**МИНОБРНАУКИ РОССИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«1» 02 2018 г.

# ИЗУЧЕНИЕ ОСНОВ HTML, CSS, JAVASCRIPT

Методические указания по выполнению лабораторной работы по дисциплине «Веб-программирование» для студентов специальности 10.05.02 и 10.03.01

Курск 2018

УДК 004.056.55

Составитель А.Л. Марухленко

Рецензент
Кандидат технических наук, доцент *И.В. Калуцкий*

**Изучение основ HTML, CSS, JavaScript:** методические указания по выполнению лабораторной работы по дисциплине «Веб-программирование» / Юго-Зап. гос. ун-т; сост. А.Л. Марухленко. Курск, 2018. - 30с.

Рассматриваются основы HTML, CSS, JavaScript. Указывается порядок выполнения лабораторной работы и содержание отчета.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по образованию в области информационной безопасности (УМО ИБ).

Предназначены для студентов специальности 10.05.02 и 10.03.01.

Текст печатается в авторской редакции

Подписано в печать 01.02.18 . Формат 60x84 1/16.
Усл.печ. л. 1,74 . Уч.-изд.л. 1,57. Тираж 30 экз. Заказ. Бесплатно. 243
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

**СОДЕРЖАНИЕ**

## ЦЕЛЬ РАБОТЫ

Целью выполнения лабораторной работы является формирование у студентов навыков и умений по разработке простых веб-приложений.

## ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

### Язык гипертекстовой разметки

HTML-документ – это обычный текстовый документ, может быть создан как в обычном текстовом редакторе (Блокнот), так и в специализированном, с подсветкой кода (Notepad++, SublimeText, PhpStorm). HTML-документ имеет расширение *.html.

Язык HTML – язык тегов. Теги описывают структуру HTML-документа. Теги оформляются угловыми скобками <имя тега>, между которыми прописывается имя тега.

Теги располагаются в HTML-документе в соответствии с правилами разметки (порядок следования, правило вложенности тегов), создавая разделы будущей веб-страницы. Кроме тегов, HTML-документы могут содержать специальные символы.

Браузер просматривает (интерпретирует) HTML-документ, выстраивая его структуру (DOM) и отображая ее в соответствии с инструкциями, включенными в этот файл (таблицы стилей, скрипты). Если разметка правильная, то в окне браузера будет отображена HTML-страница, содержащая HTML-элементы – заголовки, таблицы, изображения и т.д.

Процесс интерпретации (парсинг) начинается прежде, чем веб-страница полностью загружена в браузер. Браузеры обрабатывают HTML-документы последовательно, с самого начала, при этом обрабатывая CSS и соотнося таблицы стилей с элементами страницы.

Практически все теги имеют атрибуты (глобальные, применяемые для всех html-элементов, и собственные), указываемые в формате параметр="значение". Атрибуты позволяют изменять

свойства элемента, для которого они заданы. Атрибуты прописываются в начальном теге элемента.

Атрибуты class и id применимы ко всем HTML-элементам, за исключением элементов, содержащих техническую информацию – <html>, <head>, <meta>, <title>, <style> и <script>. Каждому конкретному элементу можно присвоить несколько значений class и только одно значение id.

Множественные значения class записываются через пробел, . Значения class и id должны состоять только из букв, цифр, дефисов и нижних подчеркиваний и должны начинаться только с букв или цифр.

Большинство тегов – парные, они состоят из начального и закрывающего тегов. Начальный тег показывает, где начинается элемент, закрывающийся – где заканчивается. Закрывающий тег образуется путем добавления слэша/ перед именем тега: <имя тега>…</имя тега>. Между начальным и закрывающим тегами находится содержимое тега – контент.

Одиночные теги не могут хранить в себе содержимого напрямую, оно прописывается как значение атрибута, например, тег <input type="button" value="Кнопка"> создаст кнопку с текстом Кнопка внутри.

Теги могут вкладываться друг в друга, например, Текст . При вложении следует соблюдать порядок их закрытия (принцип «матрёшки»), например, следующая запись будет неверной: Текст .

HTML-документ состоит из двух разделов – заголовка (между тегами <head>…</head>) и содержательной части (между тегами <body>…</body>).

Элементы, находящиеся внутри тега <html>, образуют дерево документа, так называемую объектную модель документа, DOM (document object model). При этом элемент <html> является корневым элементом (рисунок 1).

