Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Распределенные системы обработки информации

Контрольная работа №2
по теме
«Объекты JavaScript»

Выполнил студент 5-го курса группы 694051 специальности «Электронный маркетинг» — Кузьмич Павел Валерьевич

Проверил старший преподаватель кафедры экономической информатики — Атрощенко Натэлла Александровна

Минск, 2020

Содержание

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Организация работы с анимированными объектами в jQuery

Библиотека jQuery содержит несколько кросс-браузерных методов для анимации элементов, например, скольжение и плавное исчезновение, без привлечения дополнительных библиотек или плагинов. CSS-стили придают элементам страницы визуальные свойства, которые описывают их внешний вид. jQuery анимация представляет собой интерактивный процесс изменения свойств HTML-элементов от одного значения к другому.

Эффекты можно создавать с помощью метода *animate()*. Интерпретатор браузера динамически, без перезагрузки страницы, изменяет выбранные свойства на указанные значения. Анимация происходит для всех элементов обернутого набора. Чтобы добавить эффекты для конкретного элемента, нужно воспользоваться фильтрами jQuery для отбора.

Метод позволяет анимировать любое css-свойство, имеющее числовое значение, например, font-size, opacity, border-width, margin, padding, height, width, background-position и т.д. При этом имена свойств должны быть указаны слитно — fontSize, paddingLeft, или должен использоваться css-эквивалент свойства — "font-size". Числовые значения свойств не заключаются в кавычки.

Для любого свойства предварительно должно быть установлено начальное значение, а в css-объявлении должна использоваться полная запись каждого свойства, т.е., вместо свойства border должно быть заданы значения для border-style, border-width и т.д.

Функция обратного вызова вызывается один раз после завершения анимации. Функции не передается никаких аргументов, но анимации выполняется для элемента, переданного свойству this в качестве контекста. Значениями свойств могут также выступать hide, show или toggle, в результате чего к элементу применится вычисляемой значение — отображение, скрытие или переключение исходных состояний свойств.

Метод animate()позволяет изменять css-свойства выбранных элементов с возможностью одновременной анимации нескольких свойств, задавая продолжительность анимации в миллисекундах: *$("div").animate({left: "200px", top: "200px"}, 500)*. Данная анимация одновременно применяется к свойству left, для которого задано значение 200px, и к свойству top со значением 200px, продолжительность анимации задана 500ms.

Анимация элемента реализуется только с помощью методов, создающих анимационные эффекты. Если анимация реализована цепочкой методов, каждый эффект выполняется последовательно, друг за другом. Например, *$("div").slideDown().fadeOut()*; сначала начнется скольжение блока вниз, а выцветание поставится в очередь "fx" и будет вызвано только при завершении скольжения.

Чтобы добавить пользовательские эффекты, создаются функции, которые также добавляются в очередь "fx". Очередь представляет массив функций, существующих на уровне элемента и хранящихся в jQuery.data. Каждый элемент может иметь одну или несколько очередей функции, но обычно используется только одна очередь по умолчанию "fx".

Функции включаются в очередь с помощью метода *queue()*, без него функции выполняться не будут. Каждая функция по своему завершению должна вызвать метод *dequeue()*, чтобы передать управление следующей функции в очереди.

Свойство jQuery.fx.interval gозволяет изменить скорость выполнения анимации. В качестве значения указывается количество миллисекунд. Значение по умолчанию 13ms. Уменьшая скорость, можно добиться более плавного выполнения эффектов: j*Query.fx.interval = 500*.

Свойство jQuery.fx.off используется для того, чтобы глобально отключить или включить анимацию. Значение по умолчанию false, что соответствует выполнению анимации. Если задано значение true, то все методы анимации будут отключены, а элементы вернутся к своему первоначальному состоянию.

