УДК 004.738.5:37

Р.И.Допира

Карагандинский государственный университет им. Е.А.Букетова (E-mail:ritadopira@mail.ru)

Разработка стиля Web-приложения

При разработке Web-приложения большое внимание уделяется дизайну и интерфейсу страниц. Автором приведены атрибуты тегов и события JavaScript, используемые для создания эффектов. В статье описано поэтапное создание стиля.

Ключевые слова: web-программирование, CSS, JavaScript, jQuery.

Перед началом создания Web-приложения стоит придумать макет или шаблон. Например, при создании электронного Web-ресурса сайт и электронный учебник схожи по структуре и интерфейсу, во многих Web-студях рисуют макеты в графическом редакторе. Этим занимается Web-дизайнер, однако, несмотря на то, что он только рисует, он ещё обязан знать систему построения Web-ресурсов, разбираться в языке, чтобы не помешать и не запутать в дальнейшем Web-программиста при «конвертировании» рисунка в *html*-страницу, а именно соблюдать юзабилити, или удобство использования интерфейса.

Самый популярный и самый простой макет Web-приложения — это макет, состоящий из заголовка, расположенного вверху, навигационного меню, которое может располагаться как слева, так и справа обтекающего его содержания, а также расположенных внизу копирайтов.

Другой вид макета схож с первым, но содержание можно разбить на большее количество блоков, за счёт этого удобнее читать, так как содержание расположено по центру. Добавочный блок в основном служит для вторичного меню, краткого описания открытой темы или уточнения автора. Не исключено и использование автором собственного макета.

Одним из способов организации макета *Web*-приложения является блочная конструкция. Возможно расставлять *div*-блоки хаотично, вне зависимости от окружающего элемента. Данный вариант макета не может обойтись без стилей. Сложности возникают, когда автор не распределяет стили в каскадной таблице по мере заполнения *html*-страницы, а раскидывает вразброс, отсюда усложняется редактирование. Если создаются стили под каждый элемент *Web*-приложения, то размер файла со стилями увеличивается, и легкий вес *div*-блочной конструкции теряет свою актуальность. При прописывании стилей в самих тегах будет много ошибок при валидации кода.

При создании блока (будь-то табличный или *div*-блок) по умолчанию текст прилегает к краям. Для его выравнивания или передвижения в пределах блока используются методы: выставление атрибутов внутри тега; задание стиля для блочного элемента.

Для каждого тега существуют свои атрибуты, не более пятнадцати. Также атрибут одного тега может встречаться в атрибутах другого тега, из-за чего часто бывают ошибки с выставлением атрибута одного тега другому. Рассмотрим самые основные атрибуты для тегов двух структурных видов. Для табличного «td» используются атрибуты:

– *abbr*, который задает краткое описание содержимого ячейки;
– *align*, который определяет выравнивание содержимого ячейки по горизонтали;
– *bgcolor*, который задает цвет фона ячейки;
– *char*, который выравнивает содержимое ячейки по заданному символу;
– *charoff*, который смещает содержимое ячейки относительно заданного символа;
– *colspan*, который объединяет горизонтальные ячейки;
– *nowrap*, который запрещает перенос строк;
– *valign*, который задает выравнивание содержимого ячейки по вертикали;
– *dir*, который задает направление и отображение текста — слева направо или справа налево.

Для тега «th» перечисленные атрибуты соответственные. Для блочного «div» внутри используется только атрибут «align» и тот, который записывается чаще всего в стилевой файл каскадной таблицы стилей.

Как для табличного, так и для блочного тегов используются одни и те же свойства стилей:

– *position* — задает позицию как объекта внутри блока, так и самого блока в документе и может также зафиксировать его положение, игнорируя прокрутку;

