Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет

по лабораторной работе №3

«Работа с *JavaScript*»

Вариант №10

|  |  |
| --- | --- |
| Выполнила:  Дрозд В. А. | Проверил:  А. Л. Гончаревич |

Минск 2021

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc88998415)

[2 Постановка задачи 4](#_Toc88998416)

[3 Теоретические сведения 5](#_Toc88998417)

[4 Ход выполнения работы 9](#_Toc88998418)

З[аключение 21](#_Toc88998419)

1. Цель работы

Целью данной лабораторной работы является продолжение изучения языка написания сценариев *JavaScript*, в частности работа с операторами, функциями и событиями, объектами, их методами и свойствами, а также изучение возможностей *DHTML*.

1. Постановка задачи

В ходе выполнения данной лабораторной работы по *JavaScript* необходимо научиться использовать:

* оператор организации ветвлений *if*;
* операторы организации цикла *do … while*, *while*, *for* и *for … in*;
* операторы *break* и *continue*;
* оператор *switch*;
* функции, вложенные функции;
* обработчики событий (*onload*, *onclick* и другие);
* пользовательские объекты, их свойства и методы;
* встроенные объекты *JavaScript*;
* **возможности *DHTML*.**

1. Теоретические сведения

Все программы на языке *JavaScript*, независимо от их размеров и сложности, представляют собой последовательности операторов.

При необходимости выполнения различных действий в зависимости от условий используется инструкция *if* и оператор вопросительный знак. Оператор *if* вычисляет выражение в скобках и преобразует результат к логическому типу, а также позволяет выбрать и запустить на выполнение одну из альтернативных групп операторов. Необязательный блок *else* выполняется, когда условие ложно, а для проверки нескольких вариантов условия используется блок *else if*.

При написании скриптов зачастую встает задача сделать однотипное действие много раз. Например, вывести товары из списка один за другим. Или просто перебрать все числа от одного до 10 и для каждого выполнить одинаковый код. Для многократного повторения одного участка кода предусмотрены циклы.

Выражение *do...while* создаёт цикл, который выполняет указанное выражение до тех пор, пока условие не станет ложным. Условие проверяется после выполнения выражения, то есть выражение выполнится как минимум один раз.

Оператор *while* создаёт цикл, выполняющий заданную инструкцию, пока истинно проверяемое условие. Логическое значение условия вычисляется перед исполнением тела цикла. В операторе *while*, в отличие от оператора *do…while*, цикл в котором обязательно выполняется хотя бы один раз, цикл может вовсе не запускаться на выполнение в зависимости от выполнения условия.

Более сложный, но при этом самый распространенный цикл – цикл *for*. Он состоит из трех необязательных выражений в круглый скобках, разделенных точками с запятой. Начальное значение – это значение переменной, называемой параметром цикла, в исходный момент времени. Условие – это булево выражение, ложное значение которого является сигналом для прекращения цикла. Шаг – это шаг возрастания (или убывания) параметра цикла до тех пор, пока условие не примет значение *false*.

Цикл *for...in* проходит только по перечисляемым свойствам. Объекты, созданные встроенными конструкторами, такими как *Array* и *Object* имеют неперечисляемые свойства от *Object.prototype* и *String.prototype*. Цикл пройдёт по всем перечисляемым свойствам объекта, а также тем, что он унаследует от конструктора прототипа.

Обычно цикл завершается при вычислении условия в *false*. Но также есть возможность выйти из цикла в любой момент с помощью специальной директивы *break*.

Директива *continue* позволяет перейти к выполнению следующего цикла, пропустив все следующие за ним операторы текущего цикла.

Конструкция *switch* заменяет собой сразу несколько *if*. Она представляет наглядный способ сравнить выражение сразу с несколькими вариантами. Данный оператор имеет один или несколько блоков *case* и необязательный блок *default*. Каждая группа блоков *case* должна заканчиваться оператором *break*, в противном случае будут выполняться все последующие за ней операторы.

