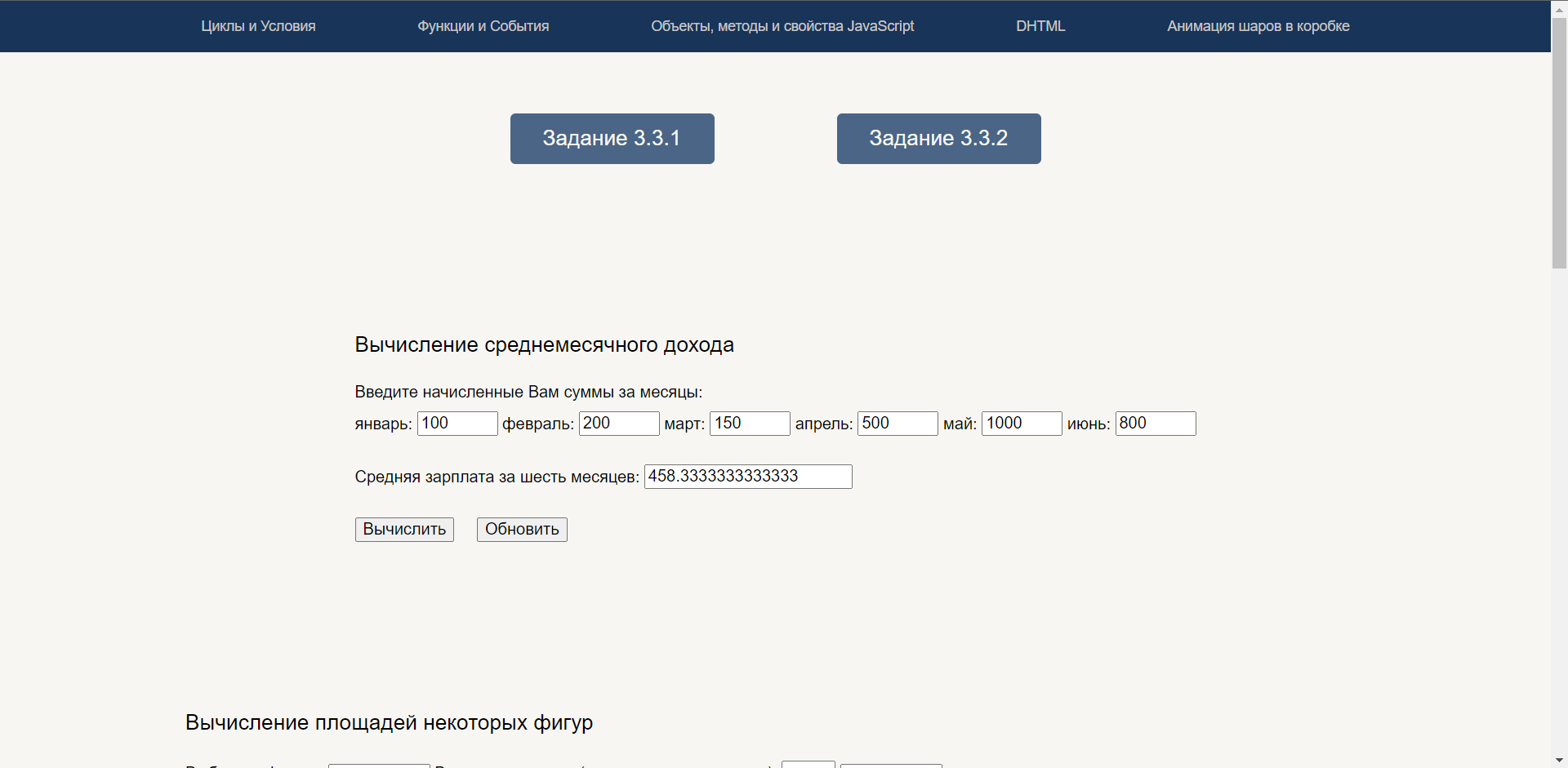


Рисунок 16 – Вычисление среднемесячного дохода за шесть месяцев посредством формы

* + 1. Страница «Примеры использования *DHTML*» содержит набор кнопок для управления анимацией движения планеты и её спутника, а также три кнопки для демонстрации возможностей *DHTML*. В шапке страницы расположена панель навигации для дальнейших переходов по страницам сайта, в подвале страницы – ссылка для возврата на «Домашнюю страницу».

