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 №4

«*DHTML*»

|  |  |
| --- | --- |
| 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 *DHTML* 5

2.2 Графические фильтры 6

2.3 *Fetch API* 7

3 Практическая часть 8

3.1 Описание используемых программных средств 8

3.2 Описание назначений страниц 8

3.3 Руководство пользователю 12

Заключение 13

Список использованных источников 14

# Введение

*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*.

*DHTML* – это набор средств, которые позволяют создавать интерактивные веб-страницы без увеличения нагрузки на сервер. Таким образом, определённые действия пользователя ведут к изменениям внешнего вида и содержания страницы без обращения к серверу [1].

# Цель работы

Закрепить знания языка программирования *JavaScript* и познакомиться с возможностями его применения на веб-страницах для изменения их содержимого. Систематизировать знания об объектных моделях браузера и документа. Создать сайт, на страницах которого продемонстрировать работу с *DHTML*, используя полученные навыки.

# Теоретическая часть

## *DHTML*

*Dynamic HTML* или *DHTML* – это подход к созданию интерактивного веб-сайта, который использует сочетание статичного языка разметки *HTML*, скриптового языка *JavaScript*, языка каскадных таблиц стилей *CSS* и объектной модели документа *DOM*.

Динамическая веб-страница представляет собой широкую концепцию, которая может охватывать любую веб-страницу, сгенерированную персонально для каждого пользователя. Эта концепция включает в себя страницы, созданные с помощью скриптов на стороне клиента и страницы, созданные с помощью скриптов на стороне сервера, где веб сервер генерирует разметку, отправляемую клиенту.

При использовании *DHTML* может вовсе отсутствовать взаимодействие между клиентом и сервером после загрузки страницы, так как вся обработка происходит в *JavaScript* на стороне клиента. Страница *AJAX* использует функции *DHTML* для отправки запроса серверу для выполнения дополнительных действий. Например, если на странице имеется несколько вкладок, *DHTML* будет подгружать содержимое всех вкладок и динамически отображать только ту, которую необходимо отобразить в данный момент, в то время как *AJAX* подгружает каждую из вкладок только тогда, когда это необходимо [2].

*DHTML* позволяет авторам добавлять на страницы эффекты, которые трудно достичь с помощью объектной модели документа или каскадной таблицы стилей. Например, на сайтах интернет-магазинов с помощью *DHTML* пользователь может выбирать товары, задавать вопросы и получать ответы, создавать заявки и оформлять заказы. И всё это осуществляется при помощи запрограммированных на определённое реагирование скриптов.

С помощью *DHTML* можно создавать сайты-презентации, слайд-шоу, проводить рекламные промоакции, развлекать и увлекать потенциальных клиентов. Разумеется, *DHTML* придумывали не для игр, а именно для решения практических коммерческих задач, для которых в обычном сайте требовалось наличии живого менеджера для ответов на вопросы посетителей.

Однако *DHTML* сегодня уже прошлый век. В настоящее время для создания динамических сайтов применяется *HTML* 5. Этот инновационный язык гипертекста представляет собой своего рода компиляцию наработок всех предыдущих версий *HTML*.

## Графические фильтры

Графические фильтры – это эффекты изменения внешнего вида графики и текста на странице. С помощью фильтров можно отражать тексты и графику, создавать эффект движения элемента и осуществлять с ним некоторые другие преобразования, как это делается в графических редакторах. Фильтр можно описать как трансформацию исходного элемента по определённым правилам.

Существуют статические и динамические фильтры. Статические фильтры просто изменяют внешний вид элемента, оставляя его неподвижным. Динамические фильтры позволяют изменять графический элемент со скоростью, задаваемой пользователем.

Статический фильтр можно задать как *CSS*-свойство. При наличии параметров можно в скобках через запятую перечислить их.

Наиболее часто используются следующие фильтры:

– *blur*, создаёт эффект размытости;

– *fliph*, создаёт горизонтальное отражение картинки или текста;

– *flipv*, создаёт вертикальное отражение картинки или текста;

– *wave*, создаёт «волнистое» искажение картинки или текста;

– *light*, осветляет объект;

– *xrау*, показывает только контур объекта;

– *alpha*, задаёт различные эффекты прозрачности;

– *grау*, делает объект чёрно-белым;

– *brightness*, делает объект ярче.