Зачастую нужно повторять одно и то же действие во многих частях программы. Например, необходимо вывести сообщение при приветствии посетителя, при выходе посетителя с сайта и где-нибудь ещё. Чтобы не повторять один и тот же код во многих местах, придуманы функции. Они являются основными строительными блоками программы. Примеры встроенных функций, таких как *alert(), prompt(), confirm()*, были рассмотрены в предыдущей лабораторной работе. Для создания функций используется объявление функции. Вначале идет ключевое слово *function*, после него имя функции, затем список параметров в круглых скобках через запятую и, наконец, код функции, также называемый телом функции, внутри фигурных скобок. Данный синтаксис носит название *function declaration*. Существует ещё один синтаксис создания функций, который называется *function expression*. В результате него созданная функция присваивается переменной, как любое другое значение. Основное отличие между ними в том, что *function declaration* можно использовать во всем скрипте, а *function expression* создается, когда выполнение доходит до него, и затем уже может использоваться.

***JavaScript* – язык, управляемый событиями.** Все происходящее при выполнении программ является результатом некоторых событий или вызывает какое-либо новое событие. Открытие новой страницы в браузере, перемещение курсора, щелчок мыши, нажатие на клавишу клавиатуры – все это относится к событиям. Каждое доступное событие имеет **обработчик событий –** блок кода, который будет запускаться при срабатывании события. Обработчик события указывается в качестве атрибута в *HTML*-тэгах, где обработчик приравнивается функции, с помощью которой в ответ на событие выполняется некоторое действие. Событие ***load*** происходит после загрузки чего-либо, например, после открытия страницы в окне браузера. Данное событие наступает, когда загрузилась вся страница, включая стили, картинки и другие ресурсы. Когда посетитель закрывает окно браузера или переходит на другую страницу, то генерируется событие *unload*. В этот момент стоит совершать простые действия, не требующие много времени, вроде закрытия связанных всплывающих окон или отсылки статистики.

В *JavaScript* существует перечень событий мыши, разделенных на две категории: простые и комплексные. Примером комплексного события является событие *click*. Оно вызывается при *mousedown*, а затем *mouseup* над одним и тем же элементом, если использовалась левая копка мыши.

К простым событиям мыши относятся следующие:

* *mousedown* – кнопка мыши нажата над элементом;
* *mouseup* – кнопка мыши опущена над элементом;
* [*mouseover*](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/Variables#%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B_objects) – курсор мыши появляется над элементом;
* [*mouseout*](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/Variables#%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B8_strings) – курсор мыши уходит с элемента;
* [*mousemove*](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/Variables#%D0%BB%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5_booleans) – любое движение мыши над элементом;
* [*context*](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/Variables#%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B_objects)*menu* – открытие контекстного меню правой кнопкой мыши;
* и многие другие.

В *JavaScript* существует восемь типов данных, семь из которых являются примитивными, так как содержат только одно значение. Объекты же используются для хранений коллекций различных значений и более сложных сущностей. Стандартными встроенными объектами языка, поддерживаемыми всеми браузерами, являются объекты *Math*, *Date* и *String*. Объект *Math* –встроенный объект, хранящий в своих свойствах и методах различные математические константы и функции. Стандартный объект *Date* содержит дату и время, а также предоставляет методы управления ими. Для создания нового объекта *Date* нужно вызвать конструктор *new Date()* с одним из следующих аргументов:

* без аргументов – создание объекта с текущими датой и временем;
* *milliseconds* – создание объекта с временем, равным количеству миллисекунд, прошедших с 1-го января 1970 года;
* [*datestring*](https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/Variables#%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D1%8B_objects) – создание объекта с датой, прочитанной из строки;
* *year, month, date, hours, minutes, seconds, ms* – создание объекта с заданными компонентами в местном часовом поясе.

Объект ***String*** представляет последовательность символов в строке и позволяет изменять и форматировать текстовую строку, выделять внутри неё часть строки, разбить строку на отдельные строки и занести их в массив и так далее.

Важным инструментом практических приложений в *JavaScript* являются формы. Они используются для ввода пользователем данных, в том числе в интерактивном режиме, для последующей обработки этих данных программой-приложением. Для интерактивного взаимодействия с компьютером посредством формы используются события и обработчики событий, а обработка данных осуществляется с помощью функций, в том числе встроенных в качестве метода в тот или иной объект. Формы в документе входят в специальную именованную коллекцию *document.forms*. Данная коллекция позволяет использовать для получения формы как ее имя, так и порядковый номер в документе. Любой элемент формы доступен в именованной коллекции *form.elements*. При этом для элемента форма доступна через *element.form*.