Для управления динамическими эффектами анимации движения планеты и её спутника предназначены пять кнопок:

– кнопка «Ускорить спутник» предназначена для «запуска» спутника и для ускорения вращения при повторном нажатии;

– кнопка «Ускорить вправо» предназначена для начала движения вправо и для ускорения движения вправо при повторном нажатии;

– кнопка «Вправо» предназначена для начала движения вправо из начала координат без изменения текущих скоростей движения элементов системы;

– кнопка «Тормоз спутника» предназначена остановки вращательного движения спутника;

– кнопка «Стоп вправо» предназначена для остановки поступательного движения системы вправо.

При достижении слоем правой границы таблицы слой появляется снова в левой части и его движение повторяется сначала, пока страница не будет обновлена.

Страница «Примеры использования *DHTML»* показана на рисунке 17.

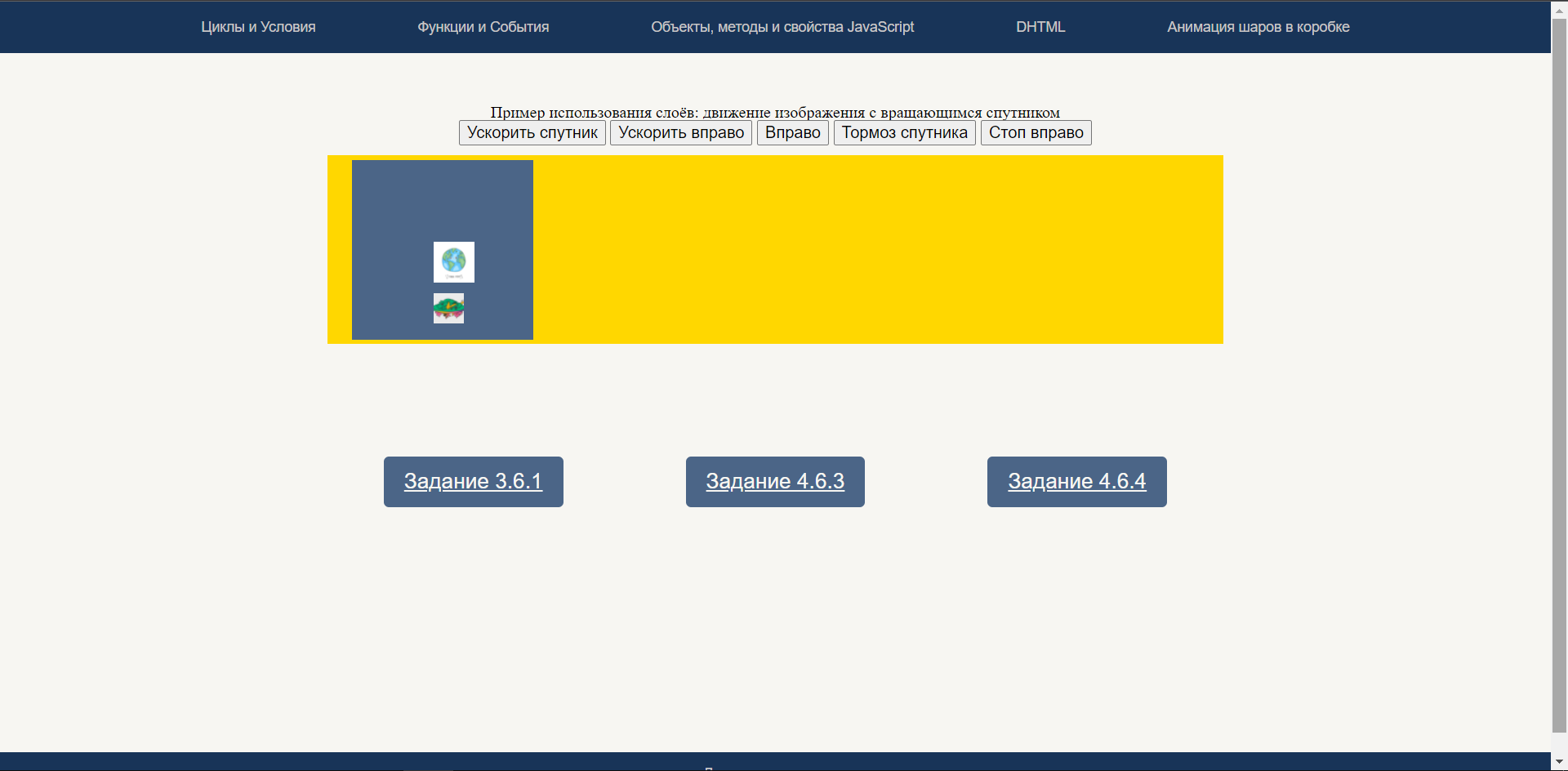


Рисунок 17 – Страница «Примеры использования *DHTML*»

Для демонстрации возможностей *DHTML* на страницу были добавлены три дополнительные кнопки.

При нажатии на кнопку «Задание 3.6.1» открывается новая вкладка, в которой сообщениями последовательно выводится пронумерованный список тегов *HTML*-страницы, а затем отображается она сама. Одно из таких сообщений представлено на рисунке 18. Сама же страница показана на рисунке 19. В данном примере отражена работа с коллекциями элементов в *DHTML*.

