МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ

РЕСПУБЛИКИ УЗБЕКИСТАН

ФЕРГАНСКИЙ ФИЛИАЛ
ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ ИМЕНИ МУХАММАДА АЛ-ХОРАЗМИЙ

МЕТОДИЧЕСКИЕУКАЗАНИЯ

к выполнению лабораторных работ
по дисциплине

СОЗДАНИЕ ВЕБ ПРИЛОЖЕНИЙ

Сфера образование: ***330000 - Компьютерные технологии и информатика***

Направлениеобразование: ***5330500 – Компьютерная инжиниринг (“Компьютерная инжиниринг”, “Сервис ИТ”)***

Фергана -2017

Лабораторная работа №1

Установка среды созданий веб приложений

1. Цель работы

Исследование установки среды созданий веб приложений на персональный компьютер

2. Краткие сведения из теории

Web-сайт состоит из связанных между собой Web-страниц. Web-страница представляет собой текстовый файл с расширением *.htm, который содержит текстовую информацию и специальные команды – HTML-коды, определяющие в каком виде эта информация будет отображаться в окне браузера. Вся графическая, аудио и видео информация непосредственно в Web-страницу не входит и представляет собой отдельные файлы.

Современные web-приложения — это, в основном, порталы, предоставляющие услуги, которыми нельзя воспользоваться откуда-либо еще, Одно из неудобств подобных сервисов — сложность обмена информацией между компаниями. В частности, даже контактную и другую личную информацию приходится на каждом таком сайте вводить заново.

Другие наиболее распространённые web-приложения:

- Региональные Интернет-порталы, универсальные по своему направлению, но ограниченные географией заинтересованных посетителей (e1.ru);
- Поисковые системы — это Интернет-порталы, которые предназначены для того, чтобы предоставить их посетителю возможность найти сайты, на которых встречаются заданные слова или целые фразы (metabot.ru);
- Каталог — это коллекция ссылок на сайты. Зачем же нужны каталоги, если есть поиск? Очень часто мы не знаем точно, что нам нужно, не можем это сформулировать парой слов (mail.ru);
- Электронные доски объявлений - являются местом в Интернет, где практически любой желающий может оставить информацию ознакомительного, пригласительного или рекламного характера;
- Форумы — это специальные сайты или разделы на сайтах, предназначенные для того, чтобы посетители, оставляя свои сообщения, обменивались мнениями;
- Чаты - являются еще одним местом для общения в Интернет, только его назначение не обмен мнениями на какую-то тему, а просто времяпрепровождение;
- Файлы для скачивания;
- Фотогалереи;
- Элементы статистики;
- Хранение в интернете различной информации;

- Серверы почтовых рассылок, они предлагают услуги по доставке информации широкому кругу читателей (subscribe.ru);
- Интернет-магазины и аукционы (ozon.ru, molotok.ru).

3. Порядок выполнения работы

Установка среди созданий веб приложений emmet-sublime-master или Notepad++. Создание проекта HTML на среде созданий веб приложений emmet-sublime-master или Notepad++.

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

4. Вопросы для самопроверки

1. Цель и задачи криптографии.
2. Шифры одиночной перестановки и перестановки по ключевому слову.
3. Шифры двойной перестановки. Шифрование с помощью магического квадрата.

Рекомендуемая литература

1. Бурлаков М. В. CorelDRAW 12. – СПб.; БХВ-Петербург, 2004. – 688 с.
2. Джамса Крис. Эффективный самоучитель по креативному Web-дизайну. HTML, XHTML, CSS,JavaScript, PHP, ASP, ActiveX. Текст, графика, звук и анимация. Пер с англ./Крис Джамса, Конрад Кинг, Энди Андерсон - М.: ООО "ДиаСофтЮП", 2005.- 672 с.
3. Инькова Н. А., Зайцева Е. А., Кузьмина Н. В., Толстых С. Г. Создание Web-сайтов: Учебно-методическое пособие. Ч. 5. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2005. – 56 с.
4. Орлов Л. В. Web-сайт без секретов. / Л. В. Орлов. – 2-е изд. – М.: Бук-пресс, 2006. – 512 с.
5. Полонская Е.Л. Язык HTML. Самоучитель.: - М.: Издательский дом "Вильяме", 2005.— 320 с.
6. Создание Web-страниц и Web-сайтов. Самоучитель : [учеб. пособие] / под ред. В. Н. Печникова. – М.: Изд-во Триумф, 2006.— 464 с.
7. Якушев, Л. В. Начинаем работать в Интернет. Краткое руководство. – М.: Издательский дом "Вильяме", 2006. —128 с

Лабораторная работа №2

Основы HTML

1. Цель работы

Изучение основных концепций HTML, необходимых для понимания принципов разработки веб-документов, исследование языка разметки гипертекста и его основных тегов,научиться создавать веб страницы используя HTML тегов.

2. Краткие сведения из теории

Hyper Text Markup Language (HTML) — язык разметки гипертекста — предназначен для написания гипертекстовых документов, публикуемых в World Wide Web.

Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Язык HTML является приложением SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879.

Гипертекстовый документ — это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла па экране компьютера.

С помощью этих меток можно выделять заголовки документа, изменять цвет, размер и начертание букв, вставлять графические изображения и таблицы. Но основным преимуществом гипертекста перед обычным текстом является возможность добавления к содержимому документа гиперссылок — специальных конструкций языка HTML, которые позволяют щелчком мыши перейти к просмотру другого документа.

HTML-документ состоит из двух частей: собственно текста, т. е. данных, составляющих содержимое документа, и ***тегов*** — специальных конструкций языка HTML, используемых для разметки документа и управляющих его отображением. Теги языка HTML определяют, в каком виде будет представлен текст, какие его компоненты будут исполнять роль гипертекстовых ссылок, какие графические или мультимедийные объекты должны быть включены в документ.

Графическая и звуковая информация, включаемая в HTML-документ, хранится в отдельных файлах. Программы просмотра HTML-документов (**браузеры**) интерпретируют флаги разметки и располагают текст и графику на экране соответствующим образом. Для файлов, содержащих HTML-документы приняты расширения **.htm** или **.html**.

В большинстве случаев теги используются парами. Пара состоит из открывающего <имя_тега> и закрывающего </имя_тега> тегов. Действие любого парного тега начинается с того места, где встретился открывающий тег, и заканчивается при встрече соответствующего закрывающего тега. Часто пару, состоящую из открывающего и закрывающего тегов, называют *контейнером*, а

часть текста, окаймленную открывающим и закрывающим тегом, — *элементом*.

Последовательность символов, составляющая текст может состоять из пробелов, табуляций, символов перехода на новую строку, символов возврата каретки, букв, знаков препинания, цифр, и специальных символов (например #, +, $, @), за исключением следующих четырех символов, имеющих в HTML специальный смысл: < (меньше), > (больше), & (амперсанд) и " (двойная кавычка). Если необходимо включить в текст какой-либо из этих символов, то следует закодировать его особой последовательностью символов.

Самым главным из тегов HTML является одноименный тег ***<html>***. Он всегда открывает документ, так же, как тег ***</html>*** должен непременно стоять в последней его строке. Эти теги обозначают, что находящиеся между ними строки представляют единый гипертекстовый документ. Без этих тегов браузер или другая программа просмотра не в состоянии идентифицировать формат документа и правильно его интерпретировать.

HTML-документ состоит из двух частей: заголовок (head) и тела (body), расположенных в следующем порядке:

<html>
<head> Заголовок документа </head>
<body> Тело документа </body>
</html>

Чаще всего в заголовок документа включают парный тег <title>... </title>, определяющий название документа. Многие программы просмотра используют его как заголовок окна, в котором выводят документ. Программы, индексирующие документы в сети Интернет, используют название для идентификации страницы. Хорошее название должно быть достаточно длинным для того, чтобы можно было корректно указать соответствующую страницу, и в то же время оно должно помещаться в заголовке окна. Название документа вписывается между открывающим и закрывающим тегами.

Тело документа является обязательным элементом, так как в нем располагается весь материал документа. Тело документа размещается между тегами <body> и </body>. Все, что размещено между этими тегами, интерпретируется браузером в соответствии с правилами языка HTML позволяющими корректно отображать страницу на экране монитора.

Текст в HTML разделяется на абзацы при помощи тега . Он размещается в начале каждого абзаца, и программа просмотра, встречая его, отделяет абзацы друг от друга пустой строкой. Использование закрывающего тега необязательно.

Если требуется «разорвать» текст, перенеся его остаток на новую строку, при этом, не выделяя нового абзаца, используется тег разрыва строки
. Он заставляет программу просмотра выводить стоящие после него символы с новой строки. В отличие от тега абзаца, тег
 не добавляет пустую строку. У этого тега нет парного закрывающего тега.

Язык HTML поддерживает **логическое н физическое форматирование содержимого документа**. Логическое форматирование указывает на

назначение данного фрагмента текста, а физическое форматирование задает его внешний вид.

При использовании *логического форматирования* текста браузером выделяются различные части текста в соответствии со структурой документа. Чтобы отобразить название, используется один из тегов заголовка. Заголовки в типичном документе разделяются по уровням. Язык HTML позволяет задать шесть уровней заголовков: h1 (заголовок первого уровня), h2, h3, h4, h5 и h6. Заголовок первого уровня имеет обычно больший размер и насыщенность по сравнению с заголовком второго уровня. Пример использования тегов заголовков:

<hl>l. Название главы</hl>
<h2>l.l. Название раздела</h2>

Теги *физического форматирования* непосредственно задают вид текста на экране браузера, например пара выделяет текст полужирным начертанием, <u></u>задает подчеркивание текста, управляет шрифтом текста.

Тег <img> вставляет изображение в документ, как если бы оно было просто одним большим символом. Пример применения тега:

<img src = "picture.gif">

Для создания **гипертекстовой ссылки** используется пара тегов <a>... **.** Фрагмент текста, изображение или любой другой объект, расположенный между этими тегами, отображается в окне браузера как гипертекстовая ссылка. Активация такого объекта приводит к загрузке в окно браузера нового документа или к отображению другой части текущей Web-страницы. Гипертекстовая ссылка формируется с помощью выражения:

ссылка на документ

Href здесь является обязательным атрибутом, значение которого и есть URL-адрес запрашиваемого ресурса. Кавычки в задании значения атрибута href не обязательны. Если задается ссылка на документ на другом сервере, то вид гиперссылки такой:

Фотография 11-А

С помощью различных тегов можно рисовать таблицы, форматировать текст, вставлять в документ изображения, видео- , звуковые файлы и прочее.

Регистр, в котором набрано имя элемента и имена атрибутов, в HTML значения не имеет (в отличие от XHTML). Элементы могут быть вложенными. Например, следующий код:

```
<!DOCTYPEhtml>
<html>
<head>
<metahttp-equiv="Content-Type"content="text/html; charset=utf-8"/>
<title>HTML Document</title>
</head>
<body>
```

```
<p>
<b>
        Этот текст будет полужирным,
<i>а этот — ещё и курсивным</i>
</b>
</p>
</body>
</html>
```

даст такой результат:

Этот текст будет полужирным, *а этот — ещё и курсивным*

3. Порядок выполнения работы

Выполнить задание которые даны внизу по вариантам.

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

4. Варианты для выполнение задачи

1. Сделайте страницу, показанную на рис. 1.

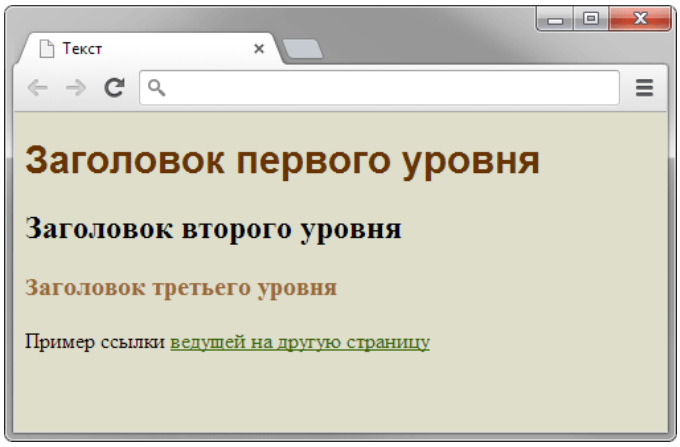


Рис. 1

2. Сделайте текст, как показано на рис. 2. В качестве шрифта укажите Impact.