К динамическим фильтрам относят *BlendTrans* и *RevealTrans* – динамические фильтры смешивания и появления объекта соответственно.

Переход *RevealTrans* может быть применён к любому объекту для создания анимации его появления, при этом устанавливается тип перехода и его продолжительность. Переход такого типа можно задать как свойство в таблице стилей.

Фильтры перехода имеют следующие методы и свойства:

– *apply*(), задействуется для применения фильтра;

– *play*(), применяется для запуска фильтра;

– *stop*(), используется для остановки применения фильтра;

– *duration*, отражает текущую продолжительность фильтра;

– *status*, представляет значение в зависимости от состояния перехода.

Фильтр *BlendTrans* может быть применён к объекту, чтобы «проявить» его или «затушить», через определённый период времени. Фильтр такого типа можно задать как свойство в таблице стилей либо часть атрибута style.

## *Fetch API*

*Fetch API* предоставляет интерфейс *JavaScript* для работы с запросами и ответами по протоколу *HTTP*. Он также предоставляет глобальный метод *fetch*(), который позволяет относительно легко получать ресурсы асинхронно по сети.

Подобная функциональность ранее достигалась с помощью *XMLHttpRequest*. *Fetch* представляет собой лучшую альтернативу, которая может быть легко использована другими технологиями. *Fetch API* является более мощным и гибким набором функций, обеспечивающим единое логическое место для определения других связанных с *HTTP* понятий, например, *CORS*.

*Fetch* обеспечивает обобщённое определение объектов *Request* и *Response*, а также некоторых других, связанных с сетевыми запросами. Это позволяет применять *API* практически везде, в том числе и для *Service* *Workers*, *Cache* *API* и других подобных технологий, которые обрабатывают или изменяют запросы и ответы. Также предоставляются определения таких понятий, как *CORS* и семантика заголовков *HTTP*, заменяющие их обобщённые определения.

Чтобы создать запрос и получить данные, используется метод *GlobalFetch*.*fetch*(). Он реализован во множестве интерфейсов, в том числе и в *window*. Это позволяет использовать его практически в любом контексте для получения данных [3].

Метод *fetch*() принимает обязательный аргумент – путь к данным, которые необходимо получить. Он возвращает объект *promise*, который преобразуется в *response* независимо от того, был ли запрос удачным. В другом аргументе можно передавать объект с указанием опций запроса, он является необязательным.

Когда извлечение данных выполнено успешно, становятся доступными несколько методов для определения тела контента и способов обработки его содержимого.

# 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

### На странице «Задание 1», изображённой на рисунке 1, находится набор из нескольких фреймов, для каждого из которых создан отдельный *HTML*-документ с заголовком и кратким описанием смысла освоенного подраздела лекции.