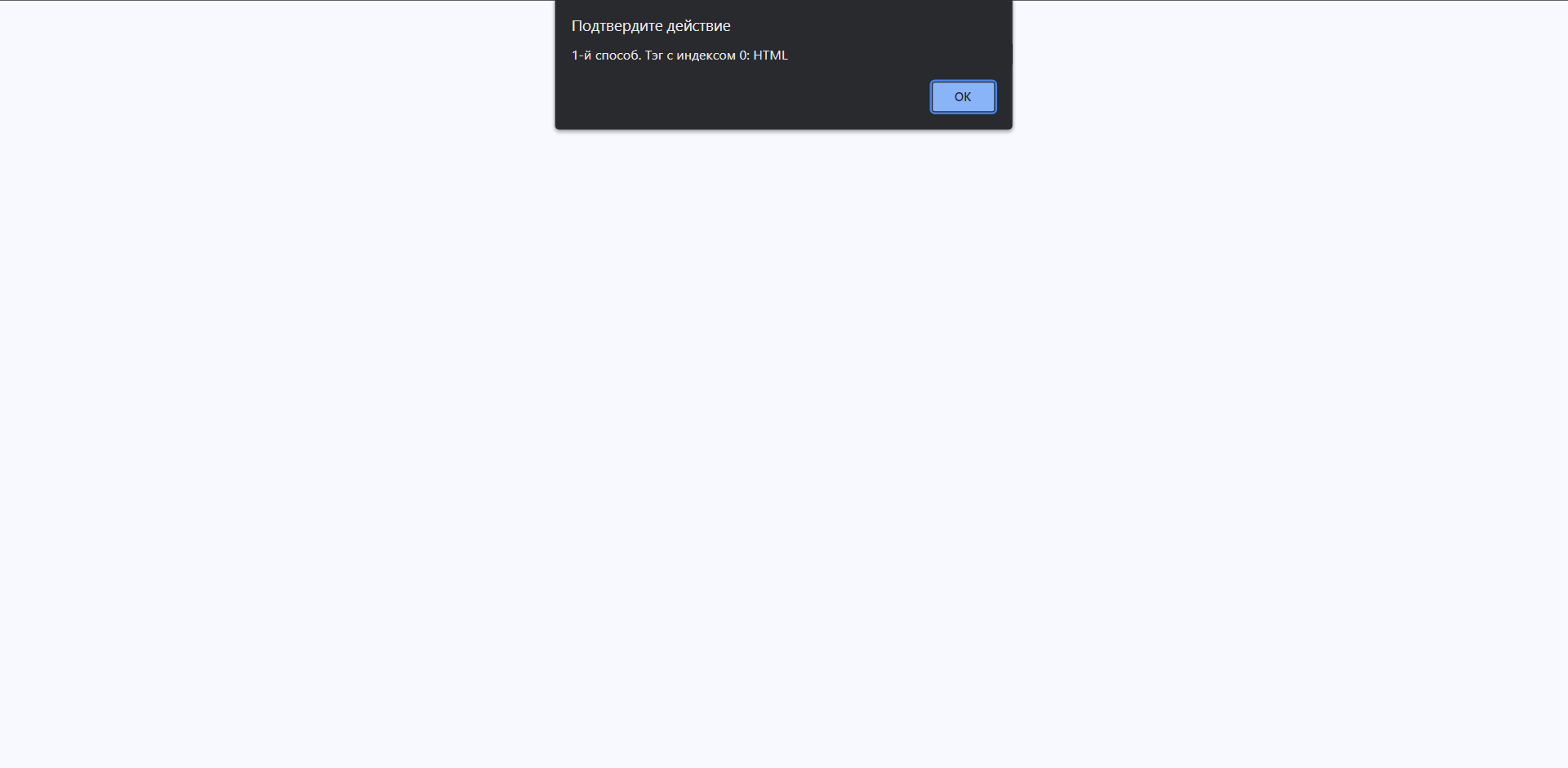


Рисунок 18 – Сообщение *alert*(), содержащее тег *HTML*-страницы и его порядковый номер