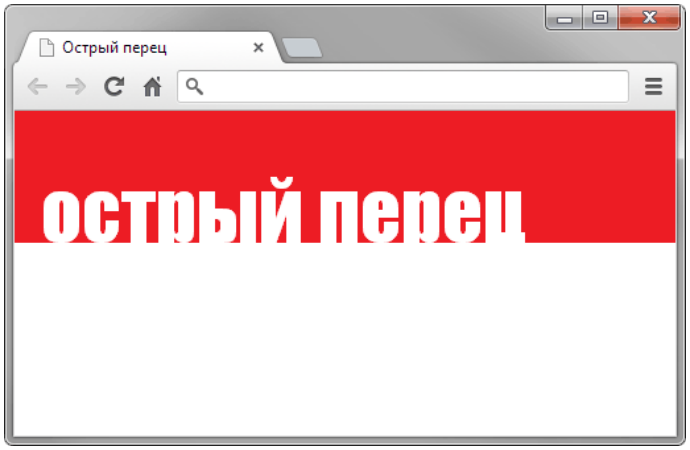
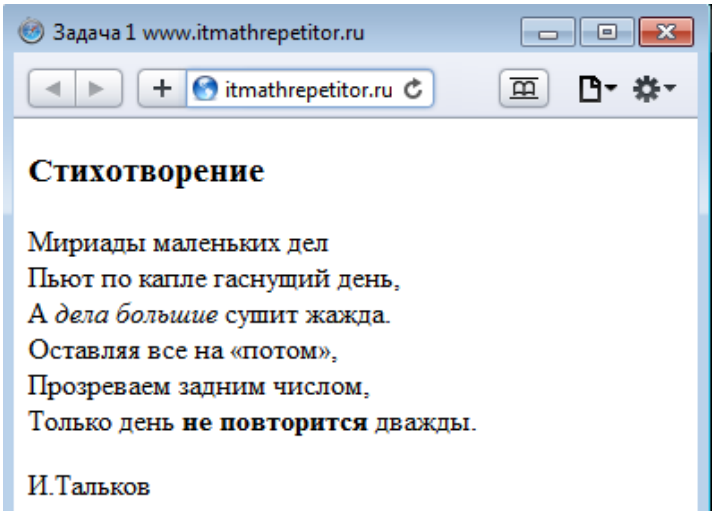
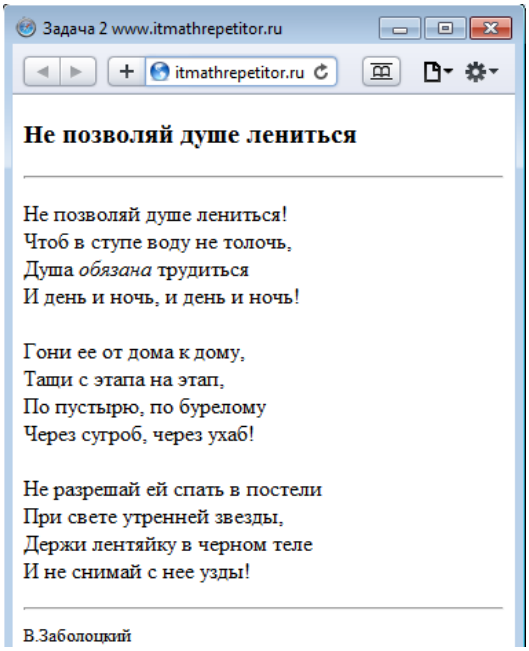


Рис. 2

3. Создайте html-файл (кодировка utf-8) с заголовком "Задача 3", результат которого показан на рисунке. Используйте подходящий тег заголовков (<h1>-<h6>), теги
, <p>, <i>, , , <b>. Выясните различие тегов <b> и , тегов <i> и . В html-код добавьте комментарии: дату решения данной задачи и ФИО.



4. Создайте html-файл (кодировка utf-8) с заголовком "Задача 4", результат которого показан на рисунке. Используйте подходящий тег заголовков (h1-h6), теги
, <p>, <i>, , , <b>, <hr>, <small>. Обратите внимание, что шрифт фамилии автора меньшего размера. Заметим, что тег <hr> в разных браузерах может отображаться по-разному. В html-код добавьте условие данной задачи в виде комментариев.



5. Создать таблицу, в которой объединено несколько ячеек в строке и несколько ячеек в столбце.
6. Создать текст, в котором будут использованы различные методы выделения, в т.ч. курсив, полужирный, подчеркнутый текст.
7. Вывести на экран формулу, содержащую переменные с индексами, возведение в степень и греческие буквы.
8.Вывести на экран формулу с определителем второго порядка без использования таблиц.
9.Вывести на экран формулу, содержащую определитель второго порядка, с использованием таблиц.
10. Вывести на экран простые списки: список определений, нумерованный и ненумерованный списки.
11. Создать документ HTML, в котором есть гиперссылки на страницы, расположенные: в текущем каталоге, в каталоге на уровень выше, во вложенном каталоге, на указанном домене. А также использовать переход на анкер (якорь).
12. Создайте документ HTML, в котором есть гиперссылки на страницы и опишите параметры тега BODY: **MARGINHEIGHT**, **TOPMARGIN**, **MARGINWIDTH**, **LEFTMARGIN**, **BACKGROUND**, **BGCOLOR**, **TEXT**, **LINK.**
13. Создайте документ HTML, в котором есть гиперссылки на страницы и опишите параметры тега font
14. Создайте документ HTML, в котором есть гиперссылки на страницы и опишите параметры тега table
15. Создайте документ HTML, в котором есть гиперссылки на страницы и опишите параметры тега frame

Лабораторная работа №3

Использование возможностей HTML5

1. Цель работы

Исследование языка разметки гипертекста HTML5 и его дополнительных тегов, научиться создавать веб страницы используя теговHTML5

2. Краткие сведения из теории

HTML5 (англ. *HyperTextMarkupLanguage, version 5*) — язык для структурирования и представления содержимого всемирной паутины. Это пятая версия HTML. Хотя стандарт был завершён (рекомендованная версия к использованию) только в 2014 году (предыдущая, четвёртая, версия опубликована в 1999 году), ещё с 2013 года браузерами оперативно осуществлялась поддержка, а разработчиками — использование рабочего стандарта (англ. *HTMLLivingStandard*). Цель разработки HTML5 — улучшение уровня поддержки мультимедиа-технологий с одновременным сохранением обратной совместимости, удобочитаемости кода для человека и простоты анализа для парсеров.

В HTML5 реализовано множество новых синтаксических особенностей. Например, элементы <video>, <audio> и <canvas>, а также возможность использования SVG и математических формул. Эти новшества разработаны для упрощения создания и управления графическими и мультимедийными объектами в сети без необходимости использования сторонних API и плагинов. Другие новые элементы, такие как <section>, <article>, <header> и <nav>, разработаны для того, чтобы обогащать семантическое содержимое документа (страницы). Новые атрибуты были введены с той же целью, хотя ряд элементов и атрибутов был удалён. Некоторые элементы, например <a>, <menu> и <cite>, были изменены, переопределены или стандартизированы. API и DOM стали основными частями спецификации HTML5. HTML5 также определяет некоторые особенности обработки ошибок вёрстки, поэтому синтаксические ошибки должны рассматриваться одинаково всеми совместимыми браузерами.

HTML5 вводит несколько новых элементов и атрибутов, которые отражают типичное использование разметки на современных веб-сайтах. Некоторые из них — семантические замены для использования универсальных блочных ( ) и строчных () элементов, например, <nav> (блок навигации по сайту), <footer> (обычно относится к нижней части страницы или последней строке HTML кода) или <audio> и <video> вместо <object>. Некоторые устаревшие элементы, которые можно было использовать в HTML 4.01, были исключены, включая чисто оформительские элементы, такие как <font> и , чьи эффекты выполняются с помощью каскадных таблиц стилей. Также в поведении веб снова заострено внимание на важности скриптов DOM (например, Javascript).

Немного связанных технологий, которые не являются частью ни одной из спецификаций, следуют далее. W3C публикует спецификации для них отдельно.

- геолокация;
- база данных SQL для Web, внутренняя база данных (больше не поддерживаемая);
- Индексированная база данных (IndexedDB) API, индексирование по типу ключ-значение (прежде — WebSimpleDB);
- Файл API, дескриптор обновления файлов и управления ими;
- Работа с системой. Этот API предназначен для того, чтобы обеспечить хранение информации со стороны клиента без управления базами данных;
- Запись в файл, использование API для записи в файл информации из приложения.

Основным нововведением стало внесение давно ожидаемых новых элементов, благодаря которым содержание выглядит чуть более семантичным, хотя только разве что в блогах. Хитрые дизайнерские сайты с плавающими блоками видимо лучше делать на более общих div'ах

Структурные теги

- header — понятное дело, шапка с логотипом, логином, навигацией..
- nav — навигация.. как меню, так и «хлебные крошки»
- footer — подвал с копирайтами, контактами
- article — очевидно влияние блогосферы и rss
- section — подраздел документа
- aside — боковая панель (видимо с коментариями, самыми популярными статьями, тэгами и тп)

Теги содержания

- video и audio — как альтернатива flash-плеерам. Врядли заменит флеш из-за проблем с кодеками и отсутствии поддержки уже созданных .flv видео. Мало поддерживается пока браузерами.
- progress — полоса завершённости процесса. Полезно при заполнении форм
- time — полезно для указания точного времени элемента
- details — просто дополнительная информация

datalist — нечто типа автозаполнения, но с прописанными вариантами

Пример веб-страницы HTML5

```
<!DOCTYPE html>

<html>
<head>
<metacharset="utf-8">
<title>(Это title) Пример страницы на HTML5</title>
</head>
```

<**body**>
<**header**>
<**hgroup**>
<**h1**>Заголовок "h1" из hgroup</**h1**>
<**h2**> Заголовок "h2" из hgroup
</**h2**>
</**hgroup**>
</**header**>
<**nav**>
<**menu**>
<**li**>
<**a**href="link1.html">
Первая ссылка из блока "nav"
</**a**>
</**li**>
<**li**>
<**a**href="link2.html">
Вторая ссылка из блока "nav"
</**a**>
</**li**>
</**menu**>
</**nav**>
<**section**>
<**article**>
<**h1**>
Заголовокстатьиизблока "article"
</**h1**>
<**p**>
Текст абзаца статьи из блока "article"
</**p**>
<**details**>
<**summary**>
Блок "details", текстгега "summary"
</**summary**>
<**p**>
Абзацизблока "details"
</**p**>
</**details**>
</**article**>
</**section**>
<**footer**>
<**time**>
Содержимоетега "time" блока "footer"
</**time**>

```
<p>
        Содержимое абзаца из блока "footer"
</p>
</footer>
</body>
</html>
```

3. Порядок выполнения работы

Выполнить задание которые даны внизу по вариантам.

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

4. Варианты для выполнение задачи

Сделайте тот же макет с использованием таблиц.

Вариант №1.

<table>
<tr><td>1.</td><td colspan="2">2.</td><td>3.</td><td>4.</td></tr>
<tr><td rowspan="2">5.</td><td>6.</td><td colspan="3">7.</td></tr>
<tr><td>8.</td><td>9.</td><td>10.</td><td>11.</td></tr>
<tr><td colspan="5">12.</td></tr>
</table>

Вариант № 2.

<table>
<tr><td colspan="2">1.</td><td>2.</td><td>3.</td></tr>
<tr><td rowspan="3">4.</td><td>5.</td><td colspan="2">6.</td></tr>
<tr><td>7.</td><td>8.</td><td>9.</td></tr>
<tr><td>10.</td><td colspan="2">11.</td></tr>
</table>

Вариант № 3.

<table>
<tr><td>1.</td><td>2.</td><td>3.</td><td colspan="2">4.</td></tr>
<tr><td colspan="2">5.</td><td colspan="3">6.</td></tr>
<tr><td>7.</td><td>8.</td><td>9.</td><td>10.</td><td rowspan="2">11.</td></tr>
<tr><td colspan="2">12.</td><td colspan="2">13.</td></tr>
</table>

Вариант № 4.

<table>
<tr><td>1.</td><td>2.</td><td colspan="2">3.</td><td>4.</td></tr>
<tr><td colspan="2">5.</td><td>6.</td><td colspan="2">7.</td></tr>
<tr><td rowspan="2">8.</td><td colspan="2">9.</td><td colspan="2">10.</td></tr>
<tr><td colspan="3">11.</td><td>12.</td></tr>
</table>

Вариант № 5.

<table>
<tr><td>1.</td><td colspan="2">2.</td><td>3.</td><td>4.</td></tr>
<tr><td rowspan="2">5.</td><td>6.</td><td colspan="3">7.</td></tr>
<tr><td>8.</td><td>9.</td><td>10.</td><td>11.</td></tr>
<tr><td colspan="5">12.</td></tr>
</table>

Вариант № 6.

<table>
<tr><td colspan="2">1.</td><td>2.</td><td>3.</td></tr>
<tr><td rowspan="3">4.</td><td>5.</td><td colspan="2">6.</td></tr>
<tr><td>7.</td><td>8.</td><td>9.</td></tr>
<tr><td>10.</td><td colspan="2">11.</td></tr>
</table>

Вариант № 7.

<table>
<tr><td rowspan="4">1.</td><td>2.</td><td>3.</td><td>4.</td></tr>
<tr><td colspan="2">5.</td><td>6.</td><td>7.</td></tr>
<tr><td>8.</td><td colspan="2">9.</td><td>10.</td></tr>
<tr><td>11.</td><td colspan="3">12.</td></tr>
</table>

Вариант № 8.

<table>
<tr><td colspan="5">1.</td></tr>
<tr><td>2.</td><td>3.</td><td rowspan="2">4.</td><td>5.</td><td rowspan="2">6.</td></tr>
<tr><td colspan="2">7.</td><td>8.</td></tr>
<tr><td>9.</td><td colspan="3">10.</td><td>11.</td></tr>
</table>

Вариант № 9.

<table>
<tr><td rowspan="2">1</td><td>2</td><td rowspan="2">3</td><td>4</td><td rowspan="2">5</td></tr>
<tr><td>6</td><td>7</td></tr>
<tr><td>8</td><td colspan="2">9</td><td>10</td></tr>
</table>

Вариант № 10.

<table>
<tr><td rowspan="3">1.</td><td>2.</td><td rowspan="3">3.</td><td colspan="2">4.</td></tr>
<tr><td>5.</td><td>6.</td><td>7.</td></tr>
<tr><td>8.</td><td colspan="2">9.</td></tr>
</table>

Вариант № 11.

<table>
<tr><td colspan="2">1.</td><td>2.</td><td>3.</td><td rowspan="2">4.</td></tr>
<tr><td>5.</td><td>6.</td><td colspan="2">7.</td></tr>
<tr><td colspan="2">8.</td><td>9.</td><td>10.</td><td>11.</td></tr>
</table>

Вариант № 12.

<table>
<tr><td>1.</td><td rowspan="2">2.</td><td>3.</td><td rowspan="2">4.</td><td>5.</td></tr>
<tr><td rowspan="2">6.</td><td rowspan="2">7.</td><td>8.</td></tr>
<tr><td>9.</td><td colspan="2">10.</td></tr>
</table>

Вариант № 13.

<table>
<tr><td colspan="2">1.</td><td>2.</td><td>3.</td></tr>
<tr><td>4.</td><td rowspan="2">5.</td><td colspan="2">6.</td></tr>
<tr><td>7.</td><td>8.</td><td>9.</td><td>10.</td></tr>
</table>

Вариант № 14.