– *visibility* — это свойство предназначено для отображения или скрытия элемента блока, включая рамку вокруг него и фон. При скрытии элемента, хотя он и становится не виден, место, которое элемент занимает, остается за ним;
– *vertical-align* — свойство выравнивает элемент по вертикали относительно своего родителя, окружающего текста или ячейки таблицы;
– *text-align* — это свойство выравнивает элемент по горизонтали, также относительно своего родителя;
– *text-decoration* — это свойство добавляет оформление текста в виде его подчеркивания, перечеркивания, линии над текстом и мигания. Одновременно можно применить более одного стиля, перечисляя значения через пробел. Является альтернативным способом форматирования текста;
– *text-indent* — это свойство устанавливает величину отступа первой строки блока текста. Воздействия на все остальные строки он не оказывает и является альтернативой тегу «p» форматирования текста;
– *text-overflow* — это свойство определяет параметры видимости текста в блоке, если текст целиком не помещается в заданную область;
– *text-shadow* — это свойство добавляет тень к тексту, а также устанавливает её параметры: цвет тени, смещение относительно надписи и радиус размытия;
– *padding* — это свойство устанавливает значение полей вокруг содержимого элемента. Полем называется расстояние от внутреннего края рамки элемента до воображаемого прямоугольника, ограничивающего его содержимое;
– *margin* — это свойство устанавливает величину отступа от каждого края элемента. Отступом является пространство от границы текущего элемента до внутренней границы его родительского элемента;
– *opacity* — это свойство определяет уровень прозрачности элемента *html*-страницы. При частичной или полной прозрачности через элемент проступает фоновый рисунок или другие элементы, расположенные ниже полупрозрачного объекта;
– *list-style* — это универсальное свойство, позволяющее одновременно задать стиль маркера, его положение, а также изображение, которое будет использоваться в качестве маркера;
– *font* — это универсальное свойство, которое позволяет одновременно задать несколько характеристик шрифта и текста. Надо отметить, что такие свойства, как *font* и ему подобные, являющиеся свойствами для форматирования текста, имеют свои подсвойства для точности. В основное свойство можно прописать сразу все значения, относящиеся к его подсвойствам через пробел;
– *color* — это свойство определяет цвет текста внутри блока. Может задаваться RGB-значения, HEX-значениями, а также обычными названиями цвета, к примеру: *red, yellow, blue* и т.д. [1].

Во время разработки дизайна вместе со стилями используется *JavaScript* для создания эффектов некоторым элементам *html*-страницы. JavaScript — это прототипно-ориентированный сценарный язык программирования, но используемое в языке прототипирование обусловливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Огромное количество Web-страниц сделано с использованием сценариев, и сайты без них кажутся блеклыми и скучными. Конечно, нет смысла оспаривать важность текстового содержания для любой *html*-страницы, однако использование JavaScript не только улучшит подачу материала, но и сделает страницу более запоминающейся [2].

Рассмотрим стилистику *html*-страницы с использованием jQuery. jQuery — это JavaScript-библиотека, фокусирующаяся на взаимодействии JavaScript, HTML и CSS.

Основные функции данной библиотеки:
– умеет обращаться к любому элементу DOM (объектной модели документа) и не только обращаться, но и манипулировать ими;
– умеет работать с событиями;
– легко осуществляются различные визуальные эффекты;

– умеет работать с AJAX (очень полезная технология, позволяющая общаться с сервером без перезагрузки страницы);
– имеет огромное количество JavaScript-плагинов, предназначенных для создания элементов пользовательских интерфейсов [3].

Для использования данной библиотеки нужно ее прежде скачать бесплатно с сайта разработчика и подключить к Web-приложению, а именно встроить код в страницу, где используется эта библиотека. Для этого в *html* существует тег <*script*>, который и отвечает за подключение внешних файлов и скриптов.

Достоинства данной библиотеки:

– *кроссбраузерность*. Это основное преимущество JavaScript, так как у него много разных синтаксисов. Чего стоят хотя бы способы работы с Ajax. Во всех браузерах работа эта организована по-разному. С jQuery все единообразно; использовать его становится очень просто, достаточно написать одну строку кода. Многие задачи, которые решаются на JavaScript, небольшими функциями на jQuery решаются одной строкой. Особенно если мы говорим о визуальных эффектах;
– *производительность*. Конечно, существует множество других фреймворков, которые делают примерно то же самое, но jQuery — самый быстрый из них;
– *общедоступность и распространение*. На данный момент jQuery используют Яндекс и Google. И скачать ее можно с их серверов;
– наличие не большого, а просто огромного количества плагинов для jQuery. Одних только фотогалерей больше ста, диалоговые окна, экранные лупы, виртуальные клавиатуры, тултипы, балуны и многое другое [4].