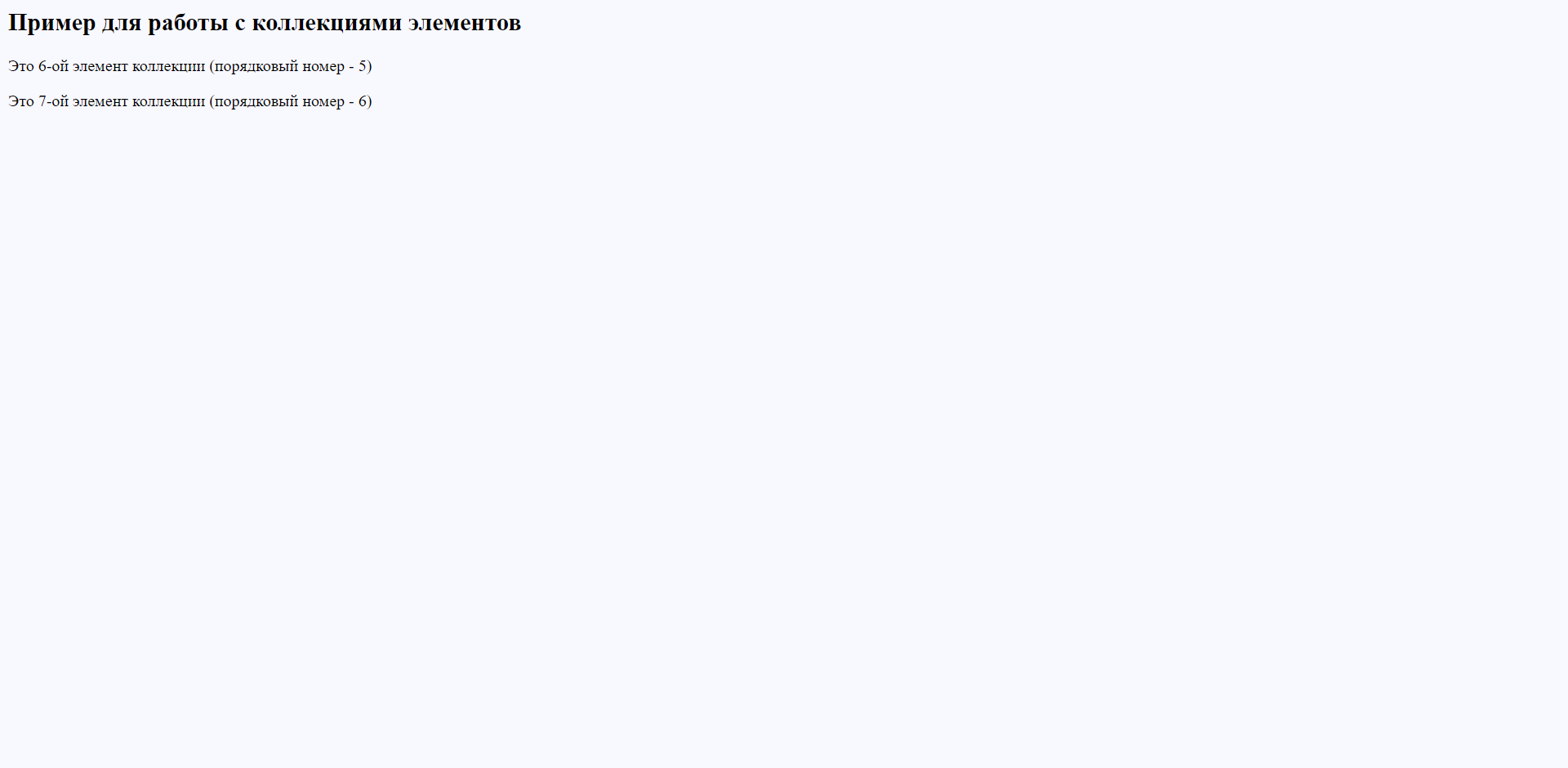


Рисунок 19 – Страница «Пример для работы с коллекциями элементов»

При нажатии на кнопку «Задание 4.6.3» открывается новая вкладка, на которой размещается страница для демонстрации работы с событиями в *DHTML*. На странице размещено *GIF*-изображение, которое изменяется при наведении и отведении курсора мыши. Эта страница представлена на рисунке 20.



Рисунок 20 – Страница для демонстрации работы с событиями в *DHTML*

При нажатии на кнопку «Задание 4.6.4» открывается новая вкладка, на которой размещается страница с периодически сменяющимися изображениями. В данном примере показан один из способов генерации новых элементов страницы (изображений). Эта страница представлена на рисунке 21.