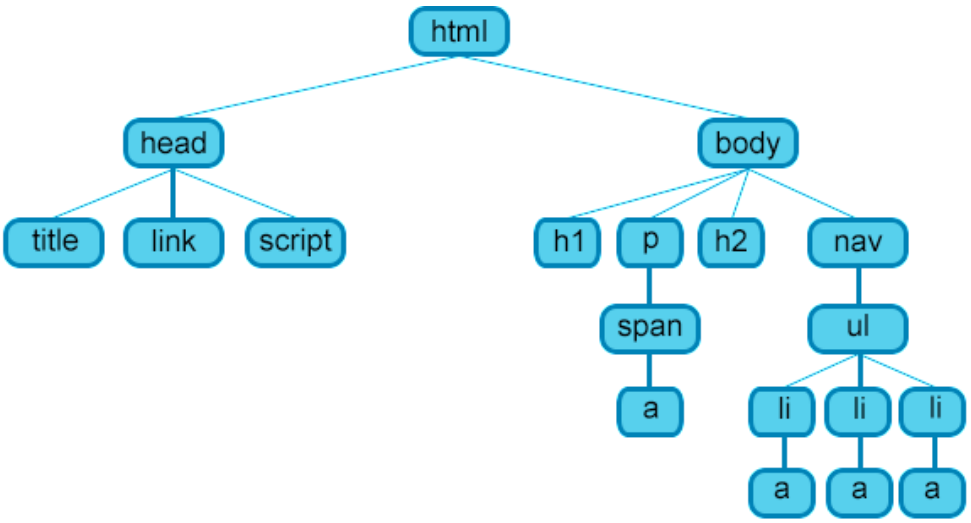


Рисунок 1 – Структура веб-страницы.

Чтобы разобраться во взаимодействии элементов веб-страницы, необходимо рассмотреть так называемые «родственные отношения» между элементами. Отношения между множественными вложенными элементами подразделяются на родительские, дочерние и сестринские.

Предок – элемент, который заключает в себе другие элементы. На рисунке 1 предком для всех элементов является <html>. В то же время элемент <body> является предком для всех содержащихся в нем тегов: <h1>, <p>, , <nav> и т.д.

Потомок – элемент, расположенный внутри одного или более типов элементов. Например, <body> является потомком <html>, а элемент <p> является потомком одновременно для <body> и <html>.

Родительский элемент – элемент, связанный с другими элементами более низкого уровня, и находящийся на дереве выше их. На рисунке 1 <html> является родительским только для <head> и <body>. Тег <p> является родительским только для .

Дочерний элемент – элемент, непосредственно подчиненный другому элементу более высокого уровня. На рисунке 1 только элементы <h1>, <h2>, <p> и <nav> являются дочерними по отношению к <body>.

Сестринский элемент – элемент, имеющий общий родительский элемент с рассматриваемым, так называемые элементы одного уровня. На рисунке 1 <head> и <body> – элементы одного уровня, так же как и элементы <h1>, <h2> и <p> являются между собой сестринскими.

HTML теги являются основой языка HTML. Каждый тег несет в себе определенную информацию, может описывать документ в общем или способ форматирования текста. Все содержимое веб-страницы задается с помощью тегов.

Тег помещается в угловые скобки <тег>. Чаще всего для тега задается парный закрывающий тег, но в некоторых случаях он отсутствует. Информация, заключенная между открывающим и закрывающим тегом, называется его контейнером.

Список всех тегов, их атрибутов и варианты их применения можно посмотреть на различных интернет ресурсах, например, http://htmlbook.ru/.

HTML атрибуты сообщают браузеру, каким образом должен отображаться тот или иной элемент страницы. Атрибуты позволяют сделать более разнообразными внешний вид информации, добавляемой с помощью одинаковых тегов. Значение атрибута всегда заключается в двойные кавычки "". Названия и значения атрибутов не чувствительны к регистру, но, тем не менее, рекомендуется набирать их в нижнем регистре.

HTML таблицы состоят из ячеек, образующихся при пересечении строк и столбцов. Ячейки таблиц могут содержать любые HTML-элементы, такие как заголовки, списки, текст, изображения, элементы форм, а также другие таблицы. Таблицы в HTML-документах используются не только для группировки связанной информации, но и для точного позиционирования фрагментов текста и изображений относительно друг друга. С помощью таблиц можно выровнять фрагменты страниц относительно друг друга, разместить рядом изображение и текст, управлять цветовым оформлением, разбить текст на столбцы.

Создание сайта всегда начинается с создания файла index.html. Index – это общепринятое название главной страницы сайта. Общий вид HTML-документа представлен ниже.

```
<!DOCTYPE html>
<html lang="ru">
<head>
   <meta charset="utf-8">
   <title>Шаблон-пример</title>
   <script src="*.js"></script>
   <link href="*.css" rel="stylesheet">
```

```
    </head>
<body>
    <p>Привет, мир!</p>
</body>
</html>
```

<!DOCTYPE> – предназначен для указания типа текущего документа. Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, поскольку HTML существует в нескольких версиях. Чаще всего используется значение html как в нашем примере.

Теги <html>, <head>, <body> были рассмотрены ранее. Атрибут lang="ru" относящийся к тегу <html> используется для правильного отображения некоторых национальных символов

Рассмотрим по подробнее наполнение <head>:

– <meta charset="utf-8"> используется для определения кодировки документа;

– <title>Шаблон-пример</title> – это заголовок страницы, который будет отображаться в браузере;

– <script src="*.js"></script> используется для подключение внешних скриптов. Вместо * указывается путь и имя подключаемого файла;

– <link href="*.css" rel="stylesheet"> используется для подключение внешних каскадных таблиц стилей. Вместо * указывается путь и имя подключаемого файла. Атрибут rel определяет отношения между текущим документом и файлом, на который делается ссылка. Это необходимо, чтобы браузер знал, как использовать подключаемый документ. Значение stylesheet определяет, что подключаемый файл хранит таблицу стилей (CSS).