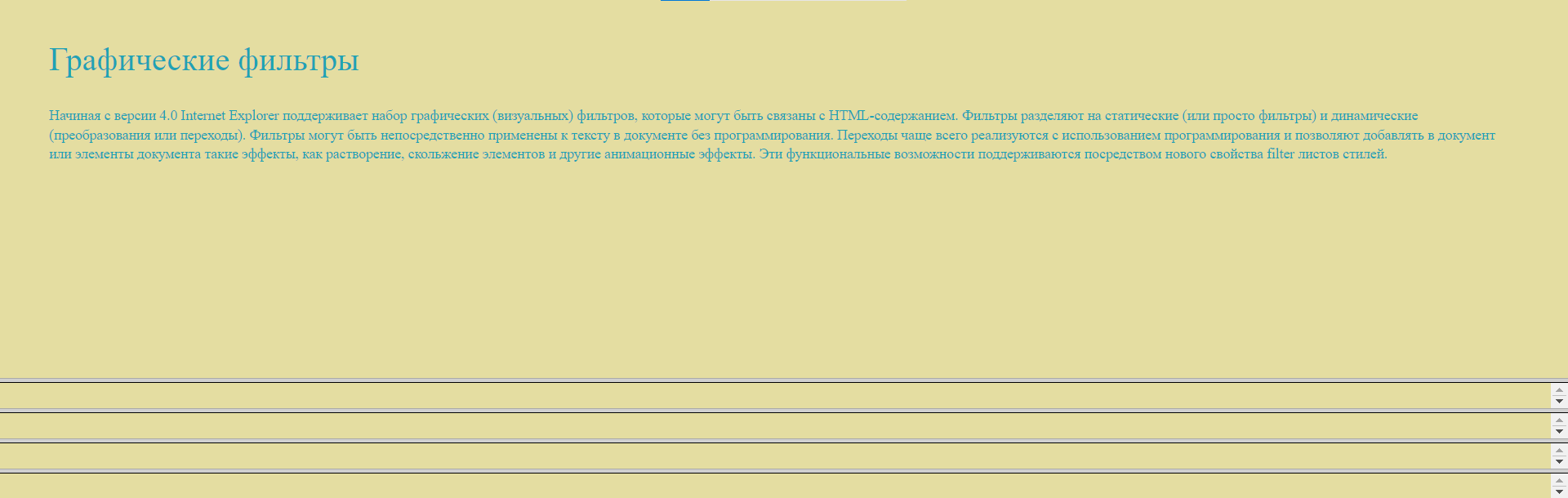


Рисунок 1 – Страница «Задание 1»

### Страница «Задание 2», приведённая на рисунке 2, содержит кнопки, запускающие демонстрации работы графических фильтров.