<table>
<tr><td rowspan="3">1.</td><td>2.</td><td rowspan="3">3.</td><td>4.</td><td>5.</td></tr>
<tr><td>6.</td><td colspan="2">7.</td></tr>
<tr><td>8.</td><td>9.</td><td>10.</td></tr>
</table>

Вариант № 15.

<table>
<tr><td>1.</td><td colspan="2">2.</td><td colspan="2">3.</td></tr>
<tr><td>4.</td><td rowspan="2">5.</td><td>6.</td><td rowspan="2">7.</td><td>8.</td></tr>
<tr><td>9.</td><td>10.</td><td>11.</td></tr>
</table>

Вариант № 16.

<table>
<tr><td rowspan="2">1.</td><td>2.</td><td>3.</td><td rowspan="2">4.</td><td>5.</td></tr>
<tr><td>6.</td><td rowspan="2">7.</td><td>8.</td></tr>
<tr><td colspan="2">9.</td><td colspan="2">10.</td></tr>
</table>

Вариант № 17.

<table>
<tr><td rowspan="2">1.</td><td>2.</td><td rowspan="2">3.</td><td>4.</td><td rowspan="2">5.</td></tr>
<tr><td>6.</td><td>7.</td></tr>
<tr><td>8.</td><td colspan="2">9.</td><td>10.</td></tr>
</table>

Вариант № 18.

<table>
<tr><td>1.</td><td>2.</td><td>3.</td><td>4.</td><td>5.</td></tr>
<tr><td>6.</td><td rowspan="2">7.</td><td colspan="2">8.</td><td rowspan="2">9.</td></tr>
<tr><td>10.</td><td colspan="2">11.</td></tr>
</table>

Вариант № 19.

<table>
<tr><td rowspan="3">1.</td><td colspan="3">2.</td><td rowspan="3">3.</td></tr>
<tr><td>4.</td><td>5.</td><td>6.</td></tr>
<tr><td colspan="3">7.</td></tr>
</table>

Вариант № 20.

<table>
<tr><td colspan="2">1.</td><td rowspan="3">2.</td><td colspan="2">3.</td></tr>
<tr><td>4.</td><td>5.</td><td>6.</td><td>7.</td></tr>
<tr><td colspan="2">8.</td><td colspan="2">9.</td></tr>
</table>

Вариант № 21.

<table>
<tr><td rowspan="2">1.</td><td rowspan="2">2.</td><td>3.</td><td>4.</td><td>5.</td></tr>
<tr><td colspan="3">6.</td></tr>
<tr><td colspan="2"></td><td>7.</td><td>8.</td><td>9.</td></tr>
</table>

Вариант № 22.

<table>
<tr><td>1.</td><td>2.</td><td>3.</td><td rowspan="3">4.</td><td rowspan="3">5.</td></tr>
<tr><td>6.</td><td colspan="2">7.</td></tr>
<tr><td>8.</td><td>9.</td><td>10.</td></tr>
</table>

Вариант № 23.

<table>
<tr><td>1.</td><td rowspan="3">2.</td><td rowspan="3">3.</td><td colspan="2">4.</td></tr>
<tr><td>5.</td><td>6.</td><td>7.</td></tr>
<tr><td>8.</td><td colspan="2">9.</td></tr>
</table>

Вариант № 24.

<table>
<tr><td rowspan="3">1.</td><td rowspan="2">2.</td><td colspan="2">3.</td><td>4.</td></tr>
<tr><td colspan="2">5.</td><td>6.</td></tr>
<tr><td>7.</td><td colspan="2">8.</td><td>9.</td></tr>
</table>

Вариант № 25.

<table>
<tr><td rowspan="3">1.</td><td>2.</td><td>3.</td><td>4.</td><td rowspan="3">5.</td></tr>
<tr><td colspan="3">6.</td></tr>
<tr><td>7.</td><td>8.</td><td>9.</td></tr>
</table>

Рекомендуемая литература

1. Ломов А.Ю., HTML, CSS, Скрипты - практика создания сайтов. – Санкт-Петербург «БХВ-Петербург» 2006. - ISBN: 5-94157-698-6.
2. Питер Лабберс, Брайан Олберс, Фрэнк Салим. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений = Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development. — М.: «Вильямс», 2011. — С. 272. — ISBN 978-5-8459-1715-7.
3. СтивенХольцнер. HTML5 за 10 минут, 5-еиздание = Sams Teach Yourself HTML5 in 10 Minutes, 5th Edition. — М.: «Вильямс», 2011. — ISBN 978-5-8459-1745-4.
4. АрсенийМирный HTML5 против Flash-видео // UP Special : журнал. — 2010. — № 5. — С. 42—45.

Лабораторная работа №4

Использование возможности CSS

1. Цель работы

Исследование и изучение каскадных таблиц стилей, создание стилей и работа сними.

2. Краткие сведения из теории

CSS (англ. *CascadingStyleSheets — каскадные таблицы стилей*) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

Правила CSS пишутся на формальном языке CSS и располагаются в таблицах стилей, то есть таблицы стилей содержат в себе правила CSS. Эти таблицы стилей могут располагаться как в самом веб-документе, внешний вид которого они описывают, так и в отдельных файлах, имеющих формат CSS. (По сути, формат CSS — это обычный текстовый файл. В файле .css не содержится ничего, кроме перечня правил CSS и комментариев к ним.)
То есть, эти таблицы стилей могут быть подключены, внедрены в описываемый ими веб-документ четырьмя различными способами:

- когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством тега <link>, располагающегося в этом документе между тегами <head> и </head>. (Тег <link> будет иметь атрибут href, имеющий значением адрес этой таблицы стилей). Все правила этой таблицы действуют на протяжении всего документа;

```
<!DOCTYPE html>
<html>
<head>
    .....
<linkrel="stylesheet"href="style.css">
</head>
<body>
    .....
</body>
</html>
```

- когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством директивы @import, располагающейся в этом документе между тегами <style> и </style> (которые, в свою очередь, располагаются в этом документе между тегами <head> и </head>) сразу после тега <style>, которая также указывает (в своих скобках, после слова url) на адрес этой таблицы стилей. Все правила этой таблицы действуют на протяжении всего документа;

```
<!DOCTYPE html>
<html>
<head>
    .....
<stylemedia="all">
@importurl(style.css);
</style>
</head>
</html>
```

- когда таблица стилей описана в самом документе, она может располагаться в нём между тегами <style> и </style> (которые, в свою очередь, располагаются в этом документе между тегами <head> и </head>). Все правила этой таблицы действуют на протяжении всего документа;

```
<!DOCTYPE html>
<html>
<head>
    .....
<style>
body{
color:red;
}
</style>
</head>
<body>
```

```
    .....
</body>
</html>
```

- когда таблица стилей описана в самом документе, она может располагаться в нём в теле какого-то отдельного тега (посредством его атрибута style) этого документа. Все правила этой таблицы действуют только на содержимое этого тега.

```
<!DOCTYPE>
<html>
<head>
    .....
</head>
<body>
<pstyle="font-size: 20px; color: green; font-family: arial, helvetica, sans-serif">
.....
</p>
</body>
</html>
```

В первых двух случаях говорят, что к документу применены *внешние таблицы стилей*, а во вторых двух случаях — *внутренние таблицы стилей*.

Для добавления CSS к XML-документу, последний должен содержать специальную ссылку на таблицу стилей. Например:

```
<?xml-stylesheet type="text/css" href="style.css"?>
```

»).

Пример таблицы стилей (в таком виде она может быть либо размещена в отдельном файле .css либо же обрамлена тегами <style> и размещена в «шапке» той самой веб-страницы, на которую она действует):

```
p{
font-family:arial,helvetica,sans-serif;
}
h2{
font-size:20pt;
color:red;
background:white;
}
.note{
color:red;
background-color:yellow;
```

```
font-weight:bold;
}
p#paragraph1{
padding-left:10px;
}
a:hover{
text-decoration:none;}
#newsp{
color:blue;}
[type="button"]{
background-color:green;
}
```

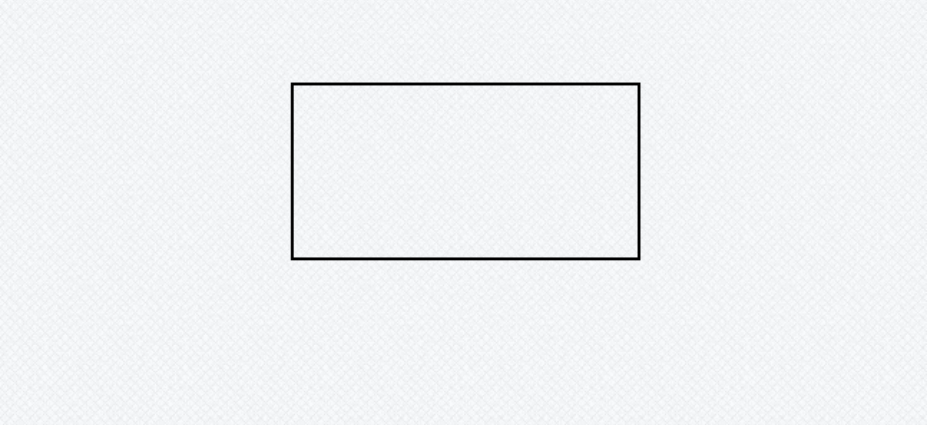
3. Порядок выполнения работы

Содержание отчета

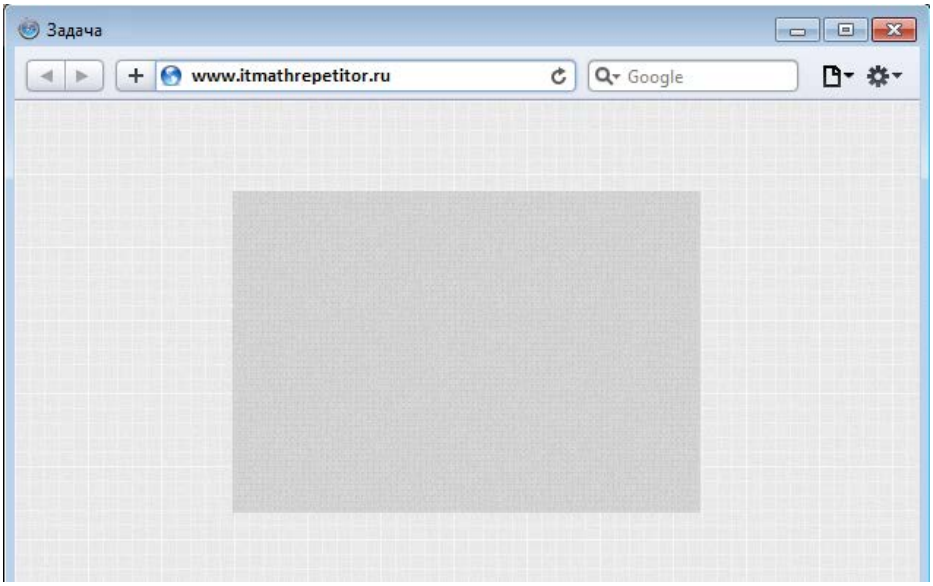
1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

4. Варианты для выполнение задачи

Вариант №1. Создайте html-файл, результат которого показан на рисунке. Блок с черной рамкой отцентрирован на странице по горизонтали.



Вариант №2. Создайте html-файл, результат которого показан на рисунке. Блок с темным фоном отцентрирован на странице по горизонтали и повертикали.

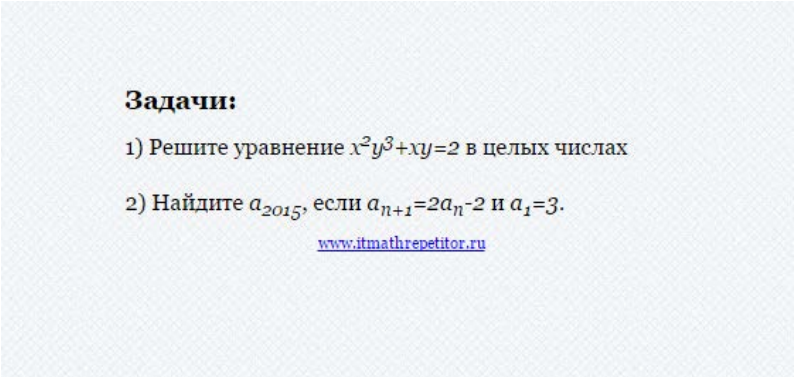


Вариант №3. Создайте html- и css- файлы, результат которых показан на рисунке.

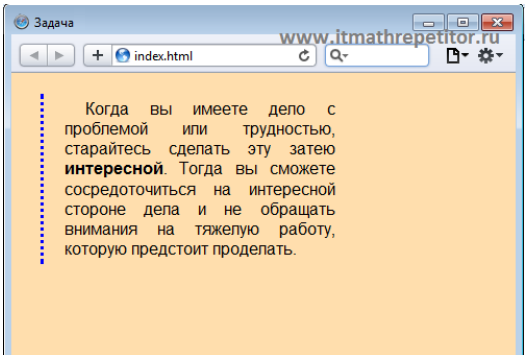
Вариант №4. Создайте html- и css- файлы, результат которых показан на рисунке.

Вариант №5. Создайте html- и css- файлы, результат которых показан на рисунке.

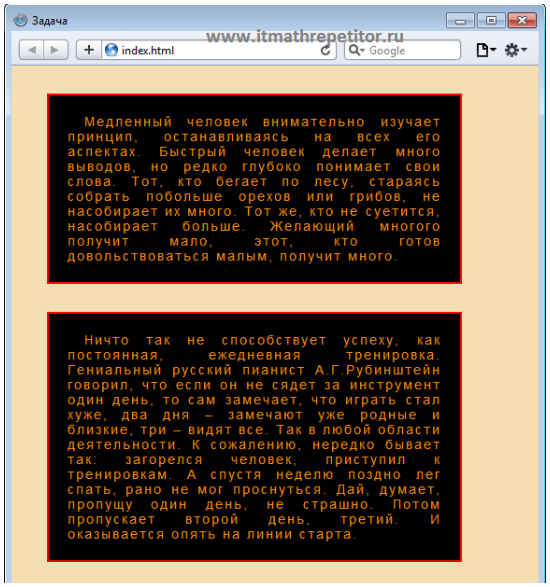
Вариант №6. Создайте html- и css- файлы, результат которых показан на рисунке.