**Каскадные таблицы стилей**

CSS (Cascading Style Sheets), или каскадные таблицы стилей, описывают правила форматирования отдельного элемента веб-страницы. Создав стиль один раз, его можно применять к любым элементам страницы сколько угодно раз.

Определение стиля состоит из двух основных частей: самого элемента веб-страницы – селектора, и команды форматирования – блока объявления.

Селектор сообщает браузеру, какой именно элемент форматировать, в блоке объявления перечисляются форматирующие команды.

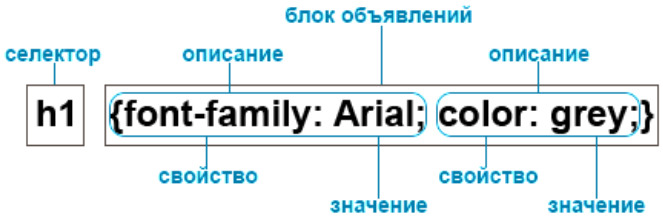


Рисунок 2 – Структура объявления стиля элемента в CSS.

Встраиваемые стили представляют собой набор стилей, являющихся частью кода веб-страницы, заключенных между тегами <style>...</style> и расположенных внутри элемента <head>. Встраиваемые стили действуют только на странице, на которой они содержатся. Встраиваемые стили имеют приоритет перед глобальными, но уступают внутритекстовым стилям. На одной странице можно размещать произвольное количество встраиваемых стилей.

Внутритекстовые стили не пользуются селекторами, их применение заключается в присваивании стиля непосредственно к html-элементу через атрибут style:

<p style="font-family: "Times New Roman", Georgia, Serif; color: #70d7700;">
Обратите внимание на этот текст.</p>

Недостаток этого способа заключается в отсутствии возможности автоматического использования данного стиля для другого элемента.

Внешняя таблица стилей представляет собой текстовый файл, в котором находится весь набор стилей CSS. Он не содержит HTML-код, поэтому его не нужно заключать внутрь тегов <style>...</style>. Название этого файла всегда должно заканчиваться расширением *.css. Заданные таким образом стили будут работать для всех страниц веб-сайта.

К веб-странице можно присоединить несколько таблиц стилей, добавляя последовательно несколько тегов <link>.

Стилизовать элементы html кода, используя внешние каскадные таблицы стилей, можно несколькими способами.

```
#primer1{
   background-color: #ffcacc;
}
.primer2{
   background-color: #9bff99;
}
p{
   font-size: 20px;
}
```

Первый способ – это использование селектора id. Селектор id (идентификатор) предназначен для применения стиля к уникальным элементам на веб-странице, уникальность подразумевает то, что данный элемент используется на странице один раз.

Для использования селектора id, нужно создать идентификатор (id), придумав ему уникальное название, и прописать его в атрибуте id элемента, к которому будет применяться стиль. В описании стиля селектор id начинается с символа # сразу после которого идет название идентификатора.

Каждый идентификатор стиля может встречаться на странице только один раз, т.е. определенный id может быть использован на странице только с тегом, для которого он предназначен, если один и тот же идентификатор будет применен более, чем к одному элементу, во-первых html-код не пройдет валидацию, во-вторых это может вызвать некорректную обработку кода браузером и вы увидите не тот результат, которого ожидали.

Второй способ – это использование селектора class. Селектор класс позволяет так же как и селектор id стилизовать конкретный элемент страницы, но в отличие от id, селектор class позволяет применить свой стиль к нескольким элементам на веб-странице, а не только к одному.

Для использования селектора class, нужно указать, к какому элементу на странице вы хотите его применить, для этого надо всего лишь добавить атрибут class к HTML-тегу, который нужно стилизовать, и указать в качестве значения нужное имя класса с описанным стилем.

Правила для имен классов:

– все названия селекторов классов должны начинаться с точки, с ее помощью браузеры находят селектор класса в таблице стилей CSS;

– в имени класса разрешается использовать только буквы, числа, дефис и знак подчеркивания;

– имя класса после точки всегда должно начинаться с буквы;

– имена классов чувствительны к регистру, например .Menu и .menu будут рассматриваться в CSS, как два разных класса.

Третий способ – это применение стилей для конкретного тега. В этом случае в файле со стилями пишется название тега без <>. При использовании этого способа во всём документе к указанному тегу будет применяться заданный стиль.

К одному и тому же элементу можно применять сразу несколько способов стилизации. Пример кода представлен ниже.

```
<!DOCTYPE html>
<html lang="ru">
<head>
   <meta charset="utf-8">
   <title>Шаблон-пример</title>
   <link href="css/style.css" rel="stylesheet">
   </head>
<body>
   <p class="primer2">Привет, мир!</p>
</body>
</html>
```

В данном примере подключены внешние таблицы стилей, код которых описан выше. То есть к тегу <p> применены второй и третий способ стилизации. В браузере будет отображаться текст с размером шрифта в 20 px и с закрашенным фоном.