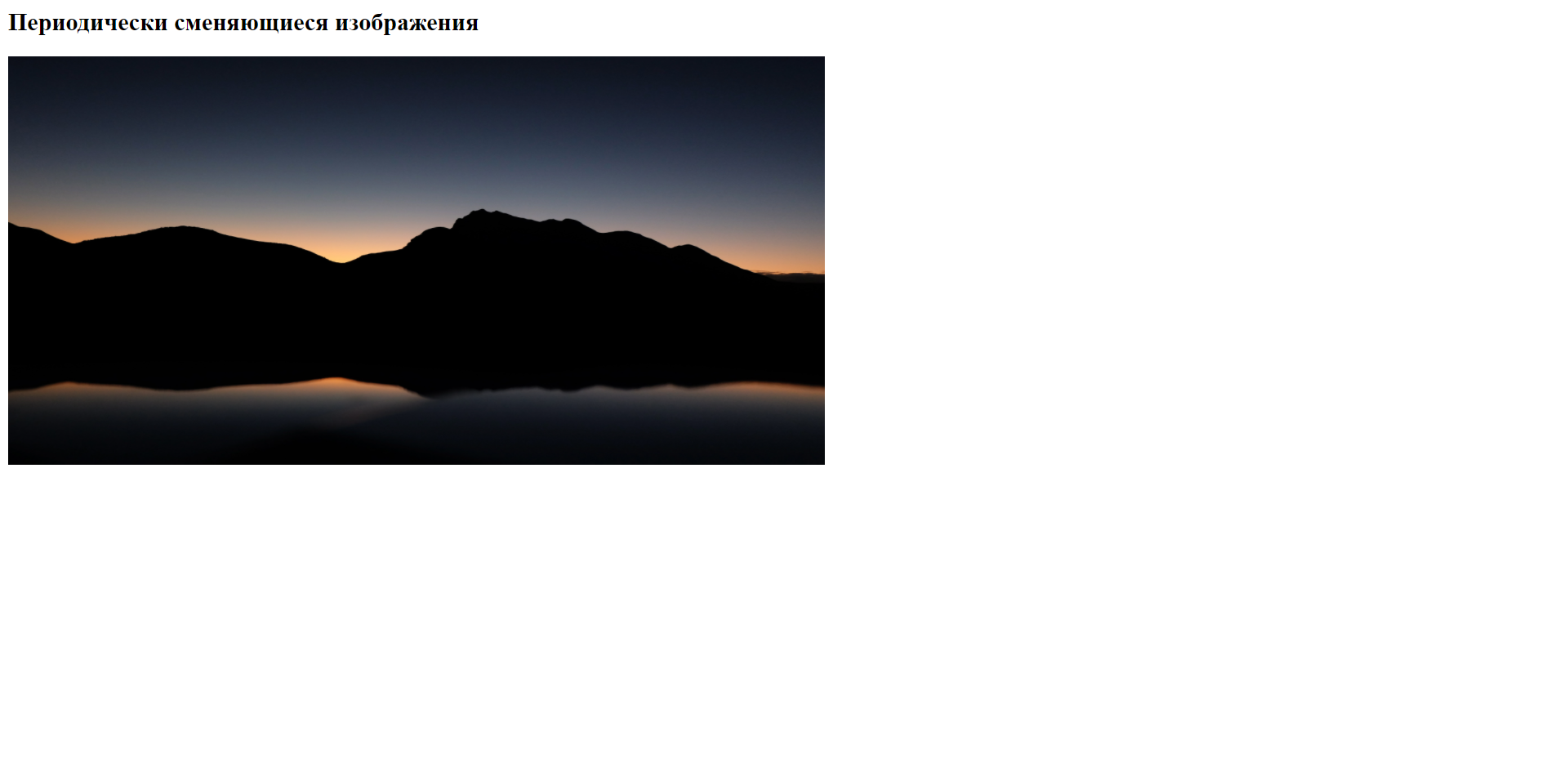


Рисунок 21 – Страница с периодически сменяющимися изображениями

* + 1. Страница «Анимация шаров в коробке» содержит *HTML*-элемент *canvas*,используемый для рисования системы моделируемых объектов в текущий момент времени. Моделируемыми объектами являются движущиеся внутри коробки шары. Шары могут упруго сталкиваться между собой без потерь энергии, мерой инертности здесь выбрана площадь поверхности шара. Кроме того, шары абсолютно упруго сталкиваются и со стенами коробки.

Создание анимации целиком описывается в скрипте *JavaScript*.

Страница «Анимация шаров в коробке» представлена на рисунке 22.