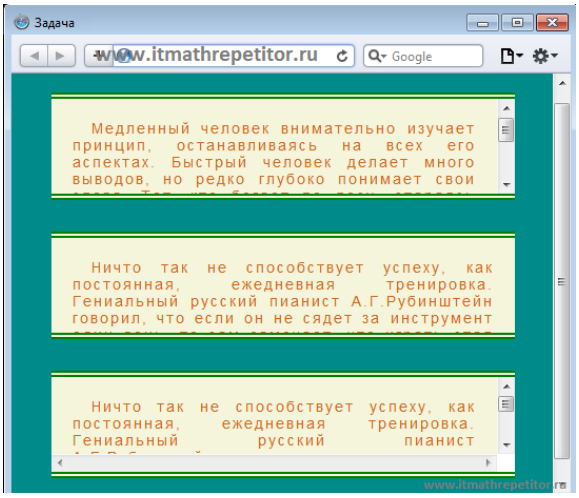
Вариант №7. Создайте html- и css- файлы, результат которых показан на рисунке.



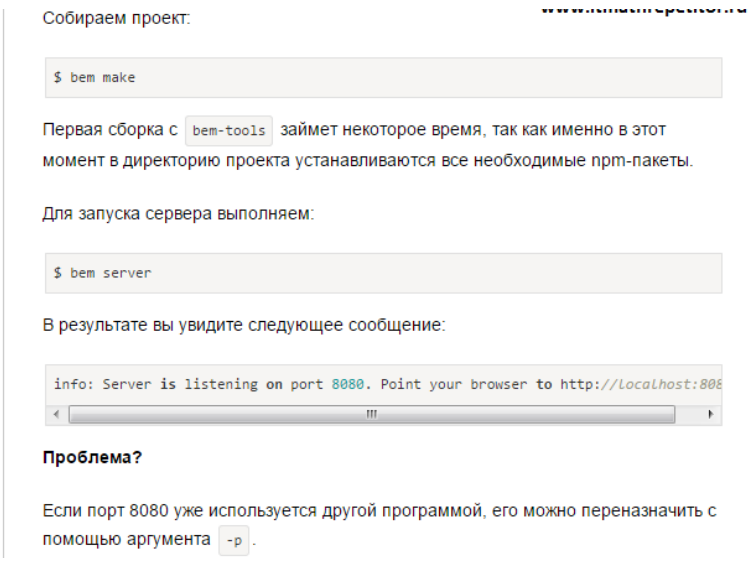
Вариант №8. Создайте html- и css- файлы, результат которых показан на рисунке.



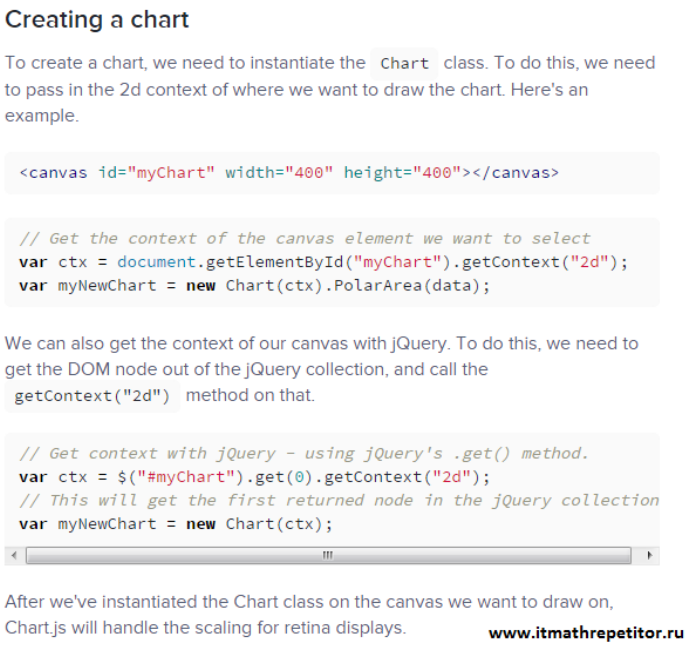
Вариант №9. Создайте html- и css- файлы, результат которых показан на рисунке.



Вариант №10. Создайте html- и css- файлы, результат которых показан на рисунке.



Вариант №11. Создайте html- и css- файлы, результат которых показан на рисунке.



Creating a chart

To create a chart, we need to instantiate the `Chart` class. To do this, we need to pass in the 2d context of where we want to draw the chart. Here's an example.

```
<canvas id="myChart" width="400" height="400"></canvas>
```

```
// Get the context of the canvas element we want to select
var ctx = document.getElementById("myChart").getContext("2d");
var myNewChart = new Chart(ctx).PolarArea(data);
```

We can also get the context of our canvas with jQuery. To do this, we need to get the DOM node out of the jQuery collection, and call the `getContext("2d")` method on that.

```
// Get context with jQuery - using jQuery's .get() method.
var ctx = $("#myChart").get(0).getContext("2d");
// This will get the first returned node in the jQuery collection
var myNewChart = new Chart(ctx);
```

After we've instantiated the Chart class on the canvas we want to draw on, Chart.js will handle the scaling for retina displays.

Вариант №12. Создайте html- и css- файлы, результат которых показан на рисунке.

The Path to Enlightenment

*These instructions are for *nix platforms. We also have Windows instructions .*

You can run the tests by calling the path_to_enlightenment.rb file.

In your terminal, while in the ruby_koans directory, type:

```
[ ruby_koans ] $ ruby path_to_enlightenment.rb
```

Red, Green, Refactor

In test-driven development (TDD) the mantra has always been red: write a failing test and run it, green: make the test pass, and refactor: look at the code and see if you can make it any better.

Вариант №13.Создайте html- и css- файлы, результат которых показан на рисунке.

ДЕТСКАЯ ЛИТЕРАТУРА

Сказки, рассказы, басни,
современная и классическая проза

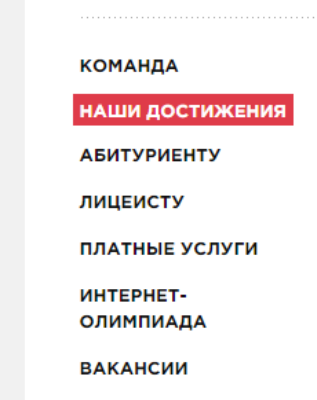
ВСЯЧИНА

Блокноты, календари, сувенирные издания

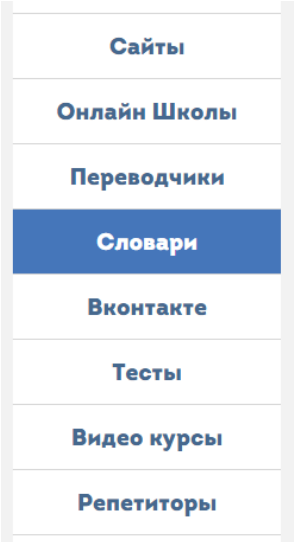
ВКУСНОСТИ

Книги о еде, кулинарные энциклопедии, книги для гурманов и ценителей хорошей кухни

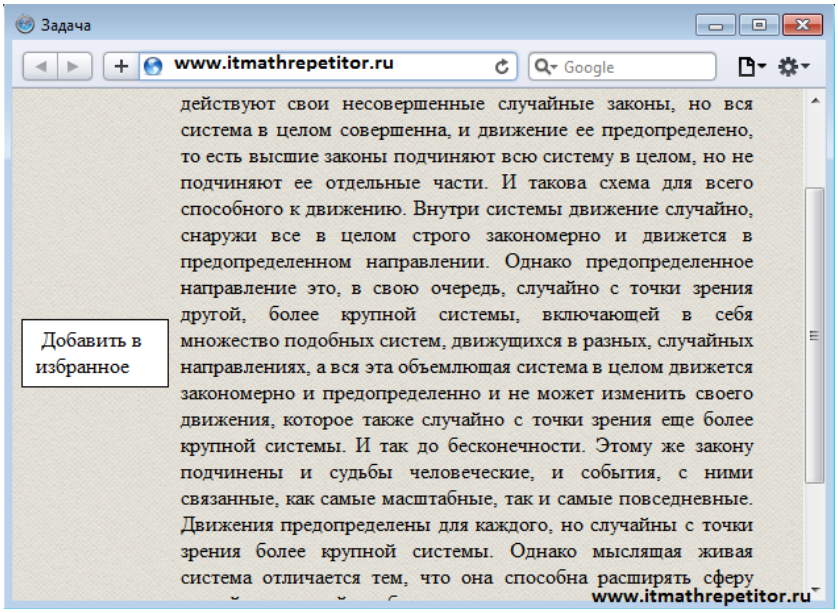
Вариант №14. Создайте html- и css- файлы, результат которых показан на рисунке.



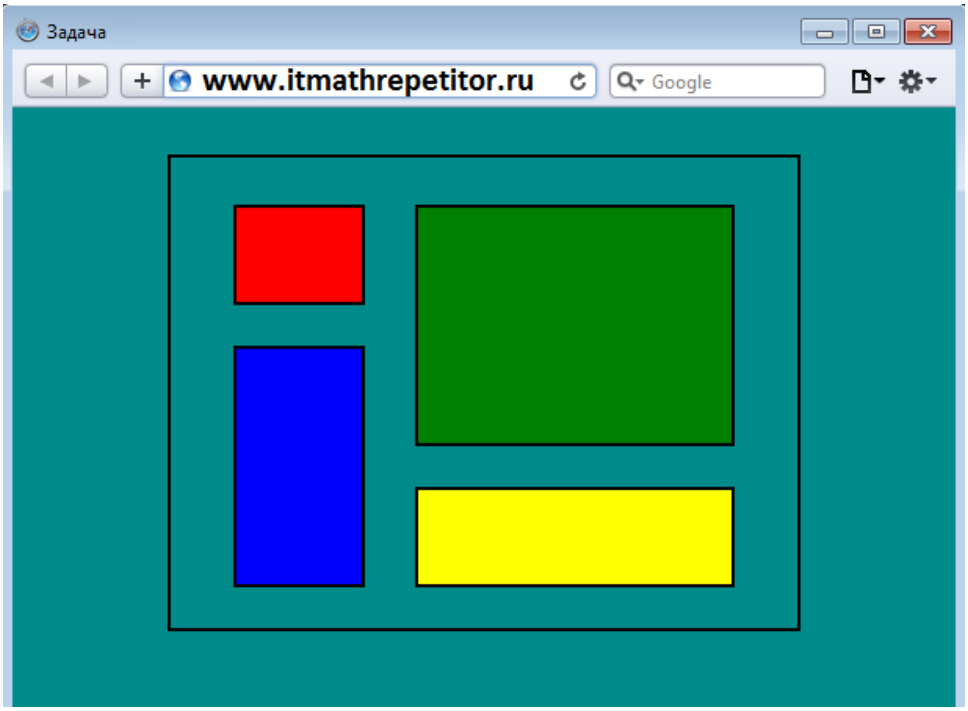
Вариант №15. Создайте html- и css- файлы, результат которых показан на рисунке.



Вариант №16. Создайте html- и css- файлы, результат которых показан на рисунке. Элемент "Добавить в избранное" фиксирован при просмотре страницы.

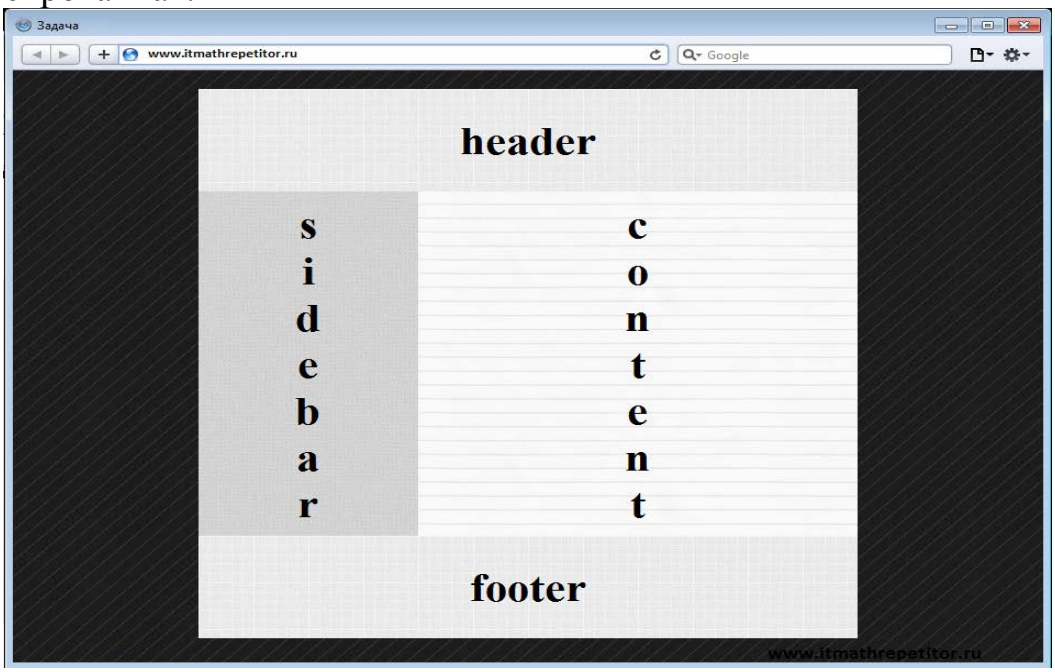


Вариант №17. Создайте html- и css- файлы, результат которых показан на рисунке.