Так же можно к одному элементу присваивать несколько различных классов. Главная особенность заключается в том, что если в стилях различных классов встречаются одинаковые свойства, то к html-элементу будет применено то свойство, которое в таблице стилей находится ниже.

Все свойства и способы их применения для стилизации можно посмотреть на различных интернет ресурсах, например, http://htmlbook.ru/.

**Прототипно-ориентированный сценарный язык программирования**

JavaScript – язык сценариев, с его помощью можно создавать интерактивные html-документы, производить вычисления, выполнять проверку допустимости данных без обращения к серверу.

Сценарий JavaScript не компилируется в бинарный код наподобие .exe, а интерпретируется и обрабатывается интерпретатором, встроенным в браузер. Браузер начинает выполнять код JavaScript сразу же, как только обнаружит его на странице. Каждая строка кода выполняется последовательно, переход к следующей строке производится только после выполнения предыдущей. Если в документе содержится несколько сценариев, то они будут исполняться в порядке их расположения в документе.

Сценарии JavaScript бывают встроенные, т.е. их содержимое является частью документа, и внешние, хранящиеся в отдельном файле с расширением *.js. Сценарии можно внедрить в html-документ следующими способами:

– в виде гиперссылки. Для этого нужно разместить код в отдельном файле и включить ссылку на файл в заголовок. Этот способ обычно применяется для сценариев большого размера или сценариев, многократно используемых на разных веб-страницах;

– в виде обработчика события. Каждый html-элемент имеет JavaScript-события, которые срабатывают в определенный момент. Нужно добавить необходимое событие в html-элемент как атрибут, а в качестве значения этого атрибута указать требуемую функцию. Функция, вызываемая в ответ на срабатывание события, является обработчиком события. В результате срабатывания события исполнится связанный с ним код. Этот способ применяется в основном для коротких сценариев, например, можно установить смену цвета фона при нажатии на кнопку;

– внутрь элемента <script>. Элемент <script> может вставляться в любое место документа. Внутри тега располагается код, который выполняется сразу после прочтения браузером, или содержит описание функции, которая выполняется в момент ее вызо-

ва. Описание функции можно располагать в любом месте, главное, чтобы к моменту ее вызова код функции уже был загружен.

Обычно код JavaScript размещается в заголовке документа (элемент <head>) или после открывающего тега <body>. Если скрипт используется после загрузки страницы, например, код счетчика, то его лучше разместить в конце документа[1].

**Типы данных и переменные**

Компьютеры обрабатывают информацию – данные. Данные могут быть представлены в различных формах или типах. Большая часть функциональности JavaScript реализуется за счет простого набора объектов и типов данных. Функциональные возможности, связанные со строками, числами и логикой, базируются на строковых, числовых и логических типах данных. Другая функциональная возможность, включающая регулярные выражения, даты и математические операции, осуществляется с помощью объектов RegExp, Date и Math.

Литералы в JavaScript представляют собой особый класс типа данных, фиксированные значения одного из трех типов данных – строкового, числового или логического.

**Переменные**

Данные, обрабатываемые сценарием JavaScript, являются переменными. Переменные представляют собой именованные контейнеры, хранящие данные (значения) в памяти компьютера, которые могут изменяться в процессе выполнения программы. Переменные имеют имя, тип и значение.

Имя переменной, или идентификатор, может включать только буквы a-z, A-Z, цифры 0-9 (цифра не может быть первой в имени переменной), символ $ (может быть только первым символом в имени переменной или функции) и символ подчеркивания _, наличие пробелов не допускается. Длина имени переменной не ограничена. Можно, но не рекомендуется записывать имена переменных буквами русского алфавита, для этого они должны быть записаны в Unicode.

В качестве имени переменной нельзя использовать ключевые слова JavaScript. Имена переменных в JavaScript чувствительные к регистру, что означает, что переменная varmessage; и varMessage: – разные переменные.

Переменная создается (объявляется) с помощью ключевого слова var, за которым следует имя переменной, например, varmessage;. Объявлять переменную необходимо перед ее использованием.

Переменная инициализируется значением с помощью операции присваивания =, например, varmessage="Hellow";, т.е. создается переменная message и в ней сохраняется ее первоначальное значение "Hellow". Переменную можно объявлять без значения, в этом случае ей присваивается значение по умолчанию undefined. Значение переменной может изменяться во время исполнения скрипта. Разные переменные можно объявлять в одной строке, разделив их запятой: varmessage="Hellow", number_msg = 6, time_msg = 50;

**Арифметические операторы**

Арифметические операторы (таблица 1) предназначены для выполнения математических операций, они работают с числовыми операндами (или переменными, хранящими числовые значения), возвращая в качестве результата числовое значение.

Если один из операндов является строкой, интерпретатор JavaScript попытается преобразовать его в числовой тип, а после выполнить соответствующую операцию. Если преобразование типов окажется невозможным, будет получен результат NaN (не число).