1.2. Управление зависимостями: AMD (Asynchronous Module Definition) концепция в JavaScript

Асинхронное определение модуля — это подход к разработке на Javascript, при котором модули и их зависимости могут быть загружены асинхронно. Асинхронная загрузка модулей пет улучшить скорость загрузки веб-страницы в целом, так как модули загружаются одновременно с остальным контентом сайта.

Кроме того, AMD может быть использован во время разработки для разбиения JavaScript-кода по разным файлам. Похожие механизмы имеются и в других языках программирования, например Java, где для определения модулей используются ключевые слова import, package и Class. Для промышленной эксплуатации JavaScript-файлы рекомендуется объединить и сжать в один маленький файл.

AMD помогает решит сразу несколько проблем: описание и удовлетворение зависимостей различных частей приложения, необходимость организации подключения зависимостей на серверной стороне; экспорт переменных в глобальную область видимости и их коллизия. Она помогает избежать хаоса при использовании большого количества JS-файлов в разработке, подталкивает к переиспользованию кода, снимает ответственность за подключение файлов с бэкенда.

Загрузка модулей с помощью AMD выглядит примерно так: define(['myModule', 'myOtherModule'], function(myModule, myOtherModule) {console.log(myModule.hello());});.

Функция определения модуля принимает первым аргументом массив зависимостей. Эти зависимости загружаются в фоновом (не блокирующим) режиме и вызывают функцию обратного вызова, которая была передана вторым аргументом. Далее, функция обратного вызова в качестве аргументов принимает уже загруженные зависимости, и позволяет использовать их. И наконец сами зависимости тоже должны быть определены с помощью ключевого слова *define*.

Кроме асинхронности, у AMD есть ещё одно преимущество. AMD модули могут быть функциями, конструкторами, строками, JSON'ом и другими типами, в то время как CommonJS в качестве модулей поддерживает только объекты. Из минусов, AMD не совместим с io, файловой системой и другими серверно-ориентированными особенностями.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1. Описание HTML кода страницы

Основное особенностью HTML-кода данного проекта является пустой тег <body>, так как контент на странице формируется за счёт возможностей библиотеки jQuery, которая подгружается из Google CDN (Content Delivery Network). Сам же скрипт записан в файле script.js. Листинг приведён ниже.

Листинг 1.1 Структура HTML страницы.

```
<html>
<head>
	<meta charset="utf-8">
	<link rel="stylesheet" href="content.css">
	<title>jQuery</title>
</head>
<body>
</body>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js">
</script>
<script src="script.js"></script>
</html>
```

2.2. Описание CSS кода каскадных таблиц стилей

Основные аттрибуты стиля для фона, таблицы (отдельно заголовков и самих строк данных) прописаны в файле content.css. Для body задан градиентный фон. Для таблицы заданы анимации события типа hover, т.е. изменение цвета ячейки при наведении. Также заданы параметры размеров

всех изображений, в данном случае 50%. Для размещения таблицы создан класс container. Аттрибуты оформления функциональных кнопок прописаны в классе button. Листинг CSS кода приведён ниже.

Листинг 2.1

```
html,
body {
	height: 100%;
}
body {
	margin: 0;
	background: linear-gradient(45deg, #49a09d, #5f2c82);
	font-family: sans-serif;
	font-weight: 100;
}
table {
	width: 1280px;
	border-collapse: collapse;
	overflow: hidden;
	box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
}
th,
td {
	padding: 15px;
	background-color: rgba(255, 255, 255, 0.2);
	color: #fff;
}
th {
	text-align: left;
}
thead th {
	background-color: #55608f;
}
tbody tr:hover {
	background-color: rgba(255, 255, 255, 0.3);
}
tbody td {
	position: relative;
}
tbody td:hover:before {
	content: "";
	position: absolute;
	left: 0;
	right: 0;
```