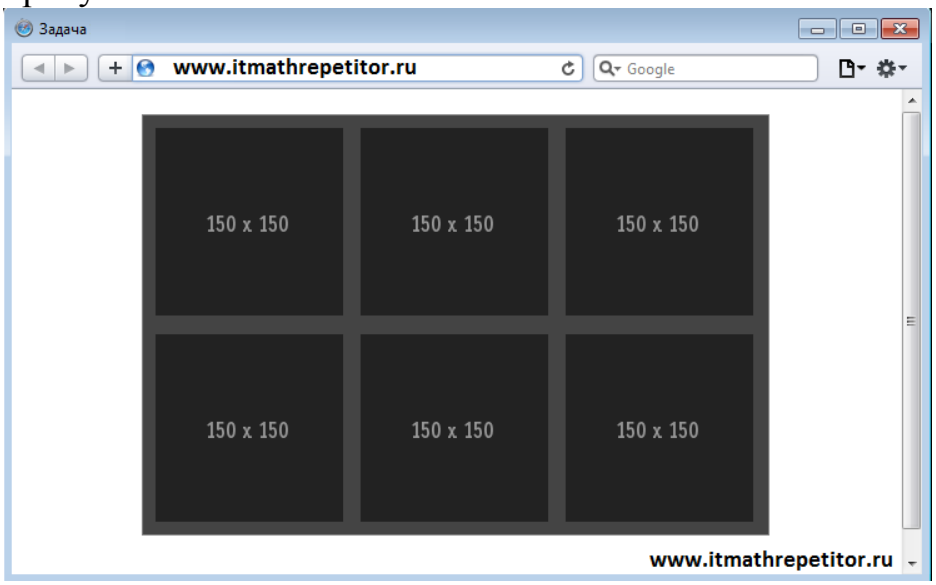


Вариант №18.

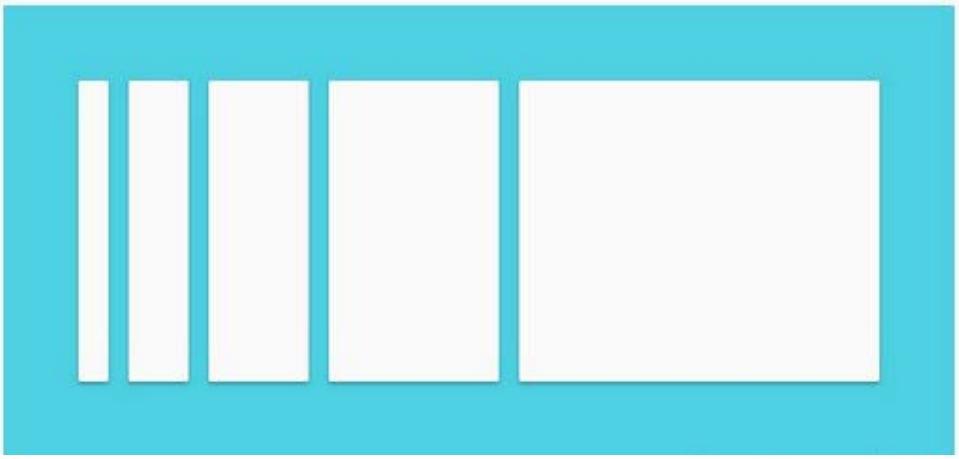
Создайте html- и css- файлы, результат которых показан на рисунке. Изображения для фона: ссылка1, ссылка2, ссылка3, ссылка4. Ширина фиксированная.



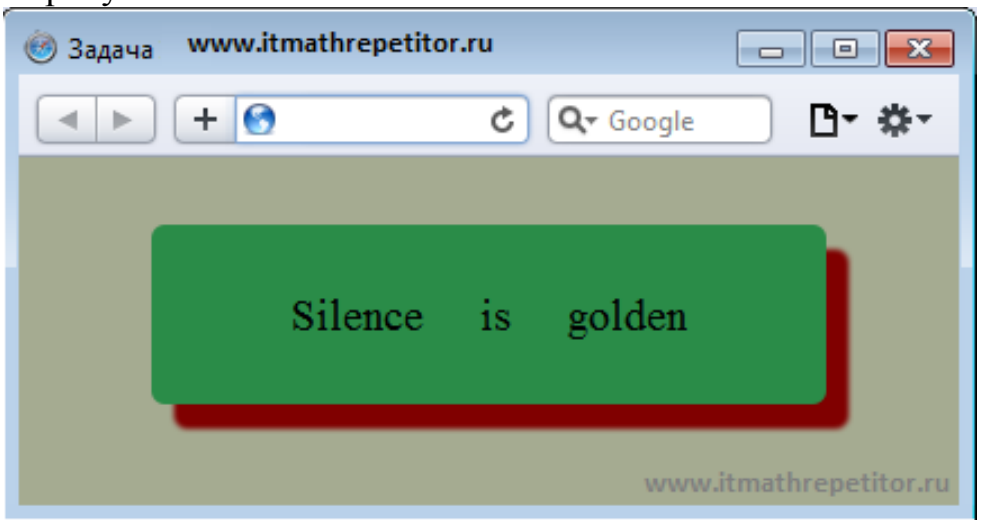
Вариант №19. Условие: Создайте html- и css- файлы, результат которых показан на рисунке.



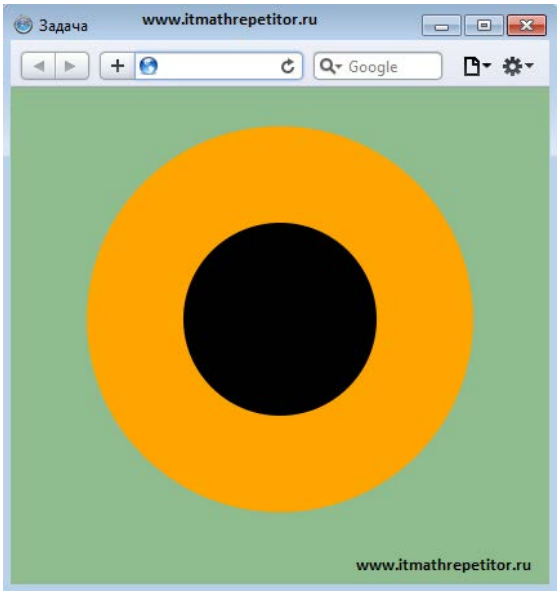
Вариант №20.Создайте html- и css- файлы, результат которых показан на рисунке. Ширина блоков динамически меняется в зависимости от ширины страницы.



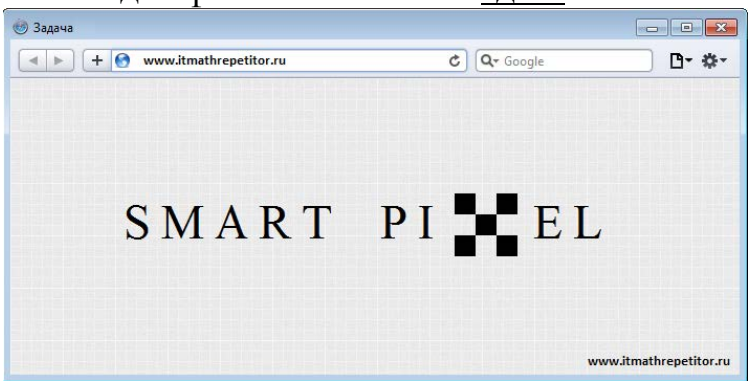
Вариант №21. Условие: Создайте html- и css- файлы, результат которых показан на рисунке.



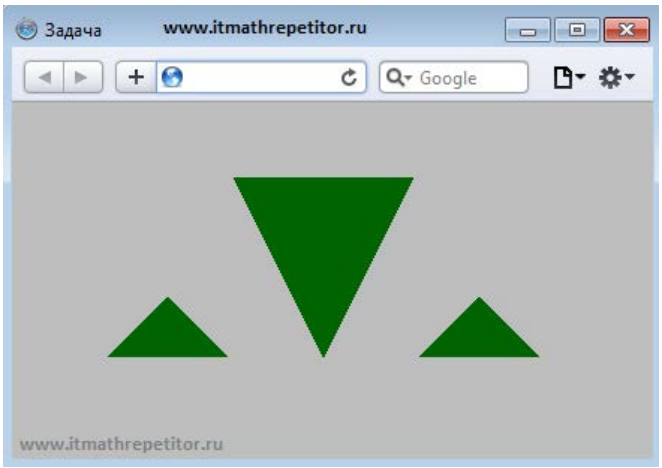
Вариант №22. Создайте html- и css- файлы, результат которых показан на рисунке.



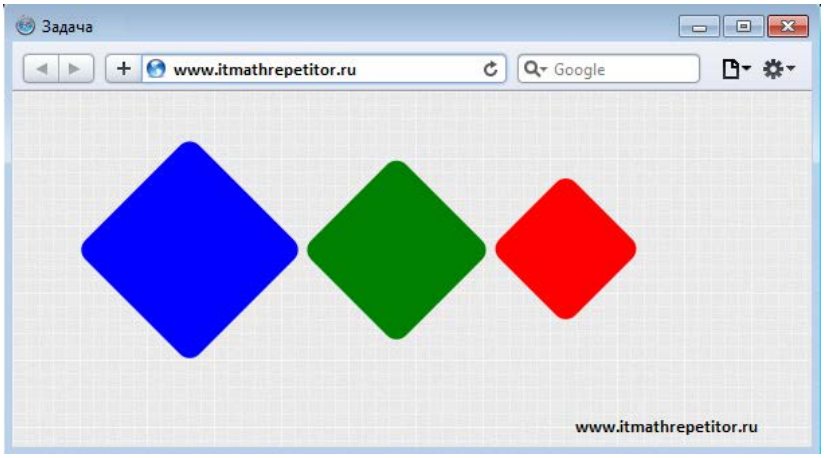
Вариант №23. Создайте html- и css- файлы, результат которых показан на рисунке. Изображение для фона можно взять здесь



Вариант №24.Создайте html- и css- файлы, результат которых показан на рисунке.

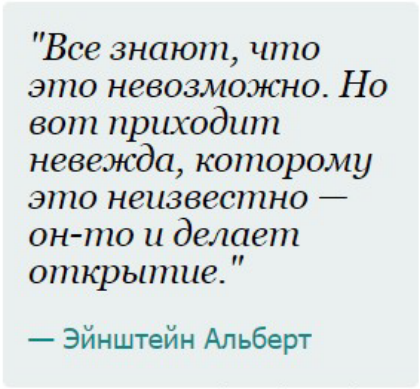


Вариант №25. Создайте html- и css- файлы, результат которых показан на рисунке.



Вариант №26. Создайте html- и css- файлы, результат которых показан на рисунке.

Вариант №27. Создайте html- и css- файлы, результат которых показан на рисунке.



Вариант №28. Создайте html- и css- файлы, результат которых показан на рисунке.

CSS

```
.something-semantic {
    display: table;
    width: 100%;
}
.something-else-semantic {
    display: table-cell;
    text-align: center;
    vertical-align: middle;
}
```

Вариант №29. Создайте html- и css- файлы, результат которых показан на рисунке. Надписи "LIST-ENGLISH" и "СПИСОК РЕСУРСОВ ДЛЯ ИЗУЧЕНИЯ АНГЛИЙСКОГО ЯЗЫКА" оформлены как текст и являются ссылками.

Рекомендуемая литература

1. Ломов А.Ю., HTML, CSS, Скрипты - практика создания сайтов. – Санкт-Петербург «БХВ-Петербург» 2006. - ISBN: 5-94157-698-6.
2. *Эд Титтел, Джефф Ноубл.* HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.
3. *Стивен Шафер.* HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2011. — 656 с. — ISBN 978-5-8459-1676-1.
4. *Энди Бадд, Камерон Молл, Саймон Коллизон.* CSS: профессиональное применение Web-стандартов = CSS Mastery: Advanced Web Standards Solutions. — М.: «Вильямс», 2008. — 272 с. — ISBN 978-5-8459-1199-5.
5. *Кристофер Шмитт.* CSS. Рецепты программирования = CSS. Cookbook. — СПб.: БХВ-Петербург, 2007. — 592 с. — ISBN 978-5-9775-0075-3.
6. *Эрик А. Мейер.* CSS-каскадные таблицы стилей: подробное руководство = Cascading Style Sheets: The definitive Guide. — М.: Символ, 2006. — 576 с. — ISBN 5-93286-075-8.

Лабораторная работа №5

Создание веб страниц используя блоков с помощью CSS

1. Цель работы

Исследование и изучение каскадных таблиц стилей, создание структур веб страниц используя блоков и классов CSS.

2. Краткие сведения из теории

В HTML для позиционирования элементов на странице мы использовали таблицы. У таблиц есть как преимущества (легкость использования, одинаковое отображение браузерами), так и недостатки (объемный, нечитабельный код, нелогичность верстки и т.д.).

В CSS для позиционирования элементов используются блоки (div-ы). Код при этом становится компактным, логичным и легко изменяемым. К недостаткам блочной верстки можно отнести неодинаковую поддержку браузерами, поэтому приходится писать кроссбраузерный код (т.е. код, который отображается разными браузерами почти одинаково).

Элемент является блочным элементом и предназначен для выделения фрагмента документа с целью изменения вида содержимого. Как правило, вид блока управляется с помощью стилей. Чтобы не описывать каждый раз стиль внутри тега, можно выделить стиль во внешнюю таблицу стилей, а для тега добавить атрибут class или id с именем селектора.

Как и при использовании других блочных элементов, содержимое тега всегда начинается с новой строки. После него также добавляется перенос строки.

Синтаксис

...

Атрибуты

align Задает выравнивание содержимого тега .

title Добавляет всплывающую подсказку к содержимому.

Закрывающий тег Обязателен.