В форме используются следующие элементы управления:

* *input*;
* *textarea*;
* *select*;
* *option.*

*DHTML* – набор средств, который позволяет создавать более интерактивные *web*-страницы без увеличения загрузки сервера. *DHTML*построен на объектной модели документа (*DOM*), которая расширяет традиционный статический *html*-документ. *DOM* обеспечивает динамический доступ к содержимому документа, его структуре и стилям. В *DOM* каждый элемент *web*-страницы является объектом, который можно изменять. Он не определяет новых тегов и атрибутов, а просто обеспечивает возможность программного управления всеми тегами, атрибутами и каскадными таблицами стилей.

1. Ход выполнения работы

Ниже продемонстрирована работа программ всех заданий данной лабораторной работы.

В задании 3.1.1 необходимо, чтобы при выборе альтернативы выводились разные варианты сообщений с помощью трех операторов *alert()*. Окно для ввода альтернативы изображено на рисунке 4.1. Вывод самих сообщений в зависимости от введенного текста представлен на рисунках 4.2, 4.3.

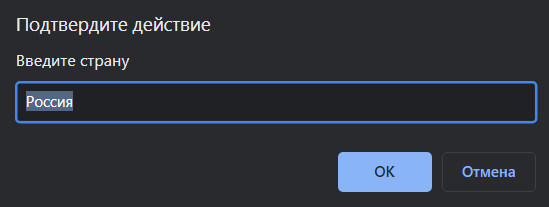


Рисунок 4.1 – Окно для ввода страны

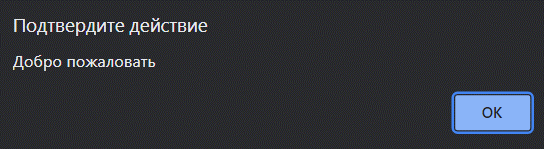


Рисунок 4.2 – Сообщение для варианта Россия

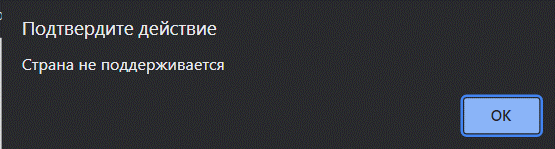


Рисунок 4.3 – Сообщение для варианта,

не предусмотренного программой

В следующем задании нужно, чтобы на экран выводились цифры от одного до пяти. Для этого нужно использовать оператор *do…while*. Последнее выводимое окно показано на рисунке 4.4.



Рисунок 4.4 – Последнее число,

выведенное в цикле *do…while*

В задании 3.1.3 требуется, чтобы в примере 3.5.3 из лекции часть *do* при *while* равном *false* выполнялось не более трех раз. Последнее выведенное окно показано на рисунке 4.5.

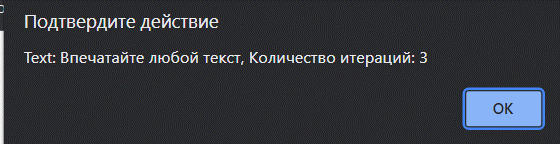


Рисунок 4.5 – Последнее окно, которое

выводится после трех выполнений цикла

Далее необходимо переписать код задания 3.1.2, но с помощью оператора *while*. Результат выполнения программы аналогичен заданию 3.1.2.

В задании 3.1.5 необходимо вывести на экран массив со значениями в порядке убывания (от девяти до нуля). Для этого использовать оператор *for*. Результат выполнения задания изображен на рисунке 4.6.

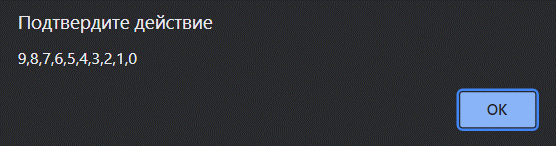


Рисунок 4.6 – Окно вывода значений

массива в порядке убывания

В задании 3.1.6 необходимо вывести на экран массив, содержащий до пяти ссылок на картинки или фотографии с расширением *jpg*. Для этого следует использовать оператор *for … in*. Результат выполнения программы представлен на рисунке 4.7.

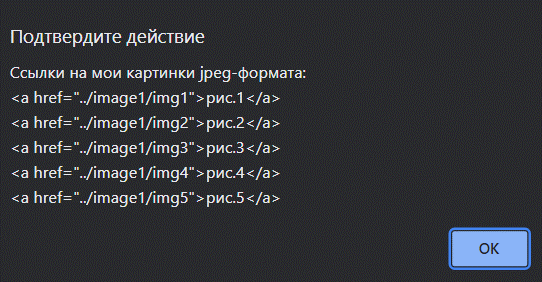


Рисунок 4.7 – Окно вывода массива ссылок

на изображения

Для задания 3.1.7 нужно написать программу для вывода только чётных чисел от нуля до 10, используя оператор *continue*. Последнее выводимое окно показано на рисунке 4.8.

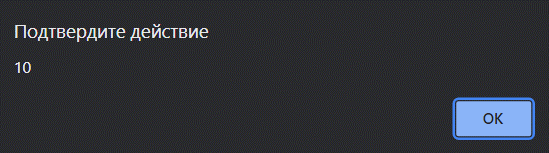


Рисунок 4.8 – Последнее выведенное число

В задании 3.1.8 требуется написать программу, в которой пользователь вводит число, при котором следует прервать вывод чисел на экран. При этом следует использовать оператор *break*. Результат выполнения программы аналогичен рисунку из предыдущего задания.

В следующем задании нужно написать программу, в которой будут выполняться различные операторы в зависимости от введенной в окне запроса буквы: *a*, *b*, *c*, *d* или *e*. Требуется воспользоваться оператором *switch*.

Окно для ввода буквы изображено на рисунке 4.9.

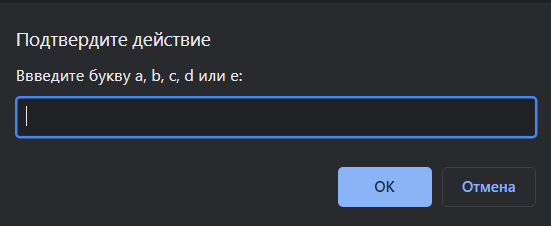


Рисунок 4.9 – Окно для ввода буквы

Вывод сообщений в зависимости от введенной буквы представлен на рисунках 4.10, 4.11.

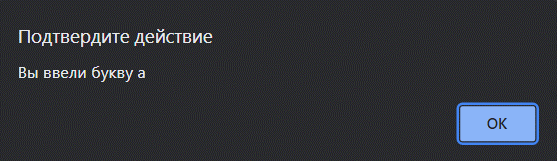


Рисунок 4.10 – Окно сообщения для варианта

ввода буквы *а*

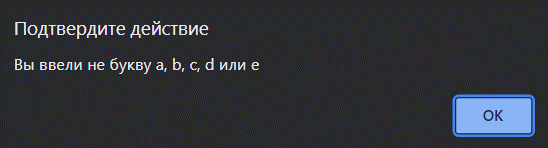


Рисунок 4.11 – Окно сообщения для варианта

ввода непредвиденного значения

Для выполнения задания 3.1.10 необходимо создать вариант защиты страницы от нежелательных посещений с помощью пароля. Посетитель страницы вынужден, прежде чем он получит доступ к информации, ввести пароль в окне, созданном с помощью метода *prompt*. Если пароль неверен, должно выводиться соответствующее сообщение и загружаться дополнительная страница, на которой помещается лишь та информация, к которой разрешен доступ без пароля. Посетителю дается три попытки для ввода пароля.

Окно для ввода пароля представлено на рисунке 4.12.

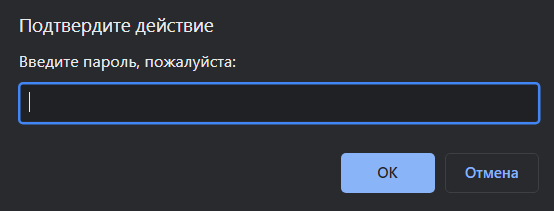


Рисунок 4.12 – Окно для ввода пароля

При вводе неверного пароля на экран выводится предупреждение, изображенное на рисунке 4.13.

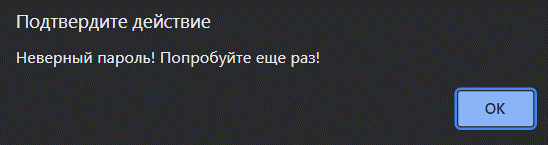


Рисунок 4.13 – Окно предупреждения

при вводе неверного пароля

При трехкратном вводе неверного пароля пользователь перенаправляется на дополнительную страницу, к которой разрешен доступ без пароля. Она представлена на рисунке 4.14.

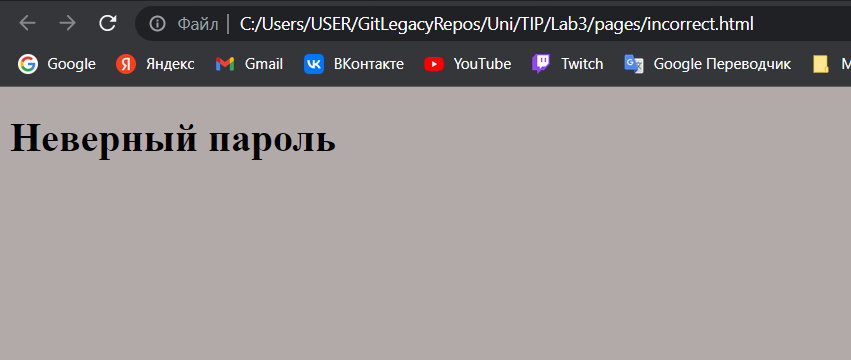


Рисунок 4.14 – Дополнительная страница,

доступ к которой разрешен без пароля

Если введен верный пароль, на экран выводится окно с сообщением, изображенным на рисунке 4.15.

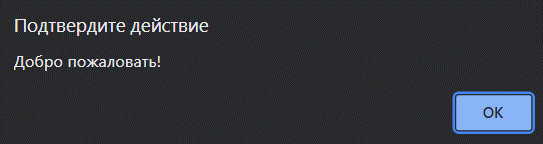


Рисунок 4.15 – Сообщение о вводе

верного пароля

Для задания 3.2.1 нужно создать страницу с функцией, которая будет выполняться после того, как пользователь уберет курсор с гиперссылки. При этом должна загружаться страница со списком примеров и заданий.

Результат выполнения программы представлен на рисунке 4.16.

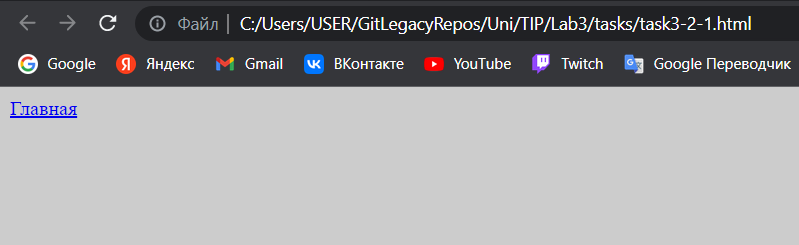


Рисунок 4.16 – Пример выполнения функции

в ответ на событие *mouseout*

В следующем задании на основе примера 3.6.9 необходимо создать свой сценарий, дополнив сценарий из примера еще четырьмя функциями и кнопками переводов: Цельсий-Реомюр, Фаренгейт-Реомюр, Реомюр-Цельсий, Реомюр-Фаренгейт. Кнопки должны активизировать добавленные функции после двойного щелчка мышью с помощью соответствующего события.

При двойном щелчке мыши на любой из кнопок выводится окно для ввода температуры для расчета, как это показано на рисунке 4.17.

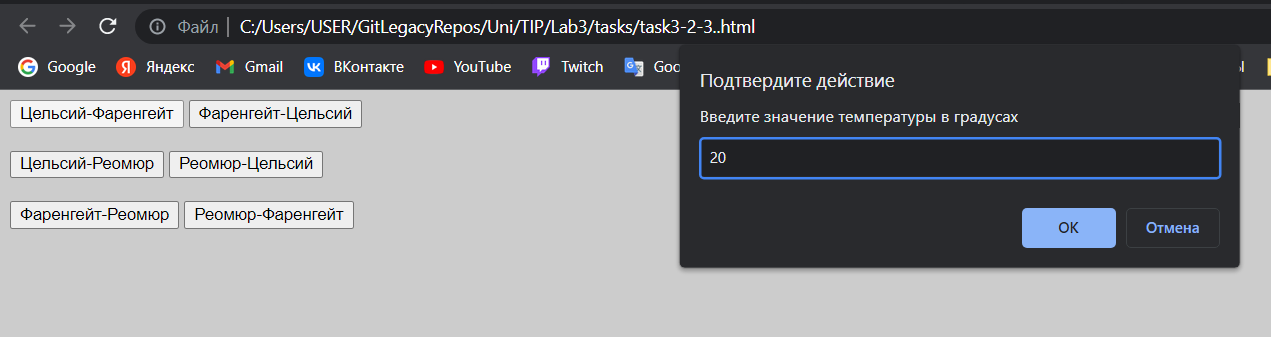


Рисунок 4.17 – Кнопки и окно для ввода

значения температуры

После ввода температуры и нажатия кнопки *ОК* выводится сообщение с переводом указанного значения температуры из одной шкалы в другую. Пример такого сообщения изображен на рисунке 4.18.

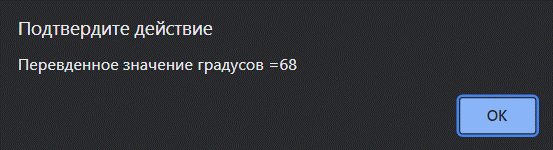


Рисунок 4.18 – Сообщение с переводом

температуры в другую шкалу

В задании 3.3.1 необходимо дополнить пример 3.7.1 из [лекции 16](file:///D:\%D1%83%D0%BD%D0%B8%D0%B2%D0%B5%D1%80\4%20%D0%BA%D1%83%D1%80%D1%81\%D0%A2%D0%98%D0%9F\%D0%AD%D0%A0%D0%A3%D0%94_%D0%A2%D0%98%D0%9F\Теория\content\Part1\lk1_16\lk1_16.htm) вычислением псевдослучайного числа в промежутке от нуля до 100, которое затем, предварительно округлив до целого, нужно использовать в качестве радиуса для вычисления площади круга. Само число и соответствующая площадь круга представлены на рисунке 4.19.

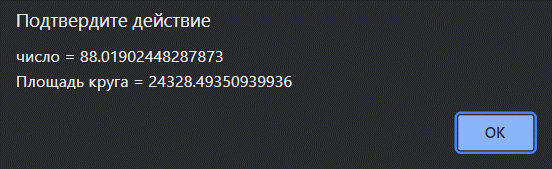


Рисунок 4.19 – Псевдослучайное число от нуля

до единицы и соответствующая площадь круга

В следующем задании необходимо выполнить расчет и вывод на экран даты, которая наступит через неделю без учета времени. Результат выполнения программы показан на рисунке 4.20.

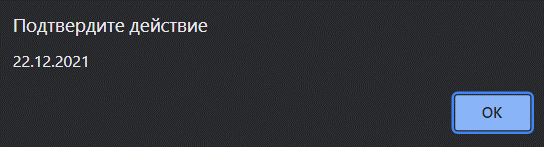


Рисунок 4.20 – Окно с датой, которая

наступит через неделю

В задании 3.3.3 требуется написать собственный сценарий для определения среднемесячного дохода за шестимесячный период с указанием данных о себе. Для обращения к значениям, передаваемым из формы следует использовать свойство *elements* частной коллекции *forms*.

Результат выполнения программы показан на рисунке 4.21.

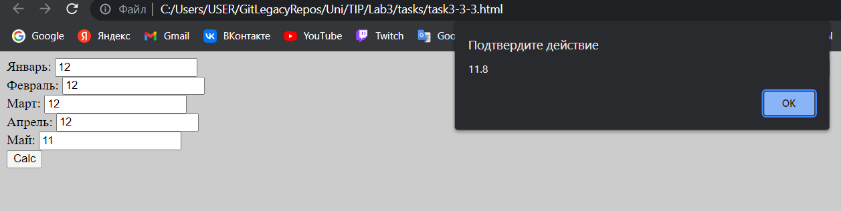


Рисунок 4.21 – Форма определения дохода

Для выполнения задания 3.4.1 на основе примера 3.8.4 необходимо усложнить программу, введя равносторонний треугольник, а также возможность расчета площадей в других единицах. Результат программы для равностороннего треугольника со стороной пять сантиметров показан на рисунке 4.22.

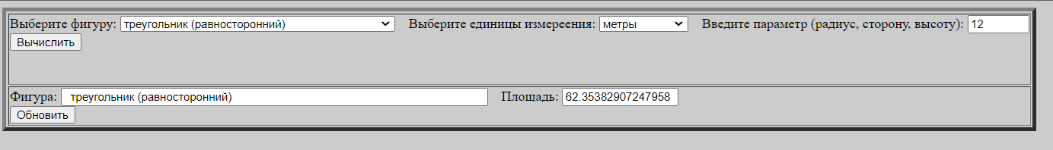


Рисунок 4.22 – Вычисление площади

равностороннего треугольника

В следующем задании нужно добавить в пример 3.10.3 вычисление и вывод максимального и минимального месячного дохода и соответствующих им названий месяцев, используя данные за 12 месяцев.

Результат выполнения программы показан на рисунке 4.23.

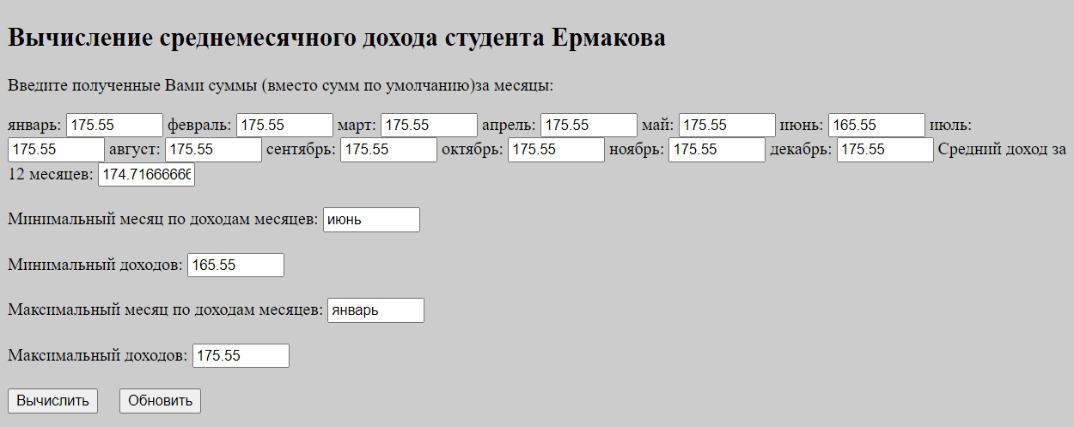


Рисунок 4.23 – Форма определения среднемесячного,

минимального и максимального доходов

В следующем задании продемонстрированы возможности *DHTML*. Анимационный фрагмент, изображенный на рисунке 4.24, представляет собой перемещение слева направо и вращение блока.

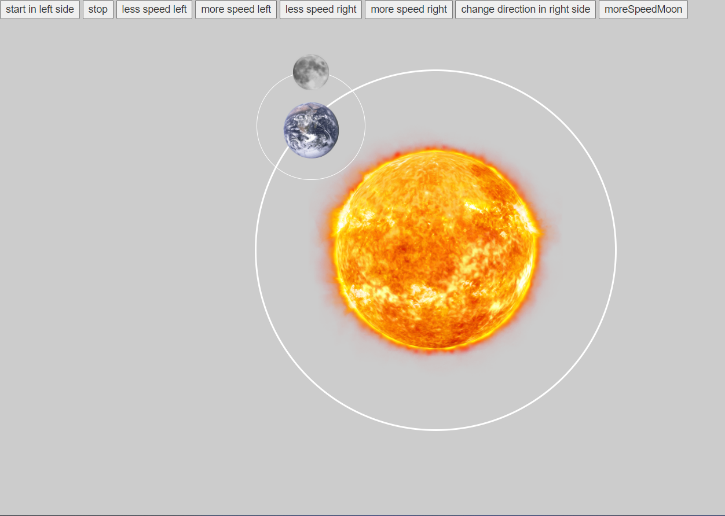


Рисунок 4.24 – Демонстрация

возможностей *DHTML*

В центре блока находятся Солнце и Земля, вращающаяся вокруг Солнца по окружности. На странице имеются кнопки для ускорения спутника, ускорения и вращения слоя, а также кнопки для остановки этих эффектов.

В задании 3.6.1 необходимо, чтобы на экран выдавалась информация о тэгах, имеющихся в коде примера. Одно из окон, выводимых в результате выполнения задания, представлено на рисунке 4.25.

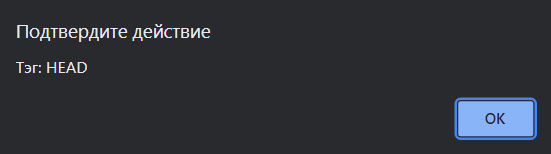


Рисунок 4.25 – Вывод информации

об одном из тегов

В задании 4.6.3 необходимо изменить код примера 4.2.3 таким образом, чтобы в качестве сменяемого изображения использовались собственные изображения. Для этого используется события мыши *mouseout* и *mouseover*. Результат выполнения программы представлен на рисунках 4.26.

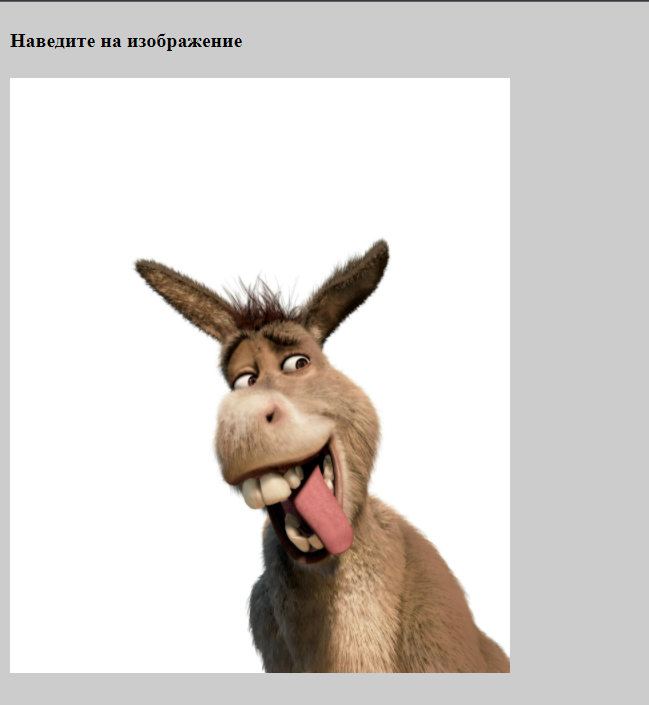


Рисунок 4.26 – Использование событий

*mouseout* и *mouseover*

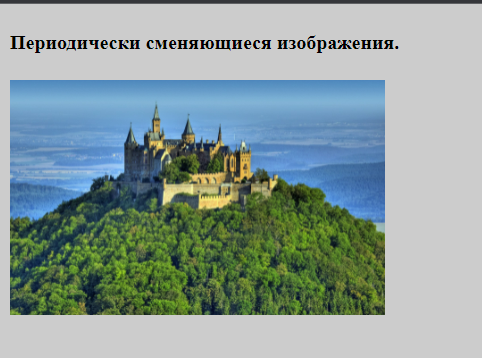
В следующем задании требуется воспроизводить изображения циклически через определенный интервал. Изображения предварительно загружаются, что позволяет исключить задержки показа объявлений в процессе работы кода. Помимо этого, обрабатывается случай возникновения ошибок и отсутствия изображений. Одно из таких объявлений изображено на рисунке 4.27.

Рисунок 4.27 – Изображение замка

В следующем задании необходимо осуществить анимацию с взаимодействием объектов. Для этого была создана игра «Змейка». Процесс игры изображен на рисунке 4.28.

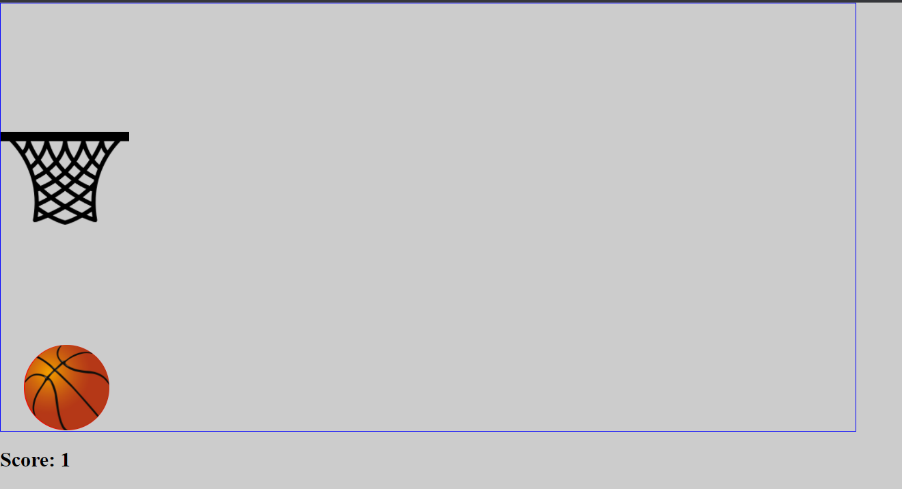


Рисунок 4.28 – Пример анимации с

взаимодействием

заключение

В ходе выполнения данной лабораторной работы я продолжила изучение языка написания сценариев *JavaScript*. Мной была освоена работа с различными операторами, обработчиками событий, функциями, встроенными объектами, их методами и свойствами. Также были изучены и использованы возможности динамического *HTML*.