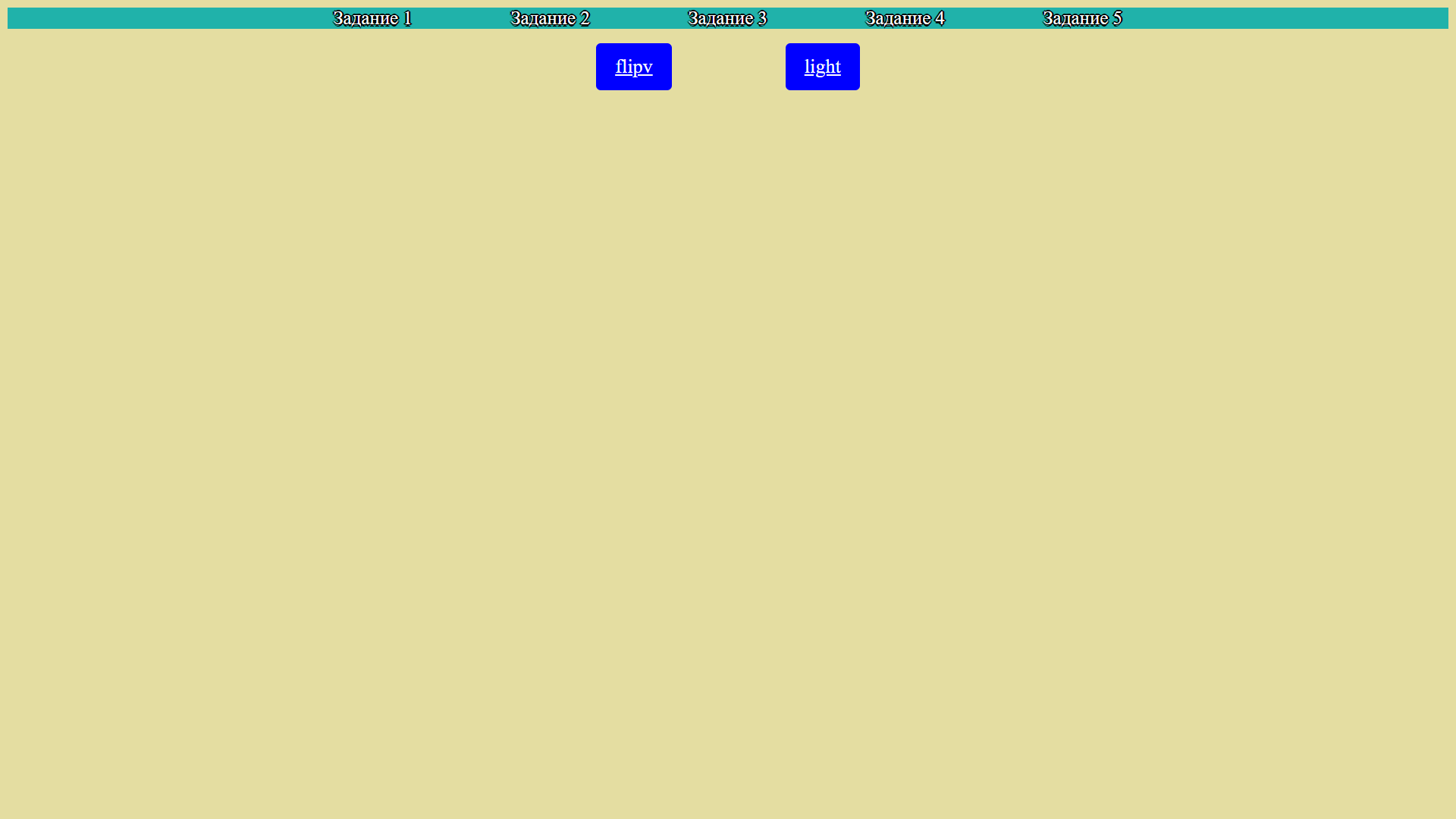


Рисунок 2 – Страница «Задание 2»

На рисунке 3 представлен пример применения статического графического фильтра *flipv*, который в данном случае отражает изображение.