Предположим, у нас есть вот такая стандартная html-страница:

Если визуально разделить нашу страницу на прямоугольные блоки, то мы получим четыре блока: шапка сайта, меню, контент и низ сайта. Таким образом, мы имеем четыре div-а. Давайте напишем html-код страницы с четырьмя div-ами и каждому дадим соответствующий идентификатор (id):

```
<html>
<head>
<title>CSS позиционирование</title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
<div id="header">шапкасайта</div>
<div id="menu">меню</div>
<div id="content">контент</div>
<div id="footer">низсайта</div>
</body>
</html>
```

Теперь, на странице style.css зададим те свойства, которые уже знаем, а именно ширину, высоту и фон каждого блока:

```
#header{
background:darkred;
width:715px;
height:100px;
}
#menu{
background:oldlace;
width:190px;
height:300px;
}
#content{
background:oldlace;
width:525px;
height:300px;
}
#footer{
background:darkred;
width:715px;
height:30px;
}
```

Сейчас наша страница в браузере (в уменьшенном варианте) выглядит так:

Такое позиционирование элементов называется *позиционированием в нормальном потоке*. Это значит, что все элементы отображаются в окне браузера сверху вниз, по вертикали, в том порядке, в каком они следуют друг за другом в html-коде.

По своей сути нормальный поток ничем не отличается от позиционирования элементов в HTML. И для верстки такой страницы без CSS, нам пришлось бы использовать таблицу, за неимением других вариантов. В CSS же нам предоставляются и другие схемы позиционирования:

- абсолютное позиционирование
- относительное позиционирование
- плавающая блоковая модель

Для определения схемы позиционирования используется свойство *position*, оно может принимать четыре значения, соответствующие выбранной схеме позиционирования:

- *static* - блок позиционируется в нормальном потоке. Это значение по умолчанию.
- *relative* - относительное позиционирование (относительно нормального потока).
- *absolute* - абсолютное позиционирование
- *fixed* - фиксированное позиционирование (фиксируется относительно области просмотра).

Сегодня мы рассмотрим *абсолютное позиционирование*, остальные схемы будем рассматривать в следующих уроках.

Абсолютное позиционирование

При этой схеме позиционирования расположение блока на странице не зависит от того, в каком месте html-кода расположен этот блок. Расположение каждого блока задается указанием, в каком месте экрана отобразить данный блок. Для этого существуют четыре свойства:

- *left* - указывает на сколько надо сместить блок относительно левого края окна.
- *right* - указывает на сколько надо сместить блок относительно правого края окна.
- *top* - указывает на сколько надо сместить блок относительно верхнего края окна.
- *bottom* - указывает на сколько надо сместить блок относительно нижнего края окна.

Вернемся к нашему примеру. Наши блоки *header*, *menu* и *footer* позиционируются в нормальном потоке, поэтому свойство *position* для них задавать не надо.

А вот блок *content* нужно расположить в другом месте, поэтому для него мы укажем свойство *position:absolute* и зададим смещение: от левого края окна на ширину блока *menu*, т.е. на 190 пикселов, а от верхнего края окна на высоту блока *header*, т.е. на 100 пикселов.

```
#header{
background:darkred;
width:715px;
height:100px;
}
#menu{
background:oldlace;
width:190px;
height:300px;
}
#content{
background:oldlace;
width:525px;
height:300px;
position:absolute;
left:190px;
top:100px;
}
#footer{
background:darkred;
width:715px;
height:30px;
}
```

Теперь наша страница в браузере выглядит так:

Наш блок расположился не совсем так, как ожидалось. Это от того, что мы не учли один нюанс: у браузеров есть свои, встроенные таблицы стилей. И, если мы не задали какое-либо свойство, то используется свойство по умолчанию.

Так, по умолчанию для элемента *body* определены поля, а мы их не учитывали при задании свойств смещения. Чтобы решить эту проблему, достаточно задать для *body* свойство *margin:0px*, т.е. явно указать размер полей (в нашем примере - их отсутствие). Добавим это в таблицу стилей:

```
body{
margin:0px;
}
```

Теперь наша страница выглядит так, как мы и ожидали:

В принципе размеры смещения можно было задать и для каждого блока, иногда это необходимо. Главное, что необходимо запомнить: при абсолютном позиционировании следует задать для блока свойство position:absolute и свойства смещения относительно "родительского" элемента. В нашем примере родительским элементом для div-ов было окно браузера, но может быть и по-другому.

3. Порядок выполнения работы

Разделите окно веб-страницы используя блоков.

Выполнить задание которые даны внизу по вариантам.

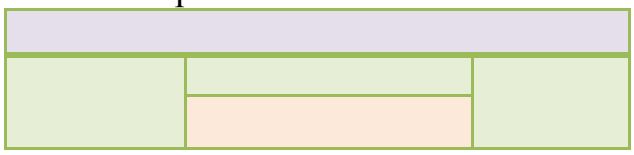
Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

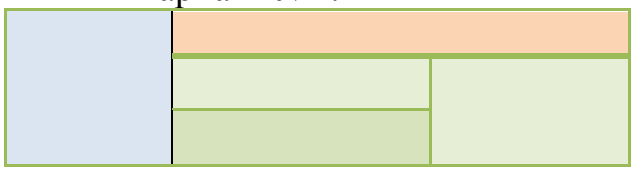
4. Варианты для выполнение задачи

Разделите окно веб-страницы используя блоков

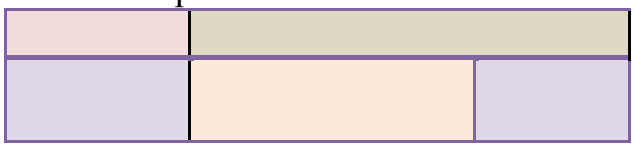
Вариант №1.



Вариант №2.

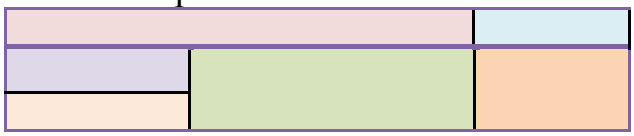


Вариант №3.

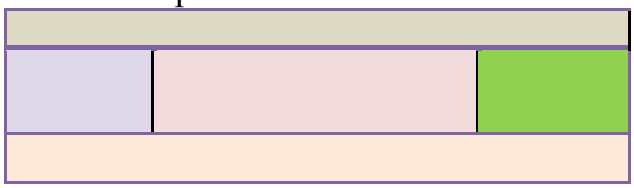


Вариант №4.

Вариант №5.



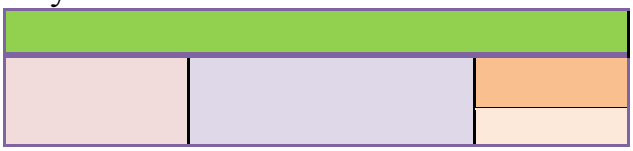
Вариант №6.



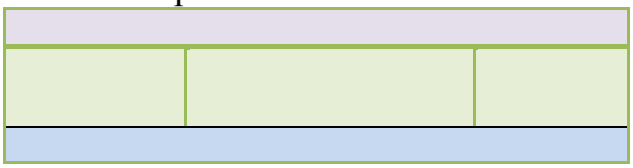
Вариант №7.

Вариант №8.

Вариант №9.



Вариант №10.



Вариант №11.

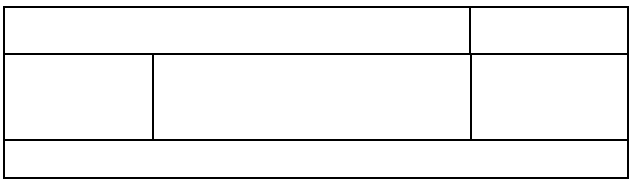
Вариант №12.

Вариант №13.

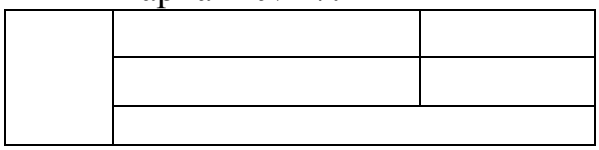
Вариант №14.

Вариант №15.

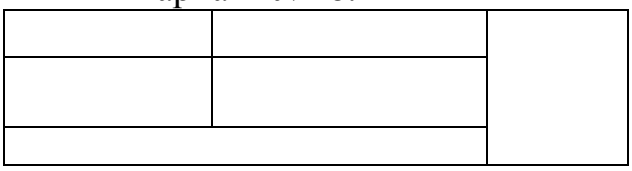
Вариант №16.



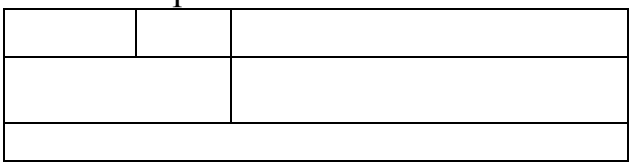
Вариант №17.



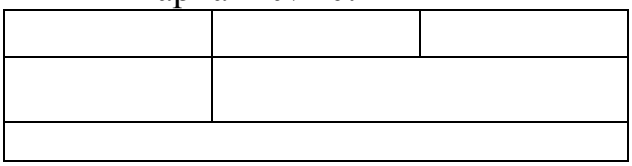
Вариант №18.



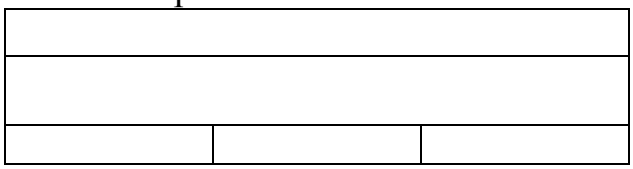
Вариант №19.



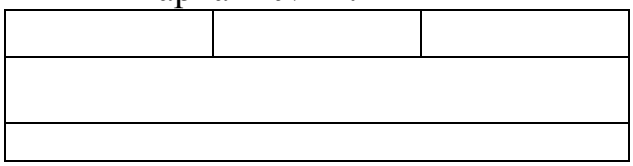
Вариант №20.



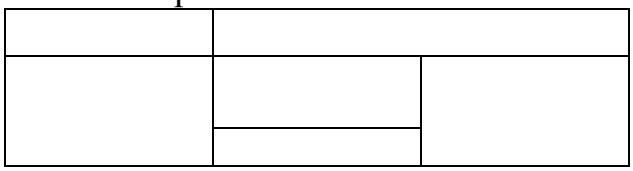
Вариант №21.



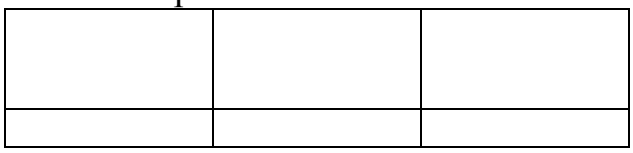
Вариант №22.



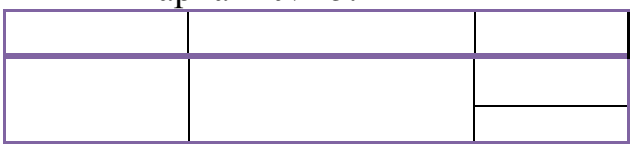
Вариант №23.



Вариант №24.



Вариант №25.



Рекомендуемая литература

1. Ломов А.Ю., HTML, CSS, Скрипты - практика создания сайтов. – Санкт-Петербург «БХВ-Петербург» 2006. - ISBN: 5-94157-698-6.
2. *Эд Титтел, Джефф Ноубл.* HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.
3. *Стивен Шафер.* HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2011. — 656 с. — ISBN 978-5-8459-1676-1.
4. *Энди Бадд, Камерон Молл, Саймон Коллизон.* CSS: профессиональное применение Web-стандартов = CSS Mastery: Advanced Web Standards Solutions. — М.: «Вильямс», 2008. — 272 с. — ISBN 978-5-8459-1199-5.
5. *Кристофер Шмитт.* CSS. Рецепты программирования = CSS. Cookbook. — СПб.: БХВ-Петербург, 2007. — 592 с. — ISBN 978-5-9775-0075-3.
6. *Эрик А. Мейер.* CSS-каскадные таблицы стилей: подробное руководство = Cascading Style Sheets: The definitive Guide. — М.: Символ, 2006. — 576 с. — ISBN 5-93286-075-8.

Лабораторная работа №6

Дополнительные возможности CSS3

1. Цель работы

Изучение каскадных таблиц стилей версии 3, создание дополнительных анимационных и спец-эффектов на CSS3.

2. Краткие сведения из теории

CSS3 (англ. *CascadingStyleSheets 3* — **каскадные таблицы стилей третьего поколения**) — активно разрабатываемая спецификация CSS. Представляет собой формальный язык, реализованный с помощью языка разметки. Самая масштабная редакция по сравнению с CSS1, CSS2 и CSS2.1. Главной особенностью CSS3 является возможность создавать анимированные элементы без использования JS, поддержка линейных и радиальных градиентов, теней, сглаживания и многое другое.

Преимущественно используется как средство описания и оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

Разрабатываемая версия (список всех модулей).

В отличие от предыдущих версий спецификация разбита на модули, разработка и развитие которых идёт независимо.

CSS3 основан на CSS2.1, дополняет существующие свойства и значения и добавляет новые.

Нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией (анимацией) и, возможно, введением переменных

Спецификация CSS3 не является частью спецификации HTML5. Эти два стандарта были разработаны отдельно друг от друга, разными людьми, работающими в разное время в различных местах. Но даже организация W3C призывает веб-разработчиков использовать HTML5 и CSS3 вместе, как часть одной новой волны современного веб-дизайна.

Внедрять CSS3 в веб-сайт можно, по большому счету, используя три стратегии:

Стратегия 1: используйте то, что можно

Логично использовать возможности с высоким уровнем поддержки на всех основных браузерах. В качестве примера одной из таких возможностей можно назвать применение веб-шрифтов. Используя шрифты правильного формата, эту функциональность можно заставить работать на таких старых браузерах, как Internet Explorer 6. К сожалению, очень немногие возможности CSS3 входят в эту категорию. Почти все иные возможности не будут работать на все еще популярных браузерах IE 7 и IE 8.

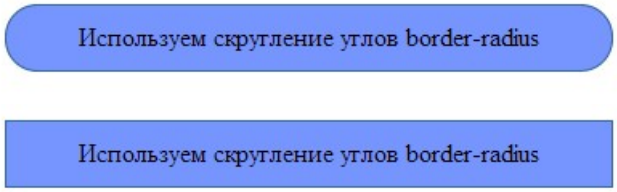
Стратегия 2: рассматривайте возможности CSS3 как усовершенствования

У фанатов CSS3 есть боевой клич: "Вебсайты не должны выглядеть абсолютно одинаково на всех браузерах".