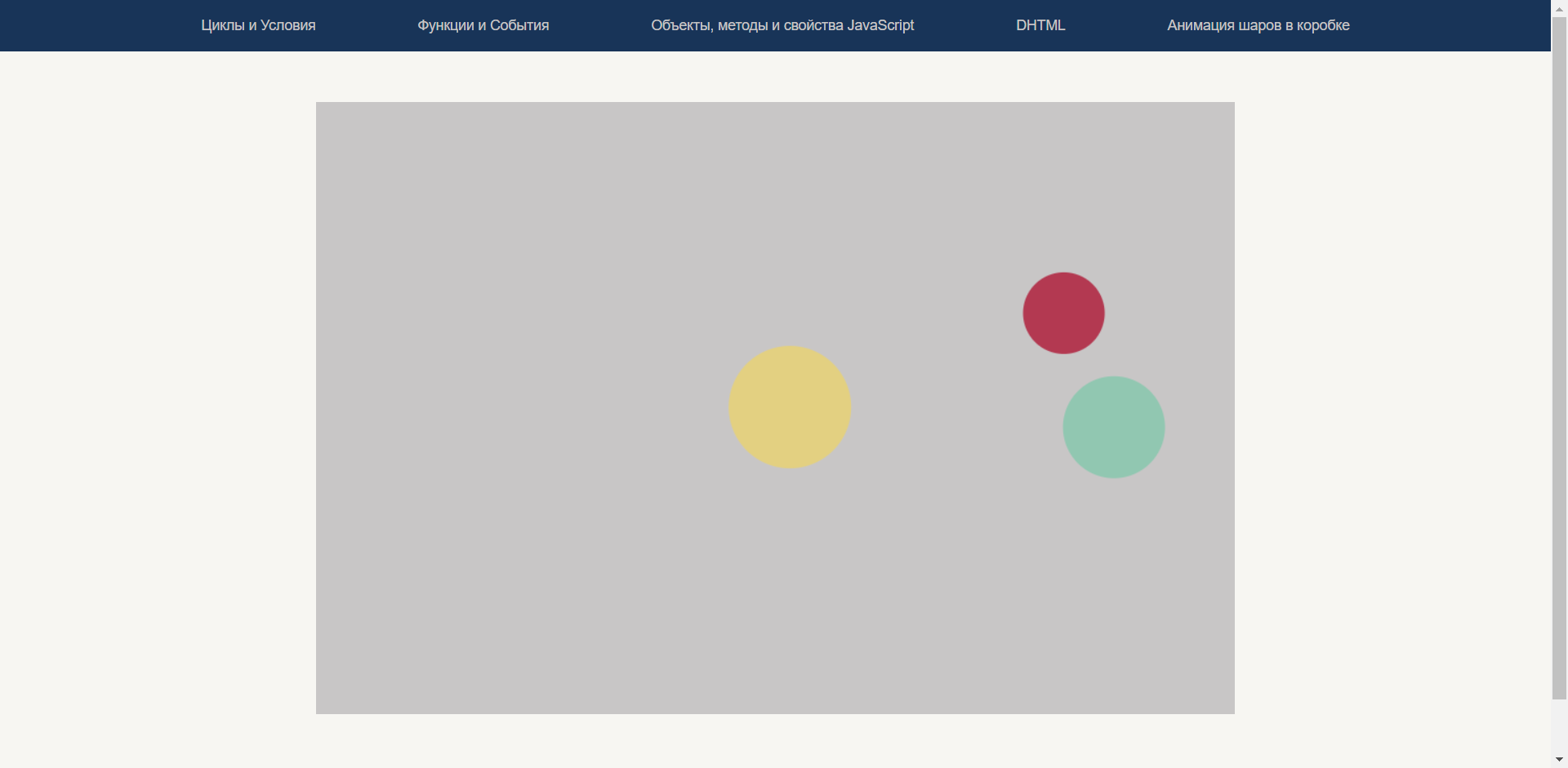


Рисунок 22 – Страница «Анимация шаров в коробке»

Заключение

В ходе выполнения данной лабораторной работы закрепил теоретические знания языка программирования *JavaScript*, а также получил дополнительные практические навыки его применения для создания анимации и динамических веб-страниц.

Список использованных источников

[1]  Флэнаган, Дэвид *JavaScript*. Подробное руководство / Дэвид Флэнаган. – М.: Питер, 2021. – 720 c.

[2]  Крокфорд, Дуглас Как устроен *JavaScript* / Дуглас Крокфорд –   
СПб. : Питер, 2019. – 304 с.

[3]  Справочник по *HTML* и *CSS* [Электронный ресурс]. – Режим доступа : *http*://*htmlbook*.*ru*/.

[4]  *HTML Tutorial* [Электронный ресурс]. – Режим доступа : *https*://*www*.*w3schools*.*com*/*html*/.