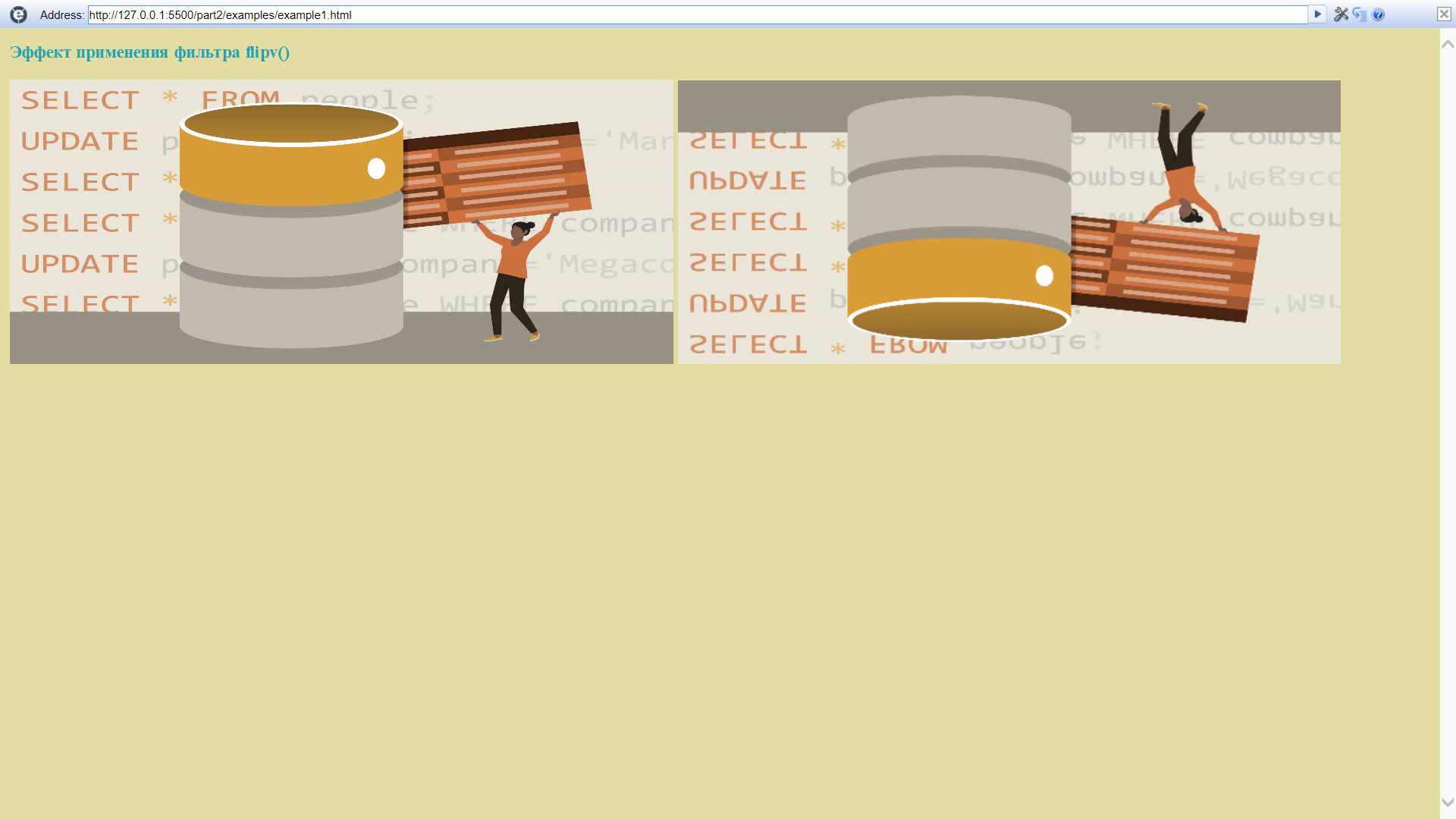


Рисунок 3 – Применение графического фильтра *flipv*

Страница, изображённая на рисунке 4, содержит изображения, к которым применены фильтры *light*, который осветляет изображение, и *invert*, инвертирующий цвета картинки. Данные фильтры являются статическими.

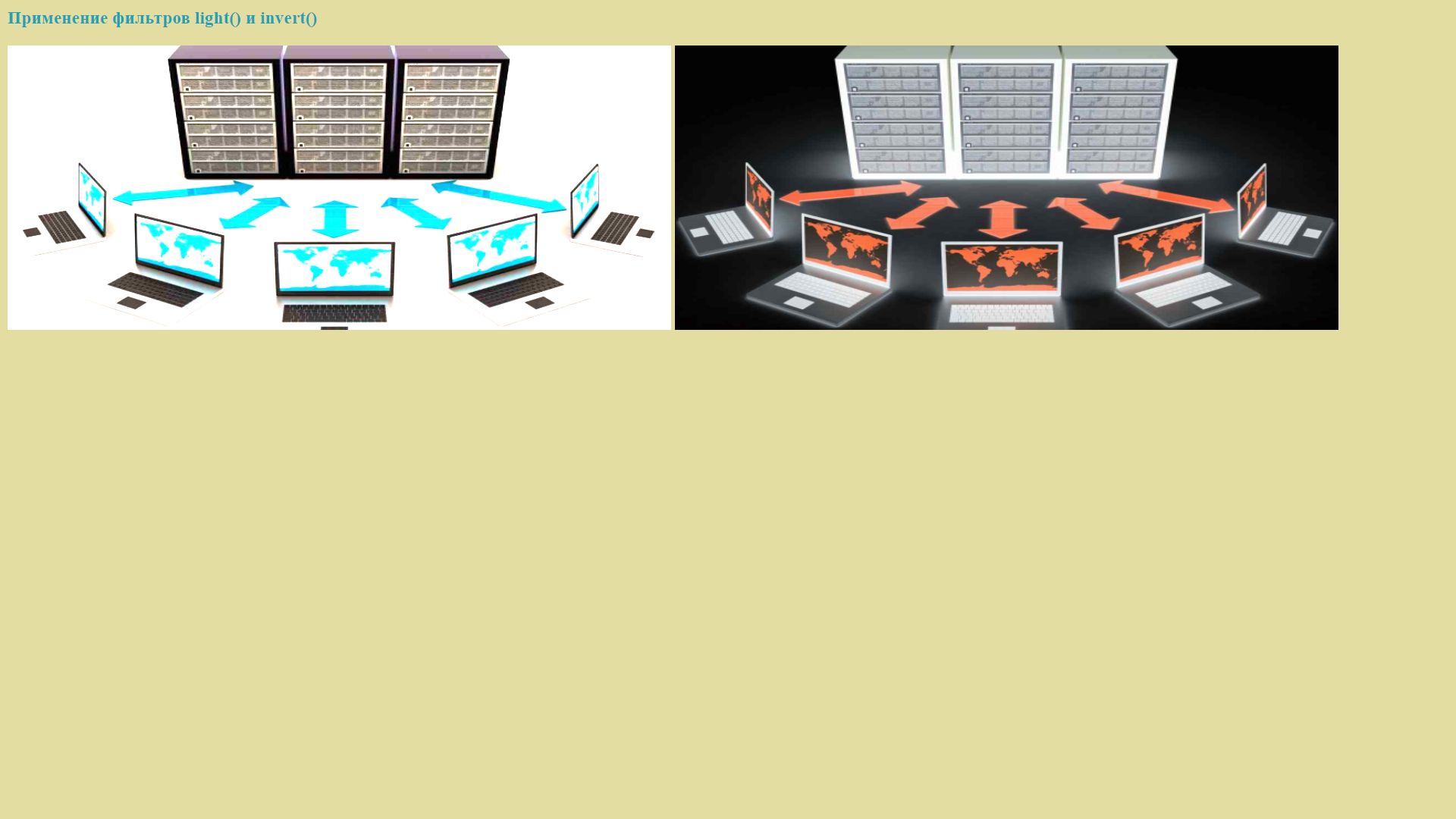


Рисунок 4 – Применение графических фильтров light и invert

### Страница «Задание 3», представленная на рисунке 5, содержит поле для ввода типа перехода *revealtrans*, поле для ввода длительности перехода и кнопку запуска перехода.