Таблица 1 – Операторы в JavaScript.

| Оператор/ операция | Описание | Приоритет |
|---|---|---|
| + Сложение | Складывает числовые операнды. Если один из операндов – строка, то результатом выражения будет строка. | 12 |
| - Вычитание | Выполняет вычитание второго операнда из первого. | 12 |
| - Унарный минус | Преобразует положительное число в отрицательное, и наоборот. | 14 |
| * Умножение | Умножает два операнда. | 13 |

| | | |
|---|---|---|
| / Деление | Делит первый операнд на второй. Результатом деления может являться как целое, так и число с плавающей точкой. | 13 |
| % Деление по модулю (остаток от деления) | Вычисляет остаток, получаемый при целочисленном делении первого операнда на второй. Применяется как к целым числам, так и числам с плавающей точкой. | 13 |
| == Равенство | Проверяет две величины на совпадение, допуская преобразование типов. Возвращает true, если операнды совпадают, и false, если они различны. | 9 |
| != Неравенство | Возвращает true, если операнды не равны | 9 |
| === Идентичность | Проверяет два операнда на «идентичность», руководствуясь строгим определением совпадения. Возвращает true, если операнды равны без преобразования типов. | 9 |
| !== Неидентичность | Выполняет проверку идентичности. Возвращает true, если операнды не равны без преобразования типов. | 9 |
| > Больше | Возвращает true, если первый операнд больше второго, в противном случае возвращает false. | 10 |
| >= Больше или равно | Возвращает true, если первый операнд не меньше второго, в противном случае возвращает false. | 10 |
| < Меньше | Возвращает true, если первый операнд меньше второго, в противном случае возвращает false. | 10 |

| | | |
|---|---|---|
| <= Меньше или равно | Возвращает true, если первый операнд не больше второго, в противном случае возвращает false. | 10 |

**Библиотека JQuery**

jQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX[2]. Сейчас разработка jQuery ведётся командой jQuery во главе с Джоном Резигом.

Возможности:

– движокккроссбраузерных CSS-селекторов Sizzle, выделившийся в отдельный проект;

– переход по дереву DOM, включая поддержку XPath как плагина;

– события;

– визуальные эффекты;

– AJAX-дополнения;

– JavaScript-плагины.

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки управление передаётся JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript.

Библиотека jQuery содержит функциональность, полезную для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, бóльшая часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно ту JavaScript-функциональность, которая на нём была бы востребована.

28 сентября 2008 года на официальном блоге jQuery сообщили о том, что компании Microsoft и Nokia собираются сотрудничать с группой разработчиков. Компания Microsoft собирается интегрировать в свой продукт ASP.NET листинги кода и примеры jQuery, а компания Nokia собирается интегрировать jQuery для своих мобильных виджетов[2].

Библиотека jQuery производит манипуляции с html-элементами, управляя их поведением и используя DOM для изменения структуры веб-страницы. При этом исходные файлы HTML и CSS не меняются, изменения вносятся лишь в отображение страницы для пользователя.

Для выбора элементов используются селекторы CSS. Выбор осуществляется с помощью функции $(). При вызове функция $() возвращает новый экземпляр объекта JQuery, который оборачивает ноль или более элементов DOM и позволяет взаимодействовать с ними различными способами.

Выполнение различных сценариев возможно только после окончания загрузки структуры документа document, когда браузер преобразует html-код страницы в дерево DOM. Управление процессом загрузки обеспечивает конструкция.

Сначала производится обертывание экземпляра document в функцию jQuery(), после применяется метод ready(), которому передается функция function() {...}, исполняемая после загрузки документа.

На практике обычно используется сокращенная форма такой записи jQuery(function() {...});, или $(function() {...});.

Для хранения информации при работе с библиотекой jQuery используются переменные JavaScript. В переменных могут храниться элементы. Имена переменных, предназначенных для хранения возвращаемых элементов, начинаются со знака $. Для хранения нескольких элементов используются массивы JavaScript:

Библиотека jQuery упрощает процесс отбора элементов HTML-страниц. С помощью методов jQuery производятся манипуляции с объектной моделью документа DOM. Чтобы отобрать группу элементов, нужно передать селектор функции jQuery. В качестве селектора элемента может выступать сам элемент, его идентификатор или класс, а также комбинация селекторов:

Функция $() возвращает объект jQuery, содержащий массив элементов DOM — так называемый обернутый набор, соответ-

ствующих указанному селектору. Большинство методов возвращает по завершении действий первоначальный набор элементов.

Селекторы jQuery выбирают элементы веб-страницы, а методы выполняют операции с этими элементами.

Чтобы выбрать элементы, нужно передать селектор функции $(), например, $("img:odd"). Данный селектор будет применен ко всему дереву DOM, и чтобы ограничить процедуру отбора элементов, можно указать определенный фрагмент дерева DOM – $("img:odd", "div#slideshow"). Таким образом будут выбраны все четные картинки из блока с id=slideshow.

Для более точного выбора элементов селекторы можно комбинировать, например, следующая запись позволит выбрать все флажки полей формы, которые были выделены пользователем – $("input[type=checkbox][checked]").

А с помощью этой комбинации селекторов $("input:checkbox:checked:enabled") можно выбрать только активные и отмеченные флажки полей формы.

Также, допускается объединять несколько селекторов в одно выражение, разделяя селекторы запятой – $("p,span"), что позволит отобрать все элементы <p> и .

Таблица 2 – Селекторы jQuery.

| Селектор | Описание, пример |
|---|---|
| Элемента | Выбирает все элементы данного типа на странице, например, $("div"). |
| Элемента 1 элемента 2 | Выбирает все элементы 2, вложенные непосредственно в элемент 1, например, $("p img"). |
| Класса | Выбирает все элементы заданного класса, например, $(".sidebar"). |
| Идентификатора | Выбирает элемент с указанным идентификатором, например, $("#main"). |
| Элемента класса | Выбирает из элементов данного типа только те элементы, которым назначен указанный класс, например, $("p.first"). |
| Потомка | Выбирает все указанные элементы выбранного селектора, например, $(".sidebar a"). |
| Дочерние | Выбирает элементы, соответствующие второму селектору, которые содержатся непосредственно внутри первого селектора, являющиеся дочерними по отношению к нему, например, $("body> p"). |
| Сестринские | Выбирает элементы, соответствующие второму селектору, идущие непосредственно за первым элементом, являющимся для него сестринским, например, $("h2 + p"). |
| Атрибута | Выбирает элементы, соответствующие второму селектору, являющиеся сестринскими по отношению к первому элементу и расположенные после него, например, $("h2 ~ p"). |
| | Выбирает все элементы, которые содержат данный атрибут или указанно значение атрибута, например, $("img[alt]"), $("a[href]"), $("input[type="text"]"). |
| | Выбирает все элементы, начинающиеся с определенного значения, например, |

| | |
|---|---|
| | $("a[href^="http://"]"). |
| | Выбирает все элементы, заканчивающиеся на определенное значение, например, $("a[href$=".pdf"]"). |
| | Выбирает все элементы, содержащие в любом месте определенное значение, например, $("a[target*="blank"]"). |
| :even | Выбирает элементы по четным значениям индекса 0, 2, 4…, т.е. выбирает 1, 3, 5… элементы, например, $("li:even"). |
| :odd | Выбирает элементы по нечетным значениям индекса, т.е. выбирает 0, 2, 4… элементы. |
| :first | Выбирает только один элемент, первый из подходящих, например, $("p:first"). |
| :last | Выбирает только один элемент, последний из подходящих. |
| :first-child | Выбирает элементы, которые являются первыми дочерними элементами своих родителей. |
| :last-child | Выбирает элементы, которые являются последними дочерними элементами своих родителей. |
| :only-child | Выбирает элементы, являющиеся единственными дочерними элементами своих родителей. |
| :nth-child(n) | Выбирает элементы, которые являются n-дочерними элементами своих родителей. |
| :nth-child(Xn+Y) | Выбирает n-элемент, порядковый номер которого рассчитывается по формуле в круглых скобках. |
| :nth-of-type(n) | Выбирает элементы, являющиеся n-ми дочерними элементами данного типа для своих родителей. |
| :parent | Выбирает непустые элементы, которые имеют вложенные (дочерние) элементы, а также содержащие текст. |
| :eq(n) | Выбирает элементы с индексом n, при этом индексы отсчитываются от нуля. |

| | |
|---|---|
| :gt(n) | Выбирает все элементы, индекс которых больше n, индексы отсчитываются от нуля. |
| :lt(n) | Выбирает все элементы, расположенные перед n-элементом, не включая n-элемент. |
| :not(селектор) | Позволяет выбрать элемент, не соответствующий данному типу селектора, например, $("a:not(.link)"), $("a:not([href$=".pdf"])"). |
| :has(селектор) | Выбирает элементы, которые содержат внутри себя указанный селектор, например, элементы списка, содержащие внутри себя ссылки: $("li:has(a)"). |
| :contains(текст) | Выбирает элементы, которые содержат указанный в скобках текст, например, $("a:contains(Скачать)"). |
| :hidden | Выбирает скрытые элементы, для которых установлено значение display: none;, а также элементы форм со значением type="hidden" например, $("ul:hidden").show() — делает скрытые элементы видимыми. |
| :visible | Выбирает видимые элементы, к видимым элементам относятся элементы, размеры которых больше нуля, а также элементы со значением visibility: hidden и opacity: 0. |
| :active | Выбирает элемент, который активизирован пользователем, например, с помощью щелчка мыши. |
| :checked | Выбирает только отмеченные флажки или радиокнопки. |
| :focus | Выбирает элемент, находящийся в фокусе. |
| :hover | Выбирает элемент, на который наведен указатель мыши. |
| :disabled | Выбирает неактивные элементы (форм). |
| :enabled | Выбирает активные элементы (форм). |
| :empty | Выбирает элементы, не содержащие дочерних элементов. |

| | |
|---|---|
| :target | Выбирает элементы, на которые ссылается идентификатор фрагмента в url-адресе. |
| :animated | Выбирает все анимируемые в данный момент элементы. |
| :button | Выбираетвсекнопки input[type=submit], input[type=reset], input[type=button], button. |
| :checkbox | Выбирает элементы-флажки input[type=checkbox]. |
| :file | Выбирает элементы типа file, input[type=file]. |
| :header | Выбирает элементы-заголовки от h1 до h6. |
| :image | Выбирает изображения в элементах форм input[type=image]. |
| :input | Выбираетэлементыформ input, select, textarea, button. |
| :password | Выбирает элементы ввода пароля input[type=password]. |
| :radio | Выбирает радиокнопки input[type=radio]. |
| :reset | Выбираеткнопкисброса input[type=reset], button[type=reset]. |
| :selected | Выбирает выделенные элементы option. |
| :submit | Выбирает кнопки отправки формы input[type=submit], button[type=submit]. |
| :text | Выбирает элементы ввода текстаinput[type=text]. |

## ЗАДАНИЕ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Написать веб-приложение, которое будет выполнять следующее:

Таблица 3 – Варианты заданий

| № | Дано | Найти |
|---|---|---|
| 1. | Две стороны треугольника и угол между ними | Длину третьей стороны и площадь этого треугольника |
| 2. | Четырёхзначное число | Произведение всех цифр числа |
| 3. | Катеты прямоугольного треугольника | Периметр и площать треугольника |
| 4. | Два действительных числа | Среднее арифметическое и среднее геометрическое этих чисел |
| 5. | Два действительных числа | Среднее арифметическое этих чисел и среднее геометрическое их модулей |
| 6. | Гипотенуза и катет прямоугольного треугольника | Второй катет и радиус вписанной окружности |
| 7. | Внешний радиус кольца равен числу r (r>20) | Площадь кольца, внутренний радиус которого равен 20 |
| 8. | Длина окружности | Площадь круга |
| 9. | Основания и высота равнобедренной трапеции | Её периметр и площадь |
| 10. | Три положительных числа | Дробную часть среднего арифметического трех заданных положительных чисел |
| 11. | Сторона равностороннего треугольника | Площадь и периметр треугольника |
| 12. | Сторона квадрата. В квадрат вписана окружность | Площадь квадрата и окружности, вписанной в этот квадрат |
| 13. | Радиус окружности. В окружность вписан квадрат | Найти площади окружности и квадрата |
| 14. | Пятизначное число | Сумму всех цифр |
| 15. | Четырёхзначное число | Записать цифры числа в обратной последовательности |

Разберём выполнение работы на примере реализации калькулятора.

Первым делом нужно придумать макет, показывающий веб-интерфейс. На рисунке 3 представлено, как будет выглядеть калькулятор.

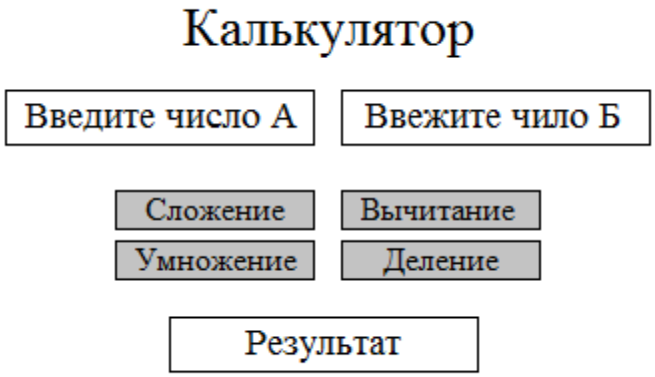


Рисунок 3 – Макет веб-интерфейса

Так же перед написанием кода нужно продумать логику работы программы. В два верхних поля будут вводиться числа, над которыми будут совершаться операции при нажатии одной из кнопок.

Хоть калькулятор на 4 действия и является простейшей задачей, но и тут есть нюансы, а именно необходимо перед проведением действий над числами нужно проверить, не пустые ли поля для ввода чисел и проверить то, чтобы в поля были введены именно числа, а не буквы или другие символы.

Так же для действия деление нужно проверять, чтобы во втором поле не был введён ноль.

Теперь можно приступать к написанию кода, но предварительно нужно создать необходимые папки и файлы.

Например, создаём папку с названием «Калькулятор», в которой будут находиться папки и файлы, указанные на рисунке 4.

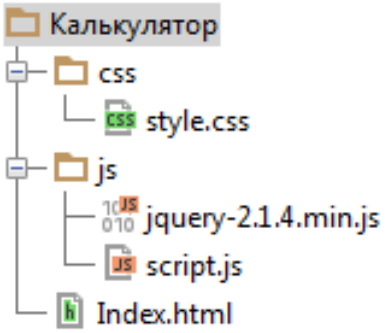


Рисунок 4 – Набор необходимых файлов.

Теперь можно приступать к написанию кода.

В файле index.html используются 4 кнопки и три формы ввода данных. Текст кода содержащегося в index.html и некоторые комментарии представлены ниже:

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>Калькулятор</title>
   <link href="css/style.css" rel="stylesheet">
   <script src="js/script.js"></script>
   <script src="js/jquery-2.1.4.min.js"></script>
</head>
<body>
   <div class="center">
      <h4>Калькулятор</h4>
      <input id="dataA" class="center" type="text"
maxlength="10" placeholder="Введите число А">  <!-- поле для первого числа -->
      <input id="dataB" class="center" type="text"
maxlength="10" placeholder="Введите число Б">  <!-- поле для второго числа -->
      <br>  <!-- перевод на следующую строку -->
      <br>
      <button type="submit" class="slojenie" onclick="slojenie()"
>Сложение</button>       <!-- кнопка сложения -->
```

```
            <button type="submit" class="vychitanie" on-
click="vychitanie()" >Вычитание</button>   <!-- кнопка вычитания --
>
            <br>
            <br>
            <button type="submit" class="umnijenie" on-
click="umnijenie()" >Умножение</button>     <!-- кнопка
умножения-->
            <button type="submit" class="delenie" onclick="delenie()"
>Деление</button>          <!-- кнопка деления -->
            <br>
            <br>
            <input id="result" class="center" type="text" value=""
readonly placeholder="Результат"> <!-- поле для вывода результата -
->
        </div>
    </body>
    </html>
```

Стили из файла style.css применяемые к html элементам представлены ниже:

```
    .center {
        text-align: center;     /* отцентровать объект */
    }
    h4 {
        font-size: 30px;        /* размер шрифта */
        color: blue;            /* цвет */
    }
    #dataA {
        width: 150px;           /* ширина поля */
    }
    #dataB {
        width: 150px;
    }
    .slojenie {
        width: 100px;           /* ширина кнопки */
    }
    .vychitanie {
```

```
    width: 100px;
}
.umnijenie {
    width: 100px;
}
.delenie {
    width: 100px;
}
```

Остаётся только реализовать логику программы.

Чтобы при нажатии на кнопку происходило действие, нужно в тег button добавить onclick="slojenie()", где slojenie – это имя вызываемой функции.

В самом файле script.js прописывается function slojenie() { }, где в фигурных скобках прописываются необходимые действия.

Для начала нужно считать переменную из input. Для этого можно воспользоваться двумя способами:

– var a=document.getElementById('dataA').value; – считывание переменной с помощью JavaScript;

– var a=$("#dataA").val(); – считывание переменной с помощью библиотеки jQuery.

var – используется для объявления переменной.

Далее используем функцию parseFloat(), которая принимает строку в качестве аргумента и возвращает десятичное число. Эта функция, если будет введены какие-то символы отличные от цифр, вернёт NaN. Это поможет сделать простую проверку на правильность ввода символов. А именно применим такой код:

```
    if (isNaN(c) || isNaN(d)) {
    alert("Введите числа!");
}
```

После можно будет делать нужные операции над переменными.

Для того чтобы вывести результат, есть два варианта:

– document.getElementById("result").value = result; – с помощью JavaScript;

– $("#result").val(result); – с помощью библиотеки jQuery.

Полный текст скрипта представлен ниже:

```
function slojenie() {
    var a=document.getElementById('dataA').value;
    var b=document.getElementById('dataB').value;
    c=parseFloat(a,10);
    d=parseFloat(b,10);

    if (isNaN(c) || isNaN(d)) {
        alert("Введите числа!");
    }
    else  {
        result=c+d;
        document.getElementById("result").value = result;
    }
}
function vychitanie() {
    var a=document.getElementById('dataA').value;
    var b=document.getElementById('dataB').value;
        c=parseFloat(a,10);
    d=parseFloat(b,10);

    if (isNaN(c) || isNaN(d)) {
        alert("Введите числа!");
    }
    else  {
        result=c-d;
        document.getElementById("result").value = result;
    }
}
function umnijenie() {
    var a=$("#dataA").val();
    var b=$("#dataB").val();
    c=parseFloat(a,10);
    d=parseFloat(b,10);

    if (isNaN(c) || isNaN(d)) {
        alert("Введите числа!");
    }
```

```
        else  {
            result=c*d;
            $("#result").val(result);
        }
    }
    function delenie() {
        var a=$("#dataA").val();
        var b=$("#dataB").val();
        c=parseFloat(a,10);
        d=parseFloat(b,10);

        if (isNaN(c) || isNaN(d)) {
            alert("Введите числа!");
        }
        else if (d==0) {
            alert("Мы не в комплексной области! Здесь делить на 0 нельзя!")
        }
        else if (d!==0) {
            result=c/d;
            $("#result").val(result);
        }
    }
```

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Муссиано Ч., Кеннеди Б. HTML и XHTML. Подробное руководство. М. : Символ-Плюс, 2008. – 748 с.

2) Мейер Э. CSS – Каскадные таблицы стилей. Подробное руководство. М. : Символ-Плюс, 3-е издание, 2008. – 575 с.

3) Шмитт К. CSS: Рецепты программирования. СПб. : БХВ-Петербург, 3-е издание, 2011. – 372.

4) Буч Г. Объектно-ориентированный анализ и проектирование. – М.: Вильямс, 2008.

5) Леоненков А.В. Самоучитель языка UML. – СПб.: БХВ-Петербург, 2004.

6) Розенберг Д., Скотт К. Применение объектного моделирования с использованием UML и анализ прецедентов. – М.: ДМК Пресс, 2002.