Листинг 2.1 Продолжение

```
        top: -9999px;
        bottom: -9999px;
        background-color: rgba(255, 255, 255, 0.2);
        z-index: -1;
}
img {
        width: 50%;
        height: 50%;
}
.container {
        position: absolute;
        top: 50%;
        left: 50%;
        -webkit-transform: translate(-50%, -50%);
                transform: translate(-50%, -50%);
}
.button {
        text-decoration: none;
        align-items: center;
        margin-top: 30px;
        margin-left: 30px;
        color: black;
        font-size: 18px;
        background: lightsteelblue;
        padding: 10px 20px;
        border-radius: 8px;
        font-weight: normal;
        text-shadow: 0.5px 1px white;
        transition: all 0.2s ease-in-out;
}
```

2.3. Описание функциональной части

Динамическое заполнение и изменения страницы реализовано с помощью средств JavaScript и библиотеки jQuery. Для добавления информации создан класс Content содержащий два статических метода creteButton() и createTable() для создания элементов кнопок и таблицы соответственно. Для создания кнопок применяется тег <input type=«button»>, таблицы — теги <thead>, <tbody>, <td>, <th>, <td>, которые в цикле заполняются данными из констант, определённых также в этом js файле. Код описан в листинге 3.1.

Листинг 3.1

```
const tableHeadersArray = ["City name", "Picture"];
const tableContentMap = {
    "Amsterdam": [
"Amsterdam is the Netherlands’ capital, known for its artistic heritage, elaborate canal system and narrow houses with gabled facades, legacies of the city’s 17th-century Golden Age. Its Museum District houses the Van Gogh Museum, works by Rembrandt and Vermeer at the Rijksmuseum, and modern art at the Stedelijk. Cycling is key to the city’s character, and there are numerous bike paths.",
    "amsterdam.jpg"
    ],
"       Malaga": [
"Malaga is a port city on southern Spain’s Costa del Sol, known for its high-rise hotels and resorts jutting up from yellow-sand beaches. Looming over that modern skyline are the city’s 2 massive hilltop citadels, the Alcazaba and ruined Gibralfaro, remnants of Moorish rule. The city's soaring Renaissance cathedral is nicknamed La Manquita ('one-armed lady') because one of its towers was curiously left unbuilt.",
    "malaga.jpg"
    ],
        "London": [
"London, the capital of England and the United Kingdom, is a 21st-century city with history stretching back to Roman times. At its centre stand the imposing Houses of Parliament, the iconic ‘Big Ben’ clock tower and Westminster Abbey, site of British monarch coronations. Across the Thames River, the London Eye observation wheel provides panoramic views of the South Bank cultural complex, and the entire city.",
    "london.jpg"
    ],
    }
    // класс с методами заполнения страницы данными
    class Content {
        static createButton(buttonName, buttonId, styleClass) {
        return `<div><input type="button" id="${buttonId}" class="$
        {styleClass}" value="${buttonName}"></div>`;
    }
    static createTable(tableHeadersArray, tableContentMap) {
        var headers = "<thead><tr>";
        var rows = "<tbody>";
        // создать заголовки таблицы
        for (var header of tableHeadersArray) {
            headers += `<th>${header}</th>`;
        }; headers += "</tr></thead>";
```

Листинг 3.1 Продолжение

```
        // создать наполнение таблицы
        for (var key in tableContentMap) {
                var data_header = `<h2>${key}</h2>`;
                var data = tableContentMap[key];
                let [text, image] = data;
                image = `<img id="${key}-img" src="${image}">`
                rows += `<tr><td>${data_header}<p>${text}</p></td><td>$
                                {image}</td></tr>`;
        }
        rows += "</tbody>";
        return `<div class="container"><table>${headers +
                                                        rows}</table></div>`;
        }
}
```

Непосредственное заполнение данными осуществляется уже с помощью jQuery, когда с помощью записи *$(«body»).append()* к телу страницы добавляется новый элемент. Функциональная логика кнопок тоже реализована на основе jQuery: получение кнопок осуществляется с помощью метода *$(«id_кнопки»)*, а привязка функции — методом *.click()*. Фильтрация и поиск текстовых ячеек происходит с помощью метода *$("td").filter(function () {return $(this).text();});*. Обновление изображений в таблице происходит с использованием эффектов fadeOut(), fadeIn(). Полное описание кода приведено в листинге 3.2.

Листинг 3.2

```
var $updateTextButton = $(Content.createButton("Update text", "update-
                                                        text", "button"));
var $updatePicturesButton = $(Content.createButton("Update pictures",
                                                "update-pictures", "button"));
var $dataTable = $(Content.createTable(tableHeadersArray,
                                                        tableContentMap));
$(document).ready(function () {
        // создать элементы страницы
        $("body").append($updateTextButton, $updatePicturesButton);
        $("body").append($dataTable);
        // сделать текст курсивом
        $("#update-text").click(function () {
                textCells = $("td").filter(function () {
                        return $(this).text();
                });
                for (var cell of textCells) {
                        cell.style.fontStyle = "italic";
```

```
                if (cell.textContent.includes("London")) {
                    cell.style.textDecoration = "underline";
                }
            };
        });;
        // изменить изображения
        $("#update-pictures").click(function () {
            $("img").fadeOut('fast');
            $("img").fadeIn('slow');
            $("img").attr("src", "stub.jpg");
        });;
});
```

Скриншоты работы проекта приведены ниже на рисунках.

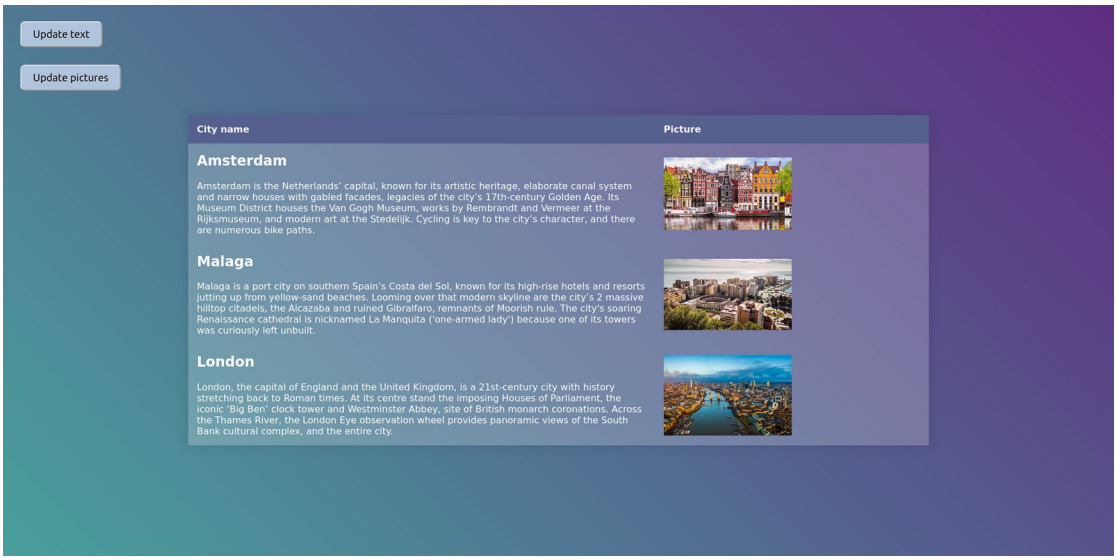

Рисунок 1 — Страница проекта

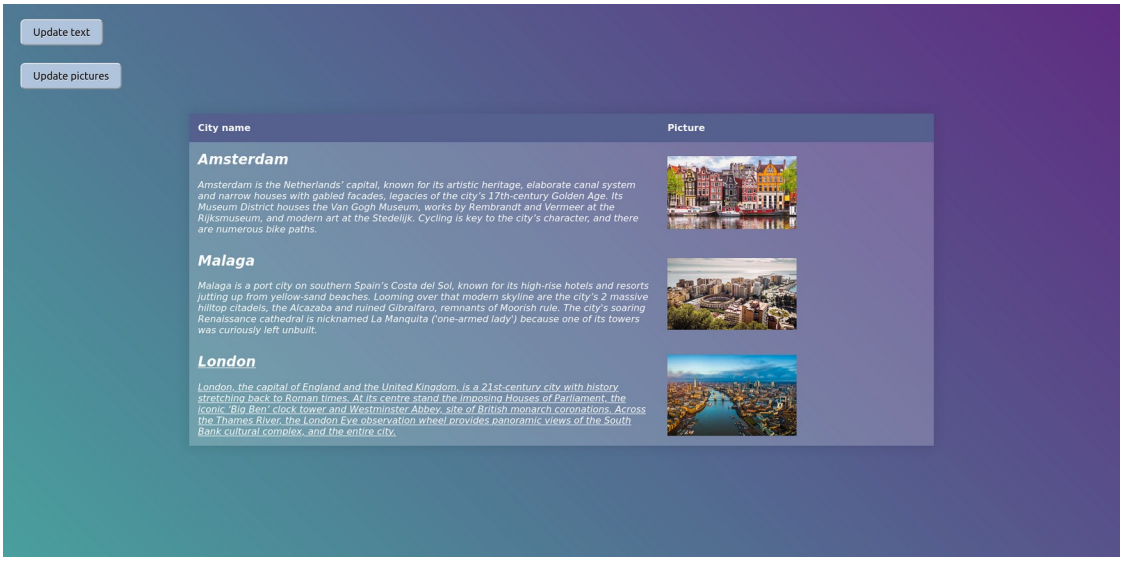

Рисунок 2 — Обновление стиля текста

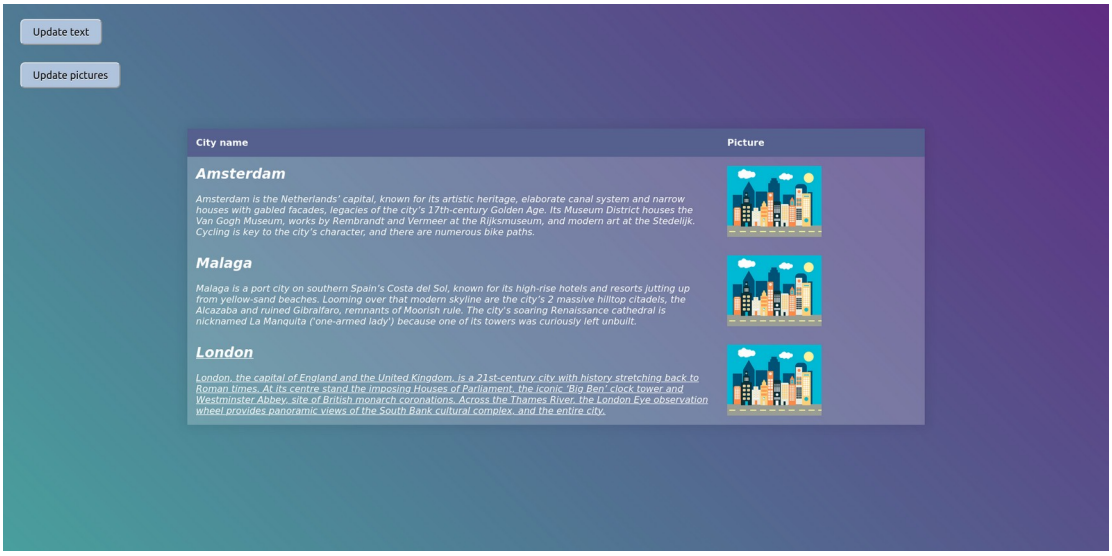

Рисунок 3 — Обновление изображений

ВЫВОДЫ

В результате выполнения контрольной работы была создана веб-страница, динамически заполняемая с использованием средств библиотеки jQuery и возможностей языка JavaScript. Было предусмотрено стилистическое оформление страницы на основе каскадных таблиц стилей.