Нельзя забывать, что для создания динамичности JavaScript связан с CSS (каскадная таблица стилей) и сырой код скрипта не может быть вписан в *html*-страницу, он попусту не будет работать, так как он пишется непосредственно для стиля, после чего стиль придает тегу эффект. Чаще они задаются не через «class», а через «id».

Создаем *html*-страницу с подключенной библиотекой jQuery, собственный скрипт и файл со стилями (рис. 1). Вписываем небольшой макет с открывающимися блоками при нажатии определенной кнопки. Всего блоков три. Как показано на рисунке, для каждой кнопки мы пишем событие *onclick*, это событие языка JavaScript, которое прописывается внутри тега. Как для тега *button*, так и для других тегов, выражающих динамику страницы и переход по гиперссылкам, имеется множество внутритеговых JavaScript событий. Вот основные из них:

– *onblur* — потеря фокуса;
– *onchange* — изменение значения элемента формы;
– *onclick* — щелчок левой кнопкой мыши на элементе;
– *ondblclick* — двойной щелчок левой кнопкой мыши на элементе;
– *onfocus* — получение фокуса;
– *onload* — документ загружен;
– *onmousedown* — нажата левая кнопка мыши;
– *onmousemove* — перемещение курсора мыши;
– *onmouseout* — курсор покидает элемент;
– *onmouseover* — курсор наводится на элемент;
– *onmouseup* — левая кнопка мыши отпущена;
– *onreset* — форма очищена;
– *onselect* — выделен текст в поле формы;
– *onsubmit* — форма отправлена;
– *onunload* — закрытие окна [2, 3].

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Эффекты с использованием библиотеки jQuery</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <link rel="stylesheet" type="text/css" href="style.css">
    <script type="text/javascript" src="jquery-1.2.6.js"></script>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    <p><h3><b>Эффекты jQuery</b></h3></p>
    <div id="ef1">1</div>
    <div id="ef2">2</div>
    <div id="ef3">3</div>
    <button onclick="addEffect1()">Эффект Show()</button>
    <button onclick="addEffect2()">Эффект SlideDown()</button>
    <button onclick="addEffect3()">Эффект Animate()</button>
  </body>
</html>
```

Рисунок 1. Подключение библиотеки jQuery

Далее создаем файл со стилями (рис. 2) и задаем свойства для них и с удобными для нас значениями для отображения.

```
@charset "utf-8";
/* CSS Document */
#ef1, #ef2, #ef3 {
    width:120px; /*ширина*/
    height:100px; /*высота*/
    float:left; /*обтекание*/
    background:#aaa; /*фон*/
    margin:15px; padding:30px; /*внешние и внутренние отступы*/
    color:#000; /*цвет текста*/
    display:none; /* целостность блока (видимость)*/
    border:3px solid #123123; border-radius:5px;
    -webkit-border-radius:5px; -moz-border-radius:5px; /*стиль краев блока*/
    text-shadow:1px 1px 2px white; /*тень текста*/
    box-shadow:0 0 5px black; -webkit-box-shadow:0 0 5px black;
    -moz-box-shadow:0 0 5px black; /*тень блока*/
}
/* Стили прописаны */
```

Рисунок 2. Вид стилевого файла

После создания файла со стилями мы приступаем непосредственно к созданию JS-файла с необходимыми для работы функциями JavaScript (рис. 3).