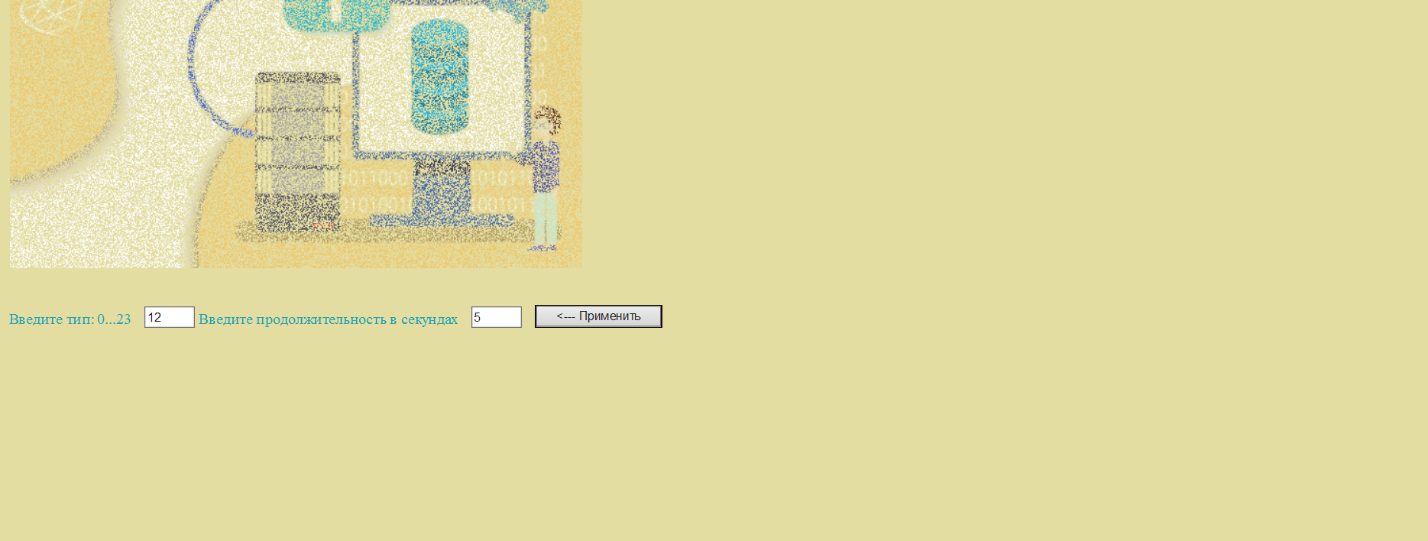


Рисунок 5 – Страница «Задание 3»

### На странице «Задание 4», приведённой на рисунке 6, изображено текущее время, которое изменяется и отображается пользователю.

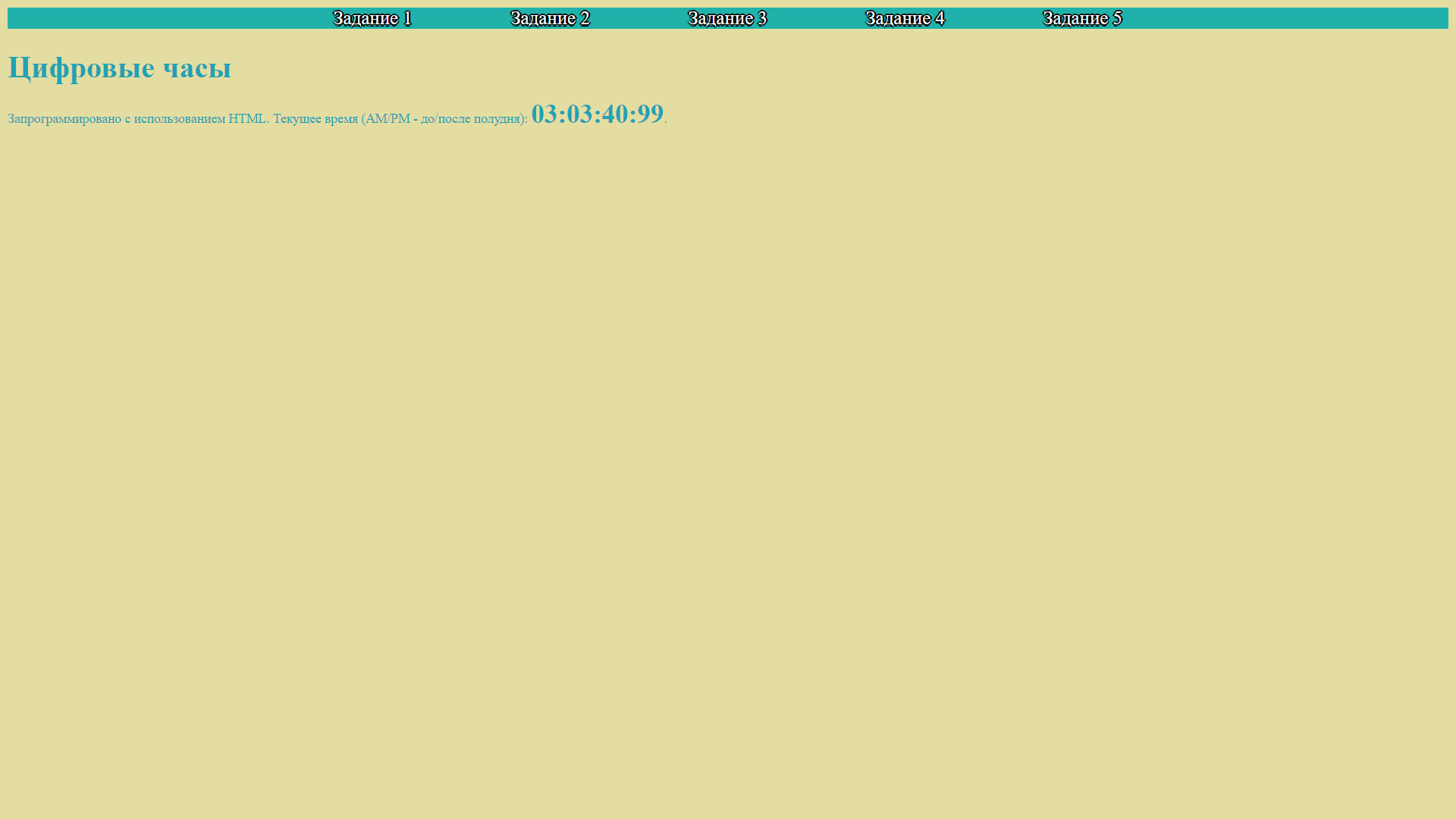


Рисунок 6 – Страница «Задание 4»

### Страница «Задание 5» изображена на рисунке 7. На ней можно в левой части увидеть таблицу, отображающую данные о товарах из текстового файла, а в правом – фрейм, в который загружается дополнительная информация о товаре, дополнительная информация содержится в файле, путь к которому указан в основном файле с данными.



Рисунок 7 – Страница «Задание 5»

## Руководство пользователю

Страница «Задание 1» содержит набор из фреймов, чтобы просмотреть содержимое файла следует навести курсор на одну из полос и нажать на неё. Откроется соответствующий фрейм с контентом.

Чтобы просмотреть демонстрации статических графических фильтров следует поочерёдно нажимать кнопки, расположенные на странице «Задание 2».

На странице «Задание 3» находятся поля для ввода типа и продолжительности перехода *revealtrans* и кнопка «Применить». Необходимо нажать на кнопку, чтобы увидеть пример работы данных переходов.

На странице «Задание 4» просто отображается текущее время.

Пример использования *Fetch API* приведён на странице «Задание 5». Для просмотра дополнительной информации о товаре следует нажать на соответствующую строку таблицы.

# Заключение

В результате выполнения работы систематизированы знания об объектных моделях браузера и документа, создан сайт, на страницах которого продемонстрирована работа с *DHTML*. Изучен способ отправки *HTTP*-запросов посредством метода *fetch* как современная альтернатива использованию объекта класса *XmlHttpRequest* и его методов.

# 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.