Идея в основе этой стратегии заключается в использовании CSS3 для тонкой доработки страниц, причем эта доработка не повлияет на возможность просмотра основного содержимого и форматирования страницы в менее способных браузерах. Одним из примеров такой тонкой настройки является свойство border-radius, позволяющее скруглять углы рамок. Далее приводится пример указания этого свойства в правиле таблицы стилей:

```
header {
    background-color:#7695FE;
    border:       thin #336699 solid;
    padding: 10px;
    margin: 10px;
    text-align: center;
    border-radius: 25px;
}
```

Браузеры, поддерживающие свойство border-radius, будут использовать его, а старые браузеры просто игнорировать его, оставляя углы рамок квадратными:



Эта стратегия явно привлекательна, так она позволяет веб-дизайнерам манипулировать последними "игрушками" этой технологии. Но если слишком увлечься, она имеет и определенный недостаток. Несмотря на то, насколько хорошо веб-страница может выглядеть при просмотре в последней версии вашего любимого браузера запустив ее в одном из старых браузеров, которые используются значительной частью ваших посетителей, вы можете быть глубоко разочарованы ее намного менее впечатляющим внешним видом. А ведь вы хотите, чтобы ваш веб-сайт производил впечатление на всех, а не только на веб-фанатов, использующих лучшие браузеры.

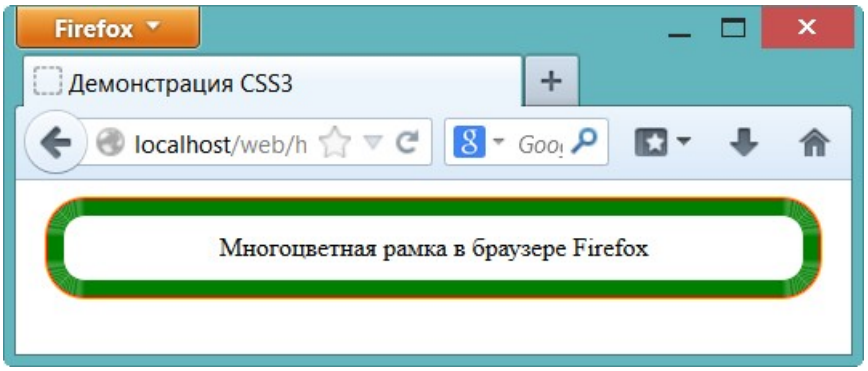
Использование частично поддерживаемой возможности CSS3 хорошая идея, если веб-сайт будет достойно выглядеть и без нее. Но иногда без этой возможности легко потерять важную часть дизайна своего веб-сайта, или же сайт может выглядеть просто неприглядно. Рассмотрим, например, что случится, если использовать многоцветную рамку, поддерживаемую только в браузере Firefox:

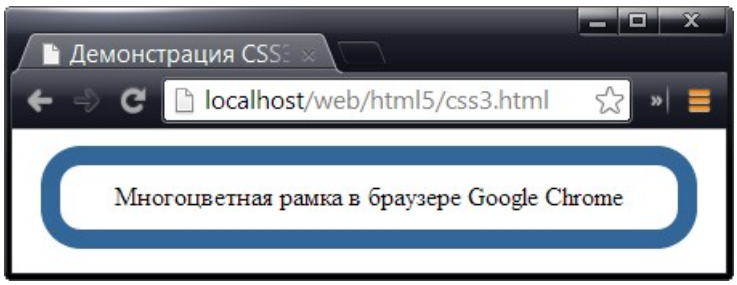
```
-moz-border-bottom-colors: orange red green;

-moz-border-left-colors:  orange red green;

-moz-border-right-colors:  orange red green;
```

-moz-border-top-colors: orange red green;

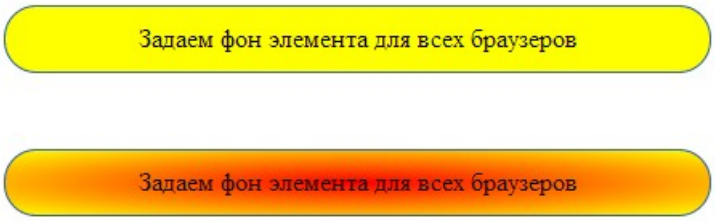




Иногда эту проблему можно решить, установив несколько свойств в правильном порядке. Здесь базовым методом будет установка сначала общих свойств, а за ними новых, которые замещают предыдущие свойства. Когда этот подход работает, он удовлетворяет все браузеры — старые браузеры используют стандартные настройки, в то время как новые браузеры замещают эти настройки новыми. Далее показан пример применения этого метода для замены обычного фона градиентным:

```
header {
    background:yellow;
    background: radial-gradient(ellipse, red, yellow);
}
```

Результаты применения этого правила показаны на рисунке



Вверху: браузеры, которые не понимают CSS3, используют первую часть правила для header и окрашивают фон сплошным желтым цветом. Внизу: браузеры, которые понимают CSS3, используют вторую часть правила и заполняют фон радиальным градиентом.

3. Порядок выполнения работы

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

Рекомендуемая литература

1. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.
2. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2011. — 656 с. — ISBN 978-5-8459-1676-1.
3. Энди Бадд, Камерон Молл, Саймон Коллизон. CSS: профессиональное применение Web-стандартов = CSS Mastery: Advanced Web Standards Solutions. — М.: «Вильямс», 2008. — 272 с. — ISBN 978-5-8459-1199-5.
4. Кристофер Шмитт. CSS. Рецепты программирования = CSS. Cookbook. — СПб.: БХВ-Петербург, 2007. — 592 с. — ISBN 978-5-9775-0075-3.
5. Эрик А. Мейер. CSS-каскадные таблицы стилей: подробное руководство = Cascading Style Sheets: The definitive Guide. — М.: Символ, 2006. — 576 с. — ISBN 5-93286-075-8.

Лабораторная работа №7

Язык программирование JavaScript

1. Цель работы

Исследование и изучение мультипарадигменного язык программированияJavaScript.

2. Краткие сведения из теории

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный,императивныйифункциональныйстили.Являетсяреализацией языкаECMAScript(стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

Название «JavaScript» является зарегистрированным товарным знаком компании Oracle Corporation.

JavaScript разработан с использованием простой объектно-ориентированной парадигмы. Объект - это конструкция со свойствами, которые являются переменными JavaScript или другими объектами. Объект также может иметь ассоциированные с ним функции, которые известны как методы объекта. В дополнение к имеющимся в Navigator клиентским и серверным объектам Вы можете определять свои собственные объекты.

Объект в JavaScript имеет ассоциированные с ним свойства. Вы получаете доступ к свойству очень просто:

Имя_объекта.Имя_свойства

И имя объекта, и имя свойства чувствительны к регистру. Вы определяете свойство, присваивая ему значение. Например, предположим, имеется объект myCar (начиная отсюда, примем для удобства, что объект уже существует).

В JavaScript 1.0 Вы можете ссылаться на свойства объектов по имени свойства или по порядковому индексу. В JavaScript 1.1 и позднее, однако, если Вы первоначально определили свойство по имени, Вы всегда обязаны будете

обращаться к нему по имени, и, если Вы первоначально определили свойство по индексу, Вы всегда обязаны будете обращаться к нему по его индексу.

Это применимо при создании объекта и его свойств с помощью конструктора функции, как в примере с типом объекта Car, и если Вы определяете отдельные свойства явным образом (например, myCar.color = "red"). Так, если Вы определили свойства объекта по индексу, как, например, myCar[5] = "25 mpg", Вы можете затем обращаться к этому свойству только как myCar[5].

Исключением из этого правила являются объекты, отражённые из HTML, такие как массивы форм. Вы всегда обращаетесь к этим объектам по порядковому номеру (зависящим от местонахождения объекта в документе) или по их именам (если они определены). Например, если второй тэг <FORM> в документе имеет в атрибуте NAME значение "myForm", Вы можете ссылаться на форму document.forms[1], или document.forms["myForm"] или document.myForm.

Вы можете добавлять свойство к ранее определённому типу объекта через использование свойства prototype. Так определяется свойство, которое совместно используется всеми объектами специфицированного типа, а не только одним данным экземпляром этого объекта. Следующий код добавляет свойство color ко всем объекта типа car, а затем присваивает значение свойству color объекта car1.

```
Car.prototype.color=null;
car1.color="black";
```

Специальные типы данных

В JavaScript существуют два специальных типа данных: null и undefined. С первого взгляда они похожи, однако семантика их применения существенно различается.

Переменные этого типа данных могут иметь единственное значение null. Такие переменные можно охарактеризовать как не содержащие данных. То есть они не содержат действительных данных любых других типов.

Изменить тип данных переменной на null можно, присвоив ей специальное литеральное значение null:

```
var x = null;
```

Такое действие используют для удаления данных, хранимых в переменной.

Для переменных данного типа оператор typeof возвращает строку "Object", что может вводить в заблуждение. Для проверки значения переменной на эквивалентность null следует использовать операции равенства либо строгого равенства.

JavaScript поддерживает три скалярных типа данных: логический, числовой и строковый.

Переменные этого типа могут иметь два значения ("истина" или "ложь"), определяемых, соответственно, литералами true и false. Часто эти значения являются результатами вычисления выражений и могут быть либо сохранены как значения переменных, либо непосредственно использованы в операторах

условия, цикла и т. д. Оператор typeof для переменных данного типа возвращает строку "boolean".

При конвертировании в boolean следующие значения считаются равными false:

- null;
- undefined;
- числовые 0 или NaN;
- строковые, если длина строки равна 0.

Все остальные значения конвертируются в boolean как true.

Переменные числового типа в JavaScript могут содержать как целые, так и действительные значения. Все числа представляются в формате IEEE 754 и могут находиться в диапазоне от -2^{53} до 2^{53} в виде целых значений либо в диапазонах от $-1{,}7976931348623157\ 10^{308}$ до $-5\ 10^{-324}$ и от $5\ 10^{-324}$ до $1{,}7976931348623157\ 10^{308}$ в виде значений двойной точности с плавающей точкой. Оператор typeof для переменных числового типа возвращает строку "number".

Как и в других языках, строки в JavaScript предназначены для хранения данных в виде последовательности символов. Реализации JavaScript современных браузеров поддерживают строки, которые могут содержать как символы из набора ASCII, так и двухбайтовые Unicode-символы.

Массивы в JavaScript представлены в виде объектов, имеющих свойства упорядоченных карт. При помощи карты производится отображение ключей в значения. Ключи массива могут быть представлены значениями разных типов (не только целым числом). На основе массивов в JavaScript достаточно просто можно реализовать различные, даже весьма сложные структуры данных: словарь, стек, дерево. Объекты массивов могут создаваться путем присвоения переменным литеральных значений массивов либо при помощи оператора new.

Литерально массив определяется перечислением значений в квадратных скобках []. При этом значения имеют целочисленный, последовательно возрастающий от нуля индекс. Тип данных значений может быть любым. Значения могут быть представлены переменными, константами, литералами, в том числе допускается использовать литералы массивов, объектов, функций. Допустимо не указывать некоторые значения (в этом случае их тип данных будет undefined). Пример литерального определения массива:

```
vararr = [1, , ,'текст', [1, 2, 3],
function(){ alert('функция')}, 0,
100, { prop0: 10, prop1: 2 }, 1000];
```

Объект массива имеет три варианта конструктора, поэтому существуют следующие способы создания массива при помощи оператора new:

```
var arr = new Array();
var arr = new Array(<размермассива>);
var arr = new Array(<значение0>, <значение1>, ...,
<значениеN>);
```

В первом случае создается пустой массив. Во втором — массив с числом элементов, равным указанным размером (тип всех элементов undefined). Третий

способ очень похож на определение массива с помощью литерала — аналогично создается массив, заполненный указанными значениями.

Адресация элементов массивов выполняется при помощи все той же нотации квадратных скобок. Причем, поскольку массивы реализованы как карты, то при попытке модификации элемента с несуществующим индексом происходит добавление пары "ключ-значение" в массив:

```
var arr = [10, 20, 30];
var A = arr[1]; // теперь переменная A
// содержит значение 20
arr['color'] = 'green'; // в массив добавлено значение 'green'
A = arr['color']; // теперь переменная A
// содержит значение 'green'
```

Элемент массива может быть удален при помощи оператора delete. При этом удаляется пара "ключ-значение", т. е. оператор typeof для удаленного элемента вернет undefined.

3. Порядок выполнения работы

Выполнить задание по варианту.

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

Рекомендуемаялитература

1. Chuck Musciano and Bill Kennedy, "HTML: The Definitive Guide", O"Reily&Associates, Inc (1996).
2. html.doc, "Время Microsoft", журналАурамедиа, спецвыпуск. "Решения Microsoft", выпуск 5 (1996).
3. Michael J. Hannah, "HTML Reference Manual" (1996), http://www.sandia.gov/ sci_compute/ html_ref.html
4. HTML 3.2. Features at a Glance, http://www.w3.org/pub/WWW/MarkUp/Wilbur/features.html
5. Netscape extensions to HTML 3.0, http://home.netscape.com/assist/net_sites/html_extensions_3.html
6. HTML 2.0 Standart, http://www.w3.org /pub /WWW/MarkUp/html-spec
7. Using JavaScript in HTML, http://home.netscape.com/eng/mozilla/2.0/handbook/javascript/index.html
8. Stefan Koch, "Introduction to JavaScript" (1996), http://www.webcom.com/java/java-script/intro/index.htm

Лабораторная работа №8

Использование библиотеки jQuery

1. Цель работы

Исследование установки на веб страницу библиотеки jQuery и использование возможностей этой библиотеки.

2. Краткие сведения из теории

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. Сейчас разработка jQuery ведётся командой jQuery во главе с Джоном Резигом.