```
// JavaScript Document
function addEffect1(){
  $("#ef1:hidden").show();
}
function addEffect2(){
  $("#ef2:hidden").slideDown("slow");
}
function addEffect3(){
  $("#ef3:hidden").show().animate( {
                fontSize:"36px"
                } , 3000 );
}
```

Рисунок 3. Вид Javascript-файла

Конечный результат невозможно показать на картинке, так как он имеет динамичность. В итоге получилось три блока с тремя разными эффектами при нажатии на соответствующую кнопку. Рассмотрим подробней функции в JS-файле:

– функция addEffect1() видит $ (знак доллара) и понимает, что это jQuery, затем она видит ("#ef1:hidden") и понимает, что ей нужно найти элемент с id="ef1", обладающий свойством

hidden (невидимый). Далее она видит .show() и понимает, что найденный элемент надо сделать видимым;

– функция addEffect2() видит $ (знак доллара) и понимает, что это jQuery, затем она видит ("#ef2:hidden") и понимает, что ей нужно найти элемент с id="ef2", обладающий свойством hidden (невидимый). Далее она видит .slideDown("slow") и понимает, что найденный элемент надо медленно ("slow") отобразить сверху-вниз (slideDown);
– функция addEffect3() видит $ (знак доллара) и понимает, что это jQuery, затем она видит ("#ef3:hidden") и понимает, что ей нужно найти элемент с id="ef3", обладающий свойством hidden (невидимый). Далее она видит .show() и понимает, что найденный элемент надо сделать видимым. Затем она видит .animate({fontSize:"36px"} , 3000) и понимает, что размер шрифта нужно за 3 секунды (3000) увеличить до 36 пикселей (fontSize:"36px").

Загрузка Web-приложения осуществляется с индексной страницы. Это страница, с которой пользователь может получить быстрый доступ к темам со страницы, которая даст возможность представить Web-приложение, а также какие темы в него включены.

Для оригинальности, а также для более стильного и современного представления разработана индексная страница в стиле Modern. Этот стиль интерфейса, вышедший недавно на обзор всему миру и который рекламировали везде и всегда вместе с операционной системой Windows 8. Ранее этот стиль интерфейса назывался Metro. Данный стиль очень актуален, он обладает красочным интерфейсом.

В наше время широко распространены планшетные персональные компьютеры, имеющие разные операционные системы, аппаратную часть и программное обеспечение, несколько отличающееся от ПО настольного компьютера. Все большее число людей читают книги именно на планшетных ПК. Учитывая это, Web-приложение разрабатывается как кроссплатформенное, обеспечивающее легкость и удобство чтения. Данная возможность стиля является главным его достоинством, выделяющим его из многообразия.

Структура как индексной страницы, так и остальных страниц табличная. При визуальной оценке она будет выглядеть из трех строк с вложенными таблицами, однако множество блоков-таблиц будут независимыми от целой таблицы, так как методом их распределения является позиционирование. В первой строке основная рассматриваемая тема, вторая и основная строка являются центром внимания, здесь будут располагаться Modern-блоки с названиями тем и их кратким описанием, появляющимся при наведении курсора.

Блоки с темами — основа стиля Modern, поэтому все визуальные эффекты будут применены к ним. Так как Web-приложение имеет шесть тем, то количество блоков соответствует данному числу. Распределение же блоков выставлено по центру. JavaScript-эффект при наведении плавный, с тридцатипроцентной прозрачностью и тенью (рис. 4). Краткая информация темы появляется также при наведении курсора, к тому же стоит отметить, что выравнивание появляющейся краткой информации обратно пропорционально выравниванию названия темы, а также расположению блока. Исключением являются средние блоки, параметры выравнивания текста в них относительны расположению блока. Данный эффект выполняется при помощи свойства *visibility* с указанием значения при неактивности — *hidden*, при наведении — *visible*.

Создается стилевой файл под названием *modernstyle.css*, в котором будут располагаться только параметры стилей, прописанные в индексной странице. Для создания фона (во второй основной строке) прописывается *div*-блок с параметрами ширины блока (100 %), фонового цвета (в том случае если по какой-либо причине прописанный в блок фон перестанет отображаться), позиционирования (фиксированная позиция с выравниванием к левому верхнему краю) и наложения слоя (для корректного отображения значение слоя -1, так как располагается в самом начале кода и по умолчанию значение слоя равно нулю). Сам же фон расположен между тегами *div*. Остальная конструкция состоит, как уже описывалось, из трех строк, но лишь визуально, на самом же деле Modern-блоки расположены по центру страницы, они не зависят от других таблиц, так как применен уже описанный ранее фиксированным позиционированием. В итоге получается, что на самом деле строк две (верхняя и нижняя), нижняя, которая также располагается благодаря фиксированному позиционированию с привязкой к нижнему краю экрана. Столь сложная и неудобная на первый взгляд структура страницы создана для кроссбраузерности, т.е. для максимально корректной работы во всех браузерах разных платформ, так как основными достоинствами позиционирования являются его наличие в версиях CSS 2.0, 2.1 и 3, поддержка всеми браузерами без дополнительных приставок.

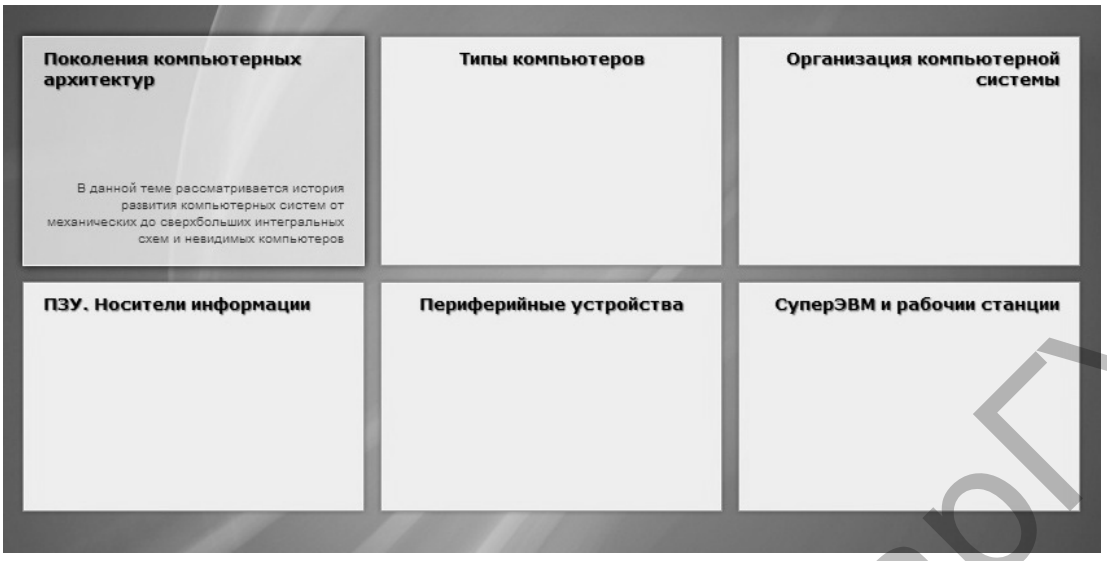

Рисунок 4. Блочная конструкция индексной страницы

Несмотря на столь плавный градиентный переход от одного цвета к другому, эффект появления текста при наведении достигается обычными стилями, без JavaScript. Скрытие текста при неактивности ссылки (рис. 5) достигается выставлением значения *hidden* (скрытый) в свойстве *visibility*. Это свойство отвечает за видимость элемента, т.е. всего блока. Несмотря на это сам блок виден.

```
/* Эффекты для блоков метро */
a.button div.view-block-metro, .button div.view-block-metro, a.button div.view-block-metro1, .button div.view-block-metro1,
a.button div.view-block-metro2, .button div.view-block-metro2 {
    visibility:hidden;
}
```

Рисунок 5. Скрытие текста при помощи свойств CSS

Хитрость заключается в том, что текст находится во вложенном блоке, поэтому скрыт целый блок с текстом, и чтобы не было видно блока при наведении курсора, его параметры фона и границ заданы не были. За появление блока с текстом в Modern-блоке отвечает значение *visible* (рис. 6).

```
a.button:hover div.name-block-metro div.view-block-metro, .button:hover div.name-block-metro div.view-block-metro {
    visibility: visible;
    font:10px Arial, Helvetica, sans-serif;
    text-align:right; color:#333; text-shadow:none;
    padding-top:70px; display:block;
}
a.button:hover div.view-block-metro1, .button:hover div.view-block-metro1 {
    visibility: visible;
    font:10px Arial, Helvetica, sans-serif;
    text-align:justify; color:#333; text-shadow:none;
    padding-top:85px; display:block;
}
a.button:hover div.view-block