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*»

|  |  |
| --- | --- |
| Проверил | Выполнил |
| Гончаревич А. Л. | ст. гр. №820602 |
|  | А. Д. Бондарчук |

Минск 2021

**СОДЕРЖАНИЕ**

Введение 3

1 Цель работы 4

2 Теоретическая часть 5

2.1 Циклы в *JavaScript* 5

2.2 Условные конструкции в *JavaScript* 6

2.3 События *DOM* 7

2.4 Получение псевдослучайных чисел 8

2.5 Дата в *JavaScript* 9

2.6 Работа с формами 9

3 Практическая часть 10

3.1 Описание используемых программных средств 10

3.2 Описание назначений страниц 10

3.3 Руководство пользователю 14

Заключение 15

Список использованных источников 16

# Введение

*JavaScript* был представлен в 1995 году как способ добавления программ на веб-страницы в браузере *Netscape Navigator*. С тех пор этот язык был принят другими основными браузерами. Это сделало возможными современные веб-приложения – приложения, с которыми можно взаимодействовать напрямую, не перезагружая страницу для каждого действия. *JavaScript* также используется на более традиционных веб-сайтах для обеспечения различных форм интерактивности [1].

Важно отметить, что *JavaScript* почти не имеет ничего общего с языком программирования *Java*. Изначально первый имел название *Mocha*. Брендан Эйх создал его в 1992 году. В это же время компания *Sun* *Microsystems* заканчивала разработку *Java*. Эйх договорился с ними – и *Mocha* стал скриптовым компаньоном языка *Java*. Позднее *Mocha* был переименован в *LiveScript*, что также не принесло языку славы. Тогда, воспользовавшись популярностью своего напарника, разработчики дали ему название *JavaScript*.

*JavaScript* используют в основном в веб-разработке. Вместе с *HTML* и *CSS*, *JavaScript* создаёт идеальный набор для фронтенд-разработчика.

В веб-разработке *JavaScript* помогает обрабатывать любое действие пользователя – от заполнения формы до прокрутки страницы. В браузере срабатывает событие, которое запускает работу *JavaScript*-кода, только после этого на странице происходит изменение. На каждом сайте используется несколько скриптов, которые собирают данные посетителей, анализируют метрики и позволяют зарегистрировать аккаунт.

*JavaScript* используют также для следующих целей:

– разработка нативных приложений;

– бэкенд-разработка;

– создание ПО для техники;

– разработка десктопных приложений;

– создание расширений для браузера.

*JavaScript* – постоянно развивающийся язык с продуманной инфраструктурой и большой экосистемой. У этого языка нет конкурентов в его основной сфере применения, он справляется со всеми поставленными задачами. Поэтому сейчас нет необходимости в создании альтернативного варианта – он достаточно быстр, удобен и универсален.

Также для этого языка существуют фреймворки, в которых есть предварительно написанный код для использования в функциях и задачах программирования. Наиболее популярны *React*, *Angular*, *Vue*.

# 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 отобразить демонстрацию работы с операторами, функциями, событиями, объектами, их методами и свойствами, а также с формами в *JavaScript*. Изучить основы *DHTML*, создать собственную анимацию

# Теоретическая часть

## Циклы в *JavaScript*

### Цикл *for* повторяет действия, пока не произойдёт какое-либо специальное событие завершения цикла. Оператор *for* в *JavaScript* аналогичен оператору *for* в *Java* и *C*. Объявление данного оператора происходит с указанием начальных значений итерационных переменных, а также объявлений этих переменных при необходимости, условия, при котором будут выполняться выражения тела цикла, и операторов, изменяющих значения итерационных переменных. Тело цикла оборачивается фигурными скобками.

### Цикл *do*...*while* повторяется пока заданное условие истинно. Сперва записывается оператор *do*. После следуют фигурные скобки, в которые помещаются выражения. Завершается запись оператора ключевым словом *while* с указанием в круглых скобках условия повторения цикла. Важно отметить, что тело цикла выполняется до проверки условия.

### Цикл *while* выполняет выражения пока условие истинно. Объявление этого оператора представляет указание ключевого слова *while*, после которого следует условное выражение, обрамлённое круглыми скобками. Тело цикла помещается в фигурные скобки. Условие проверяется на истинность до того, как выполняются выражения. В этом состоит отличие данного цикла от цикла *do*...*while*.

### Метка представляет собой оператор с идентификатором, который позволяет ссылаться на какое-то место в программе [2]. Значение метки может быть любым корректным *JavaScript*-идентификатором, не являющимся зарезервированным словом. После значения метки ставится символ двоеточия, после которого указывается любое выражение. В ходе выполнения программы есть возможность переключить управление, указав значение нужной метки.

### Оператор *for*...*in* проходит по всем свойствам объекта. *JavaScript* выполнит указанные выражения для каждого отдельного свойства. Этот оператор записывается с использованием ключевого слова *for*, после которого следуют круглые скобки с указанием названия ключа, ключевого слова *in* и итерируемого объекта. С помощью данного цикла можно вывести названия свойств, используя переменную-ключ, и их значения некоторого объекта.

### Оператор *for*...*of* создаёт цикл, проходящий по перечислимым объектам, включая *Array*, *Map*, *Set*, объект *arguments*, вызывая на каждой итерации функцию с выражениями, которые надо выполнить для получения значения каждого отдельного свойства. Запись оператора происходит таким же образом, как и запись *for*...*in*, но с указанием ключевого слова *of* вместо *in*. Разница между этим циклом и рассматриваемым ранее заключается в том, что *for*...*of* проходит не по именам, а по значениям свойств [2].

## Условные конструкции в *JavaScript*

В зависимости от результата вычисления некоторого логического выражения можно указать различные пути выполнения программы при помощи условных операторов *JavaScript*.

### Наиболее распространённым типом условного оператора является оператор *if* *else*. При использовании данного оператора сперва записывается ключевое слово *if*, после которого в круглых скобках указывается логическое выражение. Далее в фигурных скобках располагается поток действий, выполняемых, если результатом вычисления логического выражения является истина. В части *else* располагается поток действий, выполняемых при противоположном результате вычисления выражения. При отсутствии альтернативных действий последнюю часть можно опустить. Если требуется указать дополнительные варианты выполнения программы, то есть возможность расположить несколько частей *else* *if* с дополнительными условиями между основными блоками *if* и *else*.

### Оператор *switch* принимает единственное выражение, которое указывается в круглых скобках, а затем просматривает ряд вариантов, представленных блоками, начинающимися с ключевого слова *case* со сравниваемыми выражениями, пока не будет найден вариант, соответствующий этому значению, после чего выполняет код, назначенный этому варианту. Есть возможность указать набор действий, выполняемых в случае, когда ничего не соответствует входному выражению. Для этого используется ключевое слово *default*, но без указания выражения для сравнения со входным. Для того, чтобы не допустить так называемого сквозного выполнения операторов, требуется завершать каждый блок варианта ключевым словом *break*, если бизнес-логика противоречит такому поведению. Всё тело оператора *switch* заключается в фигурные скобки.

### Тернарный оператор имеет простой синтаксис: он проверяет условие и возвращает значение или выражение, если условие истинно, и другое значение или выражение, если условие ложно. Часто это очень удобная альтернатива оператору *if else*, позволяющая затрачивать меньшие усилий на написание кода, когда имеется всего лишь пара вариантов, выбираемых на основе значения условного выражения. Сперва указывается условие. Для удобства чтения его можно поместить в круглые скобки. Затем следуют символ вопросительного знака, выражение, выполняемое, если результат вычисления условного выражения – истина, символ двоеточия, выражение, выполняемое в противоположном случае.

## События *DOM*

Событие – это сигнал от браузера о том, что что-то произошло. Все *DOM*-узлы могут подавать такие сигналы. События могут возникать в результате таких действий пользователя, как использование мыши или изменение размера окна, изменения состояния базовой среды, например, низкий заряд батареи или мультимедийные события из ОС.

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

Существуют следующие способы назначения обработчиков событий:

– атрибут *HTML*;

– свойство *DOM*-объекта;

– метод *addEventListener*.

### Обработчик может быть назначен прямо в разметке, в атрибуте, который имеет название, состоящее из названия события и приставки *on*. В качестве значения используемого атрибута можно использовать *JavaScript*-код, заключённый в двойные кавычки, при этом важно учитывать, что если в коде используются строковые значения, то их необходимо записывать между одинарными кавычками. Данный подход не удобен, так как код обработчика может быть довольно объёмным, поэтому лучше создать отдельную *JavaScript*-функцию и указать в качестве значения обработчика её вызов.

### Можно назначать обработчик в *JavaScript*-коде, используя свойство *DOM*-элемента, название которого состоит из приставки *on* и названия события. Если обработчик задан через атрибут, то браузер читает *HTML*-разметку и создаёт новую функцию из содержимого атрибута и записывает в свойство [2]. Этот способ аналогичен предыдущему. Обработчик хранится в свойстве, а атрибут – лишь способ его инициализации.

### Фундаментальный недостаток описанных выше способов назначения обработчика – невозможность повесить несколько обработчиков на событие. Например, часть кода при нажатии на кнопку будет делать её подсвеченной, а другая – выдавать сообщение. Для этого можно назначить пару обработчиков. Но новое *DOM*-свойство перезапишет предыдущее при присваивании ему другого обработчика. Разработчики стандартов предложили альтернативный способ назначения обработчиков при помощи специальных методов *addEventListener* и *removeEventListener*. Метод *addEventListener* позволяет добавить обработчик события. В качестве параметров он принимает название события в виде строки и ссылку на функцию-обработчик, то есть название функции. Метод *removeEventListener* позволяет удалить обработчик события. Ему передаются те же параметры. Важно отметить, что для удаления обработчика должна быть передана в точности та же ссылка на функцию, что использовалась и при регистрации обработчика. Таким образом, если для добавления обработчика использовалась стрелочная функция, то передав такую же функцию в метод удаления обработчика, он не будет удалён. Для решения этой проблемы необходимо сохранить используемую стрелочную функцию в отдельной переменной и использовать её в качестве параметра вместо самой функции.

## Получение псевдослучайных чисел

Метод *Math*.*random* возвращает псевдослучайное число с плавающей запятой из диапазона [0, 1), то есть, включая нижнюю границу, но не включая верхнюю. Реализация сама выбирает начальное зерно для алгоритма генерации случайных чисел – оно не может быть выбрано или сброшено пользователем. Если же необходимо получить случайное число в заданном интервале, то для этого можно умножить результат рассматриваемого метода на разность максимального и минимального значений и прибавить минимальное значение интервала. Таким образом, будем получено случайное число, которое находится в диапазоне от минимального значения с и до максимального значения, но не включая последнее. Чтобы получить целое случайное число из заданного интервала, то достаточно в последнем варианте округлить результат произведения с помощью метода *Math*.*floor*.

## Дата в *JavaScript*

Для работы с датой в *JavaScript* используются объекты класса *Date*. Для создания нового объекта типа *Date* используются перегруженные конструкторы класса. Например, будет создан объект с текущим временем и датой, если не передавать параметры. Также можно передать значение, которое представляет количество миллисекунд, прошедших с «эпохи *Unix*». Создание объекта даты возможно и с использованием компонентов в местной временной зоне. Для этого формата обязательны только аргументы, которые представляют год и месяц. Чтобы получить отдельные компоненты объекта даты применяются методы *getFullYear*, *getMonth*, *getDate*, *getHours*, *getMinutes*, *getSeconds*, *getMilliseconds*. Дополнительно можно получить день недели при помощи метода *getDay*.

## Работа с формами

Веб-формы, которые также часто называют *HTML*-формами, состоят из нескольких элементов управления форм, называемых виджетами, и некоторых дополнительных элементов для структурирования формы. Элементами управления могут быть однострочные или многострочные текстовые поля, выпадающие списки, кнопки, флажки, переключатели, которые создаются с помощью *HTML*-элемента *input* с атрибутом type, имеющим соответствующие значения. Элемент *form* определяет форму. Он является элементом-контейнером, как *HTML*-элементы *div* или *p*, но при этом он поддерживает некоторые специфические атрибуты для настройки поведения формы. Все атрибуты являются опциональными, но в стандартной практике принято указывать атрибуты *action*, который определяет адрес, куда должны быть посланы данные после отправки формы, и *method*, который указывает, какой *HTTP*-метод будет использован при передаче данных, например, *get* или *post*. Нажатие кнопки с атрибутом *type*, имеющим значение *submit* отправляет данные из формы на страницу, определённую в атрибуте *action* элемента. Наличие такой кнопки, атрибутов *action* и *method* у элемента *form* минимально определяет форму [3].

# Практическая часть

## Описание используемых программных средств

### *Visual* *Studio Code* – редактор исходного кода, разработанный *Microsoft* для *Windows*, *Linux* и *macOS*. Позиционируется как лёгкий редактор кода для кроссплатформенной разработки. Включает в себя отладчик, инструменты для работы с *Git*, подсветку синтаксиса, *IntelliSense* и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Также к установке доступны различные расширения. Например, чтобы не обновлять созданную *HTML*-страницу в браузере после каждого редактирования кода, можно установить расширение *Live Server*, и открыть страничку с помощью контекстного меню. Тогда страница будет автоматически обновляться после каждого изменения. *Visual Studio Code* распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

### *Notepad*++ – бесплатный редактор исходного кода, являющийся альтернативой блокноту или любому другому редактору, поддерживающий множество языков. Работа в среде *MS Windows* регулируется Стандартной общественной лицензией *GNU*. Основанный на мощном компоненте редактирования *Scintilla*, *Notepad* ++ написан на *C* ++ и использует чистый *Win*32 *API* и *STL*, что обеспечивает более высокую скорость выполнения и меньший размер программы. Базовыми возможностями данного редактора являются подсветка синтаксиса, сворачивание кода, автодополнение и автоматическое закрытие скобок и тэгов, сравнение файлов, менеджер проектов, карта документа, переопределение любых горячих клавиш, блоковое выделение текста, многострочное редактирование.

## Описание назначений страниц

### Страница «Задания 3.1», приведённая на рисунке 1, содержит кнопки, запускающие демонстрации выполнения заданий. При наведении на каждую кнопку, она изменит свой цвет, и на экране появится вопросительная надпись. Вверху находится навигационная панель, с помощью которой можно осуществлять перемещение по страницам сайта. Данная панель содержится на всех страницах.

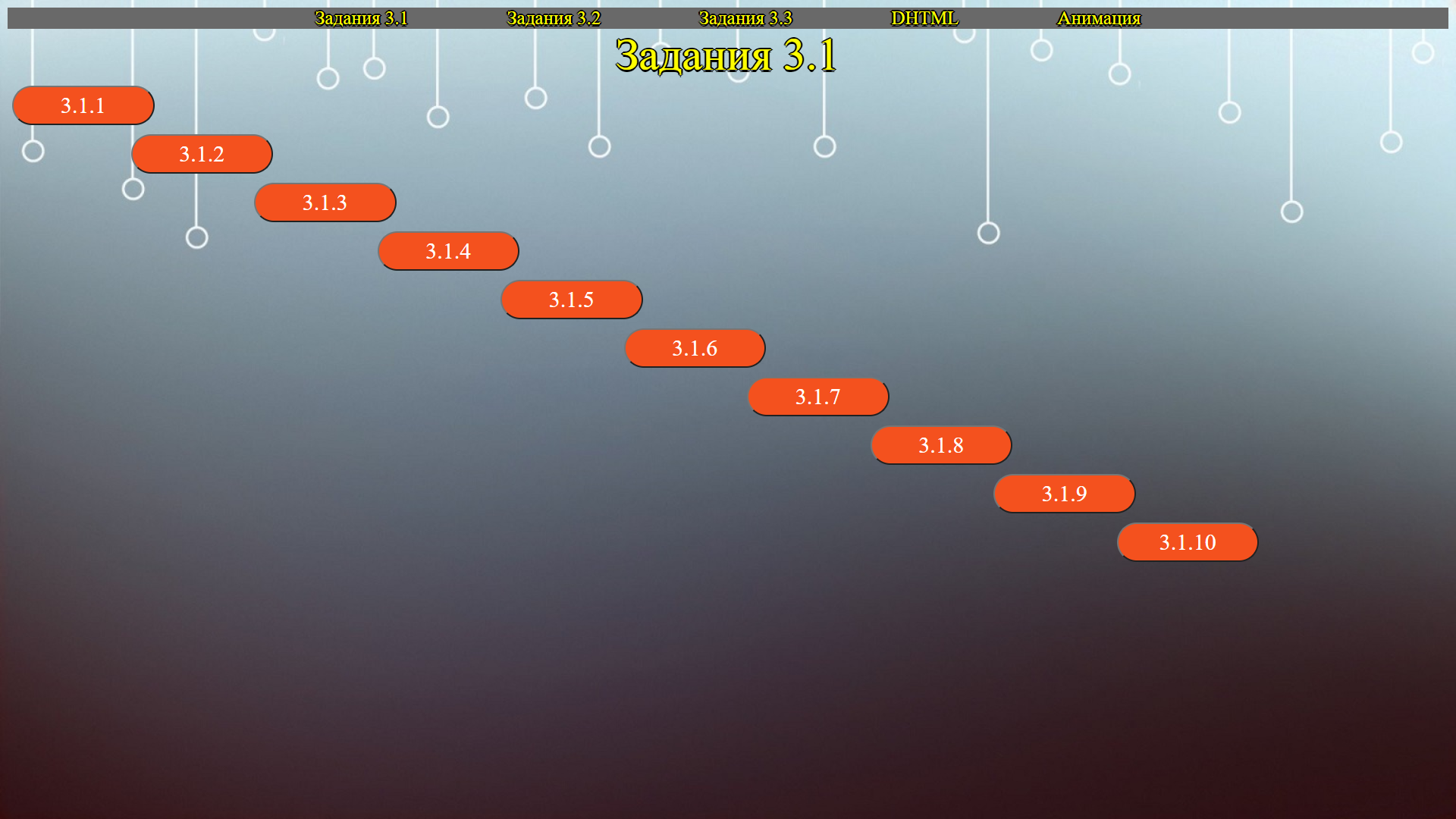


Рисунок 1 – Страница «Задания 3.1»

### На странице «Задания 3.2», изображённой на рисунке 2, есть области с текстом, представляющие единицы измерения температуры, и области со стрелками и кнопками, изменяющими цвет при наведении курсора на них и запускающими функции перевода величины из той, которая находится в начале стрелки, в ту, которая находится в конце стрелки. Также есть ссылка, которая запускается после отведения курсора с неё.

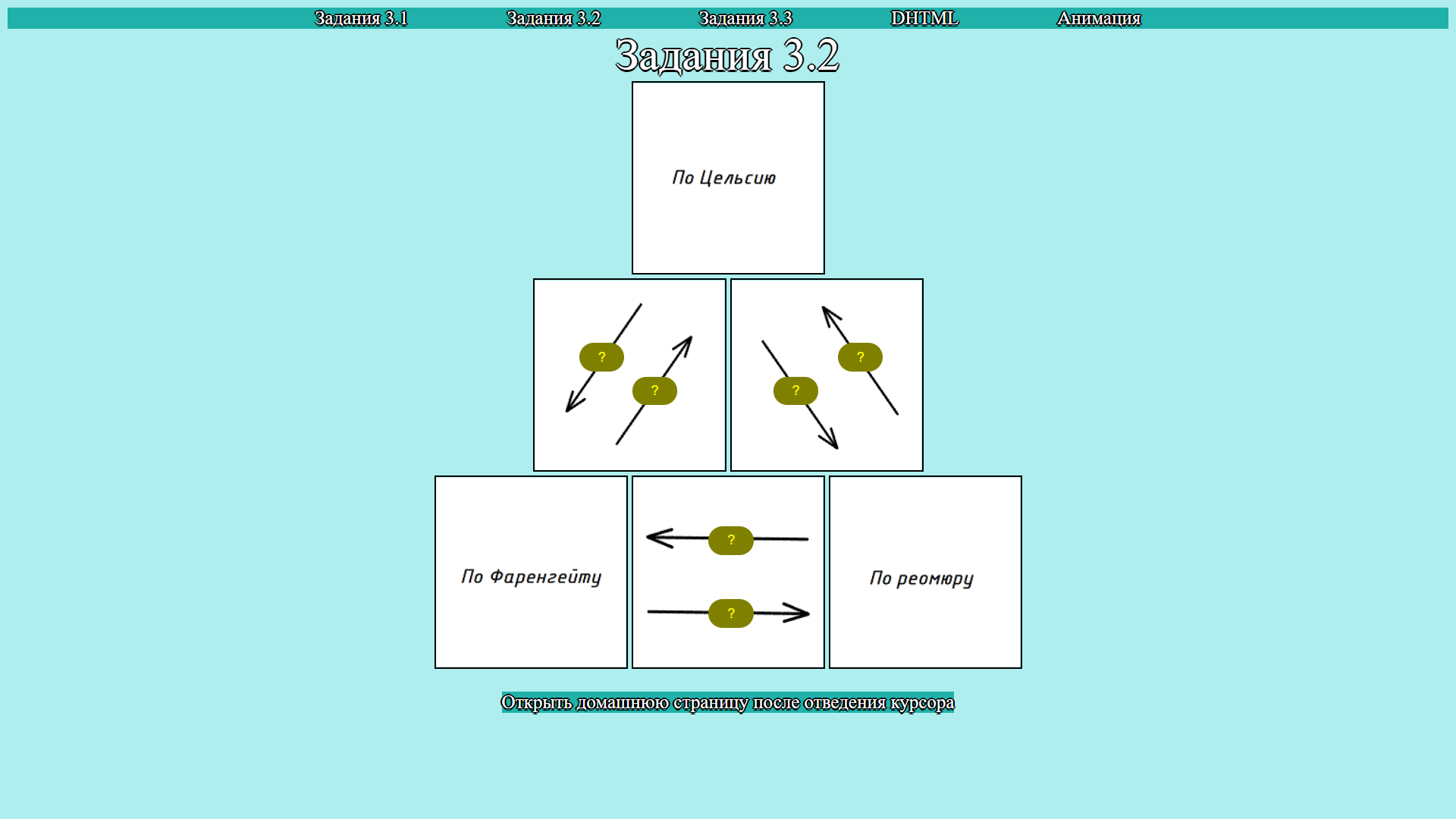


Рисунок 2 – Страница «Задания 3.2»

### На странице «Задания 3.3», представленной на рисунке 3, располагаются кнопки и формы. Кнопки «Задание 3.3.1» и «Задание3.3.2» запускают функции, вычисляющие площадь круга с радиусом равным целому псевдослучайному числу и дату через неделю от текущей даты соответственно. Ниже находится форма, с помощью которой пользователь может вычислить среднемесячный доход по значениям доходов всех месяцев. За данной формой следует другая, с помощью которой вычисляются площади некоторых фигур, тип которых устанавливается с помощью выпадающего списка, а параметр – текстового поля. В нижней части страницы есть форма для расчёта минимального и максимального дохода по значениям доходов месяцев года.

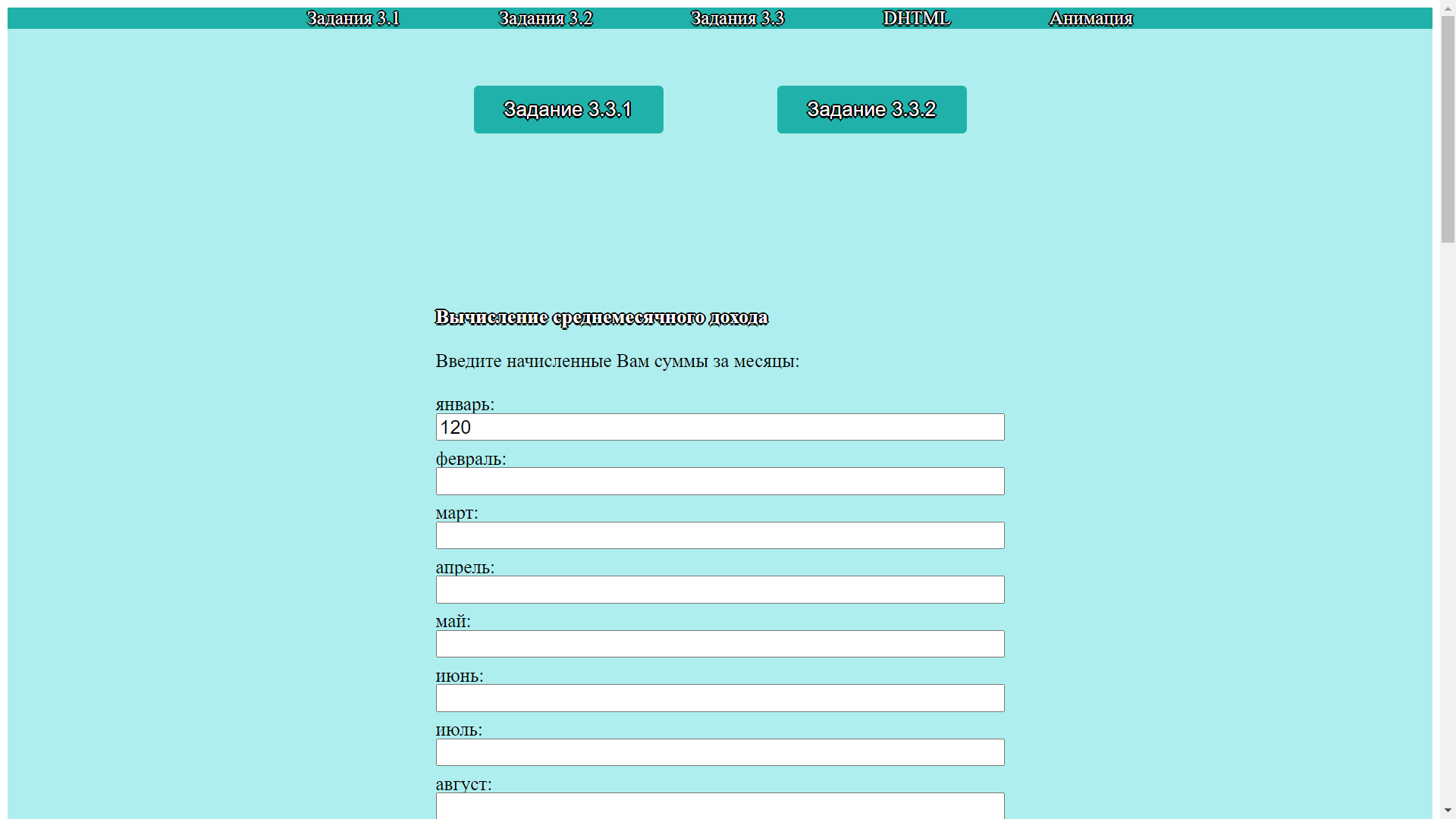


Рисунок 3 – Страница «Задания 3.3»

### Страница «*DHTML*», изображённая на рисунке 4, содержит кнопки для управления анимацией: с помощью них можно запустить движение спутника, начать движение системы вправо, сбросить положение планеты до исходного, остановить спутник или движение системы; кнопки, демонстрирующие выполнение отдельных заданий, и область, в которой отображаются анимированные объекты. Кнопки «Задание 3.6.1», «Задание 4.6.3» и «Задание 4.6.4» запускают функции, которые выводят используемые в документе теги, загружают отдельные страницы, на которых реализована работа с картинками: смена изображения при наведении курсора на неё и последовательная смена нескольких изображений.

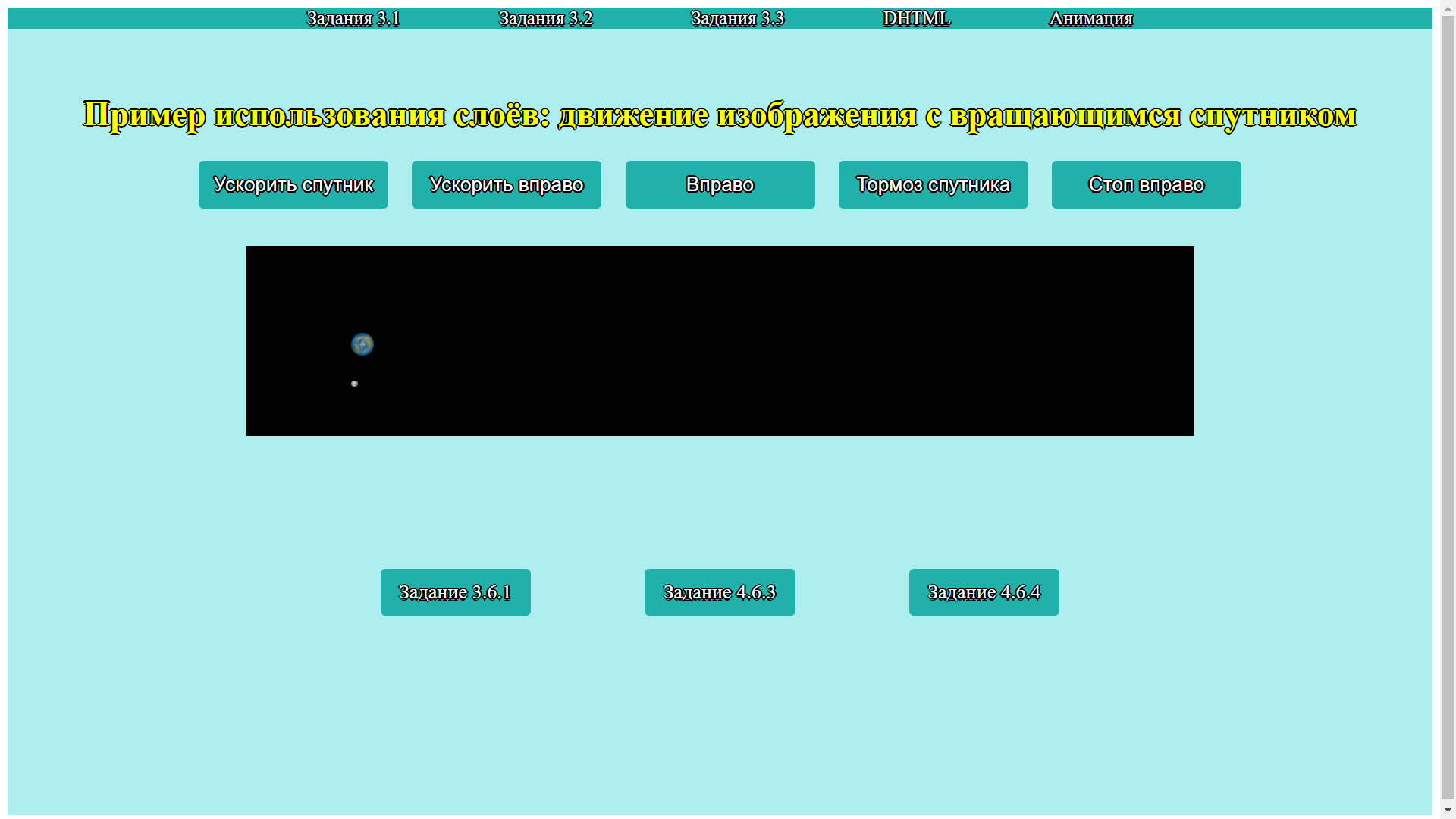


Рисунок 4 – Страница «*DHTML*»

### Пользовательская анимация находится на странице «Анимация», которая приведена на рисунке 5. На ней демонстрируется модель атома, в которой несколько электронов вращаются с разными скоростями по эллиптическим траекториям вокруг ядра. В зависимости от расстояния до ядра электроны изменяют свой цвет. Оттенок ядра зависит от того, находится ли хотя бы какой-нибудь из электронов рядом с ним.

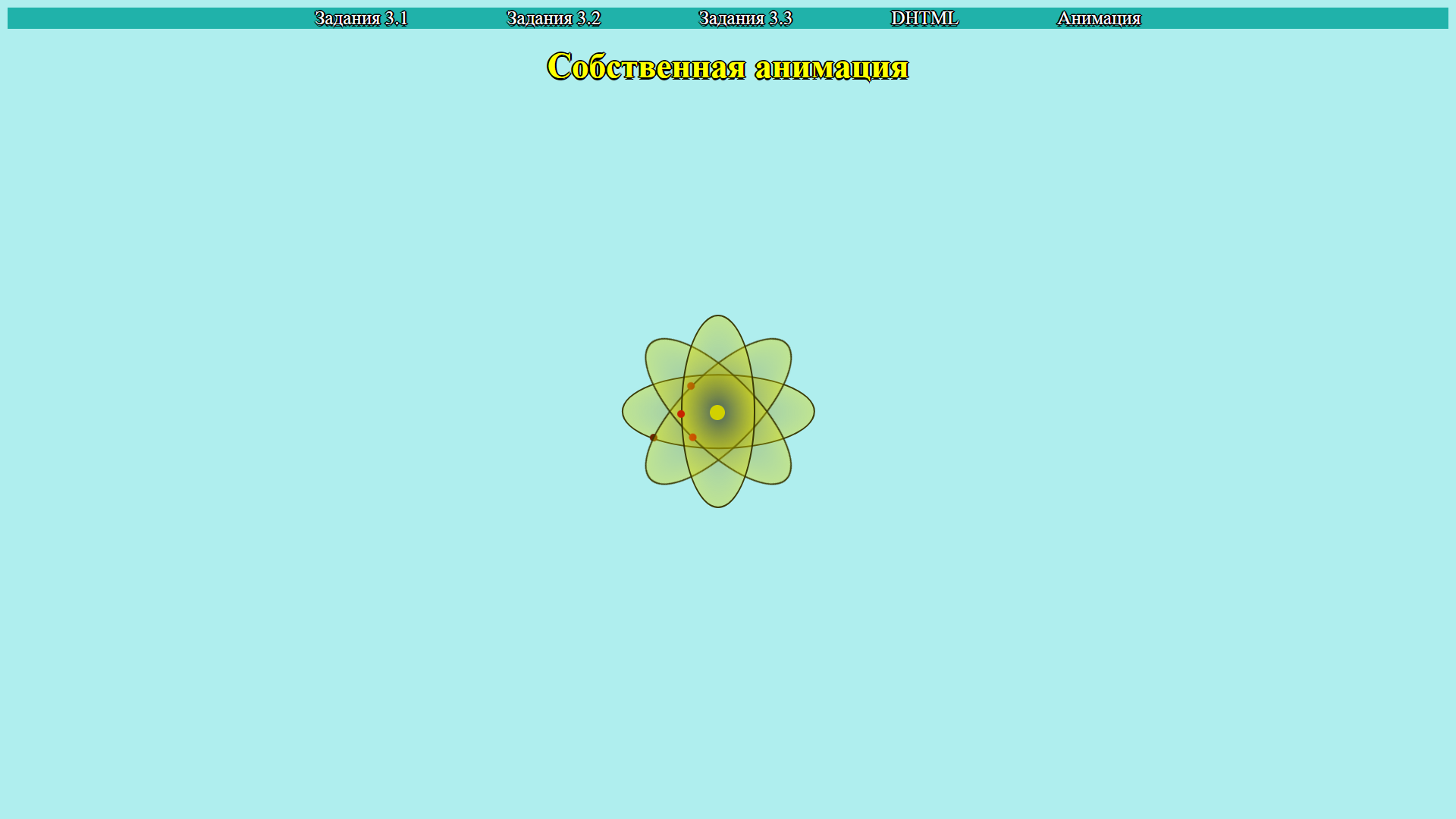


Рисунок 5 – Страница «Анимация»

## Руководство пользователю

Перемещение по сайту осуществляется с помощью переходов по ссылкам, расположенным в навигационной панели, которая присутствует на всех страницах. Демонстрация выполнения того или иного задания на странице «Задания 3.1» запускается с помощью кнопки с номером необходимого задания. Также при наведении курсора на кнопку появится всплывающая подсказка, где описано условие задания.

При переходе по ссылке «Задания 3.2» открывается страница, на которой реализован функционал перевода температуры из исходных в целевые единицы измерения, которые определяются направлениями стрелок. Например, если необходимо перевести значение, заданное в градусах по Цельсию, в градусы по Фаренгейту, то следует выбрать кнопку, расположенную на стрелке, которая направлена от области «По Цельсию» к области «По Фаренгейту». Окно для ввода значения отобразится пользователю при двойном нажатии на необходимую кнопку. На странице расположена гиперссылка, которая запускается при отведении курсора с неё. Современные браузеры блокируют запуск подобных ссылок, поэтому, например, в браузере *Google Chrome* в правом верхнем углу появится сообщение о том, что всплывающее окно заблокировано.

Чтобы получить площадь круга с радиусом равным целому псевдослучайному числу, следует перейти по ссылке «Задания 3.3» и нажать кнопку «Задание 3.3.1». Если требуется получить дату через неделю от текущей, то следует нажать кнопку «Задание 3.3.2». На данной странице находится форма «Вычисление среднемесячного дохода». Пользование ей осуществляется при помощи полей для ввода значений доходов и кнопки «Вычислить». У пользователя есть возможность очистить все поля при помощи кнопки «Обновить». Для определения площадей таких фигур, как круг, квадрат, прямоугольный равнобедренный и равносторонний треугольники, необходимо указать значение стороны квадрата, радиуса круга или высот треугольников в зависимости от выбранного вида фигуры в форме «Вычисление площадей некоторых фигур» и нажать на кнопку «Вычислить». Очистка всех полей формы производится с помощью кнопки «Обновить».

Для управления анимацией на странице «*DHTML*» следует нажимать на кнопки, расположенные под заголовком. Просмотр пользовательской анимации осуществляется на странице «Анимация», содержащая модель атома, представленную несколькими электронами и ядром. Также можно перейти на другие страницы, используя навигационную панель.

# Заключение

Закреплены знания языка программирования *JavaScript*, а также знания о возможностях его применения на веб-страницах для изменения их содержимого и создания анимации. Созданы новые страницы сайта, на которых отображена демонстрация работы с операторами, функциями, событиями, объектами, их методами и свойствами, а также с формами в *JavaScript*. Изучены основы *DHTML* и создана анимация, предложенная в варианте задания.

# Список использованных источников

[1] *Eloquent JavaScript* / *Marijn Haverbeke*. – *San Francisco*, *US*, 2018.

[2] *MDN Web Docs* [Электронный ресурс]. – Режим доступа : *https://developer.mozilla.org/.*

[3] *JavaScript for* .*NET* *Developers* / *Ovais Mehboob Ahmed Khan. – Birmingham*, *UK*, 2016.