История создания

HTML был одной из первых вещей, которую Джон Резиг освоил, когда он только начал заниматься программированием. Резиг программировал на QBasic, когда один его знакомый показал ему, как создать веб-страницу (используя Angelfire), а также основы HTML. Отец подарил ему на Рождество две книги по HTML. Именно тогда, когда он только начал программировать на Visual Basic, HTML и веб-дизайн очень заинтересовали его.

Но страсть к JavaScript пришла значительно позже, примерно в 2004 году. Тогда Резиг получал степень в области компьютерных наук и работал на полставки в местной фирме Brand Logic. Он занимался дизайном сайта, в котором создавался пользовательский скроллинг. Джон был разочарован и расстроен, особенно потому, что использовал код других разработчиков, после чего решил серьёзно изучить JavaScript. Изучив, пришёл к выводам, что JavaScript — это простой, но изящный язык, который является невероятно мощным для решения многих задач. В течение следующей пары лет Джон создал множество различных JavaScript-приложений, прежде чем закончить создание jQuery. Основной целью создания jQuery Резиг видел возможность закодировать многоразовые куски кода, которые позволят упростить JavaScript и использовать их так, чтобы не беспокоиться о кросс-браузерных вопросах. Библиотека была представлена общественности на компьютерной конференции «BarCamp» в Нью-Йорке в 2006 году.

Возможности

- Движок кросс-браузерных 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.

Использование

jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл:

```
<head>
<script src="jquery-2.2.2.min.js">
</script>
</head>
```

Вся работа с jQuery ведётся с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним — jQuery. Второй способ считается более правильным, а чтобы код не получался слишком громоздким, можно писать его следующим образом:

```
jQuery(function($) {
// здесь код скрипта, где в $ будет находиться объект, предоставляющий
доступ к функциям jQuery
})
```

Работу с jQuery можно разделить на 2 типа:

- Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса.
- Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора CSS, что возвращает

объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery. Например,
$("div.test").add("p.quote").addClass("blue").slideDown("slow");
находит все элементы div с классом test, а также все элементы p с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз. Здесь методы add, addClass и slideDown возвращают ссылку на исходный объект $("div.test"), поэтому возможно вести такую цепочку.
Методы, начинающиеся с $., удобно применять для обработки глобальных объектов. Например:

```
$.each([1,2,3], function() {
  document.write(this + 1);
});
```

добавит на страницу 234.
$.ajax и соответствующие функции позволяют использовать методы AJAX. Например:

```
$.ajax({
  type: "POST",
  url: "some.php",
  data: {name: 'John', location: 'Boston'},
  success: function(msg){
    alert( "Data Saved: " + msg );
}
});
```

В этом примере идет обращение к скрипту some.php с параметрами name=John&location=Boston, и полученный результат выдается в сообщении посредством alert().
Пример добавления к элементу обработчика события click с помощью jQuery:

```
$("a").click(function() {
    alert("Hello world!");
});
```

В данном случае при нажатии на элемент <a> происходит вызов alert("Hello world!").

Интеграция с другими продуктами

28 сентября 2008 года в официальном блоге jQuery[5] 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.

3. Порядок выполнения работы

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

Рекомендуемая литература

1. *Адам Фримен.* jQuery для профессионалов = Pro jQuery. — М.: «Вильямс», 2012. — 960 с. — ISBN 978-5-8459-1799-7.
2. *Джейсон Ленгсторф.* PHP и jQuery для профессионалов = Pro PHP and jQuery. — М.: «Вильямс», 2010. — С. 352. — ISBN 978-5-8459-1693-8.
3. *Самков Г.* jQuery. Сборник рецептов. — СПб.: БХВ-Петербург, 2010. — С. 416. — ISBN 978-5-9775-0495-9.

Лабораторная работа №9

Событие в jQuery

1. Цель работы

Исследование и изучение библиотеки jQuery,событие в JavaScript и jQuery,

2. Краткие сведения из теории

jQuery делает тривиальным добавление простых эффектов на страницу. Эффекты могут использовать встроенные настройки или подгонять продолжительность анимации индивидуально. Вы также можете создавать собственную анимацию произвольных свойств CSS.

Часто используемые эффекты встроены в jQuery как методы, которые вы можете вызвать для любого объекта jQuery:

- .show() — показать выбранные элементы;
- .hide() — скрыть выбранные элементы;
- .fadeIn() — анимация прозрачности выбранных элементов до 0%;
- .fadeOut() — анимация прозрачности выбранных элементов до 100%;
- .slideDown() — отображение выбранных элементов с помощью вертикального скользящего движения;
- .slideUp() — сокрытие выбранные элементы с помощью вертикального скользящего движения;
- .slideToggle() — показать или скрыть выбранные элементы с вертикальным скользящим движением в зависимости от того, видны элементы в данный момент или нет.

После создания выборки мы просто применяем к ней эффект.

```
$( '.hidden' ).show();
```

Вы также можете указать длительность встроенных эффектов. Есть два способа сделать это: можете задать время в миллисекундах

```
$( '.hidden' ).show( 300 );
```

или использовать одну из предустановленных скоростей:

```
$( '.hidden' ).show( 'slow' );
```

Предустановленные скорости указаны в объекте jQuery.fx.speeds. Вы можете изменить этот объект, чтобы переопределить значения по умолчанию или расширить его новыми именами:

```
// Переустановить имеющуюся предустановленную скорость
jQuery.fx.speeds.fast = 50;
 // Создать новую предустановленную скорость
jQuery.fx.speeds.turtle = 3000;

// Поскольку мы переустановили скорость 'fast', то теперь анимация
```

```
// длится 50 миллисекунд
$( '.hidden' ).hide( 'fast' );

// После их создания мы можем использовать пользовательские скорости
// подобновстроенным
$( '.other-hidden' ).show( 'turtle' );
```

Часто вам требуется сделать что-то после завершения анимации — если вы попытаетесь сделать это до окончания анимации, то это может повлиять на качество анимации или привести к удалению элементов, которые являются частью анимации. Вы можете предоставить функцию обратного вызова для методов анимации, если желаете указать, что должно произойти после завершения эффект. Внутри этой функции this указывает на исходный элемент DOM, к которому применялся эффект. Подобно событиям мы можем превратить его в объект jQuery через $(this).

```
$( 'p.old' ).fadeOut( 300, function() {
  $( this ).remove();
});
```

Обратите внимание, что если выборка не содержат каких-либо элементов, то ваша функция обратного вызова никогда не будет выполнена! Если вам требуется выполнить функцию независимо от того, есть элементы в выборке или нет, то вы можете создать функцию и использовать её так:

```
var oldElements = $( 'p.old' );
var thingToAnimate = $( '#nonexistent' );

// Эта функция будет «обратным вызовом» для метода show,
// когда элементы будут показаны. В противном случае мы просто
// вызовемеёнезамедлительно
var removeOldElements = function() {
  oldElements.remove();
};

if ( thingToAnimate.length ) {

// Когда передаётся в качестве функции обратного вызова для show,
  // эта функция будет вызвана после завершения анимации
thingToAnimate.show( 'slow', removeOldElements );

} else {
removeOldElements();
}
```

Произвольные эффекты с .animate( )

Если встроенные анимации не подходят, вы можете использовать .animate() для создания произвольной анимации большинства свойств CSS. Учтите, что вы не можете анимировать свойство color, но есть плагин, который делает его возможным.

У метода .animate( ) есть три аргумента:

- объект, определяющий свойства для анимации;
- продолжительность анимации в миллисекундах;
- функция обратного вызова, которая будет вызываться после окончания анимации.

Метод .animate( ) может анимировать до указанного конечного значения или увеличить существующее значение.

```
$( '.funtimes' ).animate({
   left: '+=50', // увеличитьна 50
   opacity: 0.25,
   fontSize: '12px'
  },
  300,
  function() {
   // выполняется, когдаанимациязавершена
}
);
```

Метод animate() принимает набор свойств элемента, которые затем изменяются, за счет чего достигается анимация.

Данный метод имеет следующую форму использования: animate(properties [,duration] [,easing] [,complete])

Обязательный параметр properties содержит набор css-свойств, у которых указываются финальные значения.

Параметр duration указывает, как долго будет длиться изменение прозрачности элемента. По умолчанию его значение равно 400 миллисекунд. Также можно использовать значения 'slow' и 'fast', которые соответствуют длительности эффекта в 600 и 200 миллисекунд.

Параметр easing принимает название функции плавности анимации в виде строки. По умолчанию его значение равно "swing".

Параметр complete представляет функцию обратного вызова, вызываемую методом по завершении анимации.

Используя анимацию, важно иметь в виду, что в данном случае мы можем использовать только те свойства css, которые принимают числовые значения, например, ширина и высота. Другие же свойства, как, например, цвета фона или шрифта, мы уже так просто не сможем использовать.

Применим простую анимацию к изображению, изменив ряд его свойств:

```
1    <img src="ars.jpg" id="ars" /><br>
```

```
2    <button id="anim">Анимация</button>
3    <script type="text/javascript">
4    $(function() {
5       $('#anim').click(function(){
6           $('#ars').animate({ opacity: 0.25,
7                'margin-left': '50',
8                height: '200'});
9       });
10   });
11   </script>
```

В данном случае у нас изменяют свое значение три свойства: opacity, margin-left и height. Сама анимация представляет переход от начальных значений к значениям, указанным для свойств в методе animate.

В предыдущем примере у нас жестко кодируются финальные значения свойств. Так, левый отступ после анимации у нас будет иметь значение 50. Но что если мы хотим, чтобы движение постоянно продолжалось при нажатии на кнопку? В этом случае мы можем использовать относительные значения:

```
1    $('#ars').animate({
2            'margin-left': '+=50',
3            width: '-=10',
4            height: '-=10'}, 1000);
5       });
```

Кроме указания относительных значений здесь также использовано время анимации - 1000 миллисекунд.

Чтобы выполнить более детальную настройку анимации мы можем использовать еще одну форму метода animate: animate(properties, options)
Здесь используется новый параметр - options. В этом параметре мы можем указать ряд конфигурационных параметров, которые будут использоваться при анимации. Этот параметр принимает следующие опции:

- duration: продолжительность анимации. По умолчанию равна 400 миллисекунд
- easing: название функции плавности анимации. По умолчанию значение 'swing'
- queue: булевое значение, указывающее, нужно ли поместить анимацию в очередь эффектов. По умолчанию имеет значение true, что значит, что анимация помещается в очередь. Если же присвоить значение false, то анимация будет выполняться немедленно.
- specialEasing: объект javascript, который сопоставляет анимируемые свойства с функциями плавности
- step: функция, вызываемая для каждого анимируемого свойства каждого участвующего в анимации элемента
- progress: функция, вызываемая на каждом этапе анимации по одному разу для каждого элемента вне зависимости от количества анимируемых свойств

- complete: функция вызываемая после завершения анимации
- done: функция, вызываемая при завершении анимации
- fail: функция, вызываемая при ошибке в процессе анимации, если анимация не сможет завершится нормальным путем
- always: функция, вызываемая после завершения анимации вне зависимости, завершится анимация обычным путем или с ошибкой

Конечно, все опции разом необязательно определять и можно остановиться лишь на нескольких. Например, применим ряд из этих опций:

```
1    $('#ars').animate({
2          'margin-left': '+=50',
3          width: '-=10',
4          height: '-=10'
5          },{
6          duration: 1000,
7          step: function(now, fx) {
8             var data = fx.elem.id + ' ' + fx.prop + ': ' + now;
9             $('body').append('<div>' + data + '</div>');
10         },
11         complete: function() {
12            alert('Анимациязавершена');
13         }
14   });
```

Здесь мы использовали три опции: продолжительность анимации и функции по шаговой обработки и завершения.

Весь процесс анимации разбивается на ряд мелких этапов, которые выполняются в течение определенного времени (в данном случае 1000 миллисекунд). На каждом этапе и вызывается функция пошаговой обработки (step: function(now, fx)), причем для каждого анимируемого элемента анимируемого свойства. Она принимает два параметра: now и fx. Now показывает текущее значение на данном этапе анимируемого свойства, а fx содержит данные о анимируемом объекте. Как в данном случае, мы получаем анимируемое свойство (fx.prop) и анимируемый элемент (fx.elem). И затем мы эти данные можем использовать по своему усмотрению.

К базовым эффектам в jQuery относятся эффекты скрытия и отображения элементов, которые достигаются с помощью методов show(),hide() и toggle().

Так, например, мы можем скрывать и отображать элементы по клику по кнопке:

```
1    <ul>
2       <li>Java</li>
3       <li>C/C++</li>
4       <li>JavaScript</li>
5    </ul>
6    <button id="show">Показать</button>
7    <button id="hide">Скрыть</button>
8    <script type="text/javascript">
9    $(function() {
```

```
10      $('button#show').click(function(){
11          $('ul').show();
12      });
13      $('button#hide').click(function(){
14          $('ul').hide();
15      });
16  });
17  </script>
```

Таким образом, при нажатии на кнопку "Показать", будет срабатывать метод $('ul').show(), а при нажатии на кнопку "Скрыть", список будет скрыт с помощью метода $('ul').hide().

Тот же самый эффект мы могли бы сделать, использовав в обоих случаях метод toggle, который просто переключает видимость:

```
1   $(function() {
2       $('button').click(function(){
3           $('ul').toggle();
4       });
5   });
```

Теперь разберем эти методы подробнее.

3. Порядок выполнения работы

Содержание отчета

1. Название лабораторной работы.
2. Цель лабораторной работы.
3. Необходимые принадлежности.
4. Краткие сведения из теории.
5. Решение примера по варианту.
6. Вывод по работе.

Рекомендуемая литература

1. *Робин Никсон.* Создаем динамические веб-сайты с помощью PHP, MySQL, javascript и CSS (2-е издание). — O'Reilly Media, Inc: Питер, 2013. — С. 560.
2. *Сэм Руби, Дэйв Томас, Дэвид Хэнссон.* Rails 4. Гибкая разработка веб-приложений. - СПб.: Питер, 2014. — 448 с.
3. *Ташков П. А.* Веб-мастеринг на 100 %: HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка - ООО Издательство «Питер», 2010 - 263 с. ISBN 978-5-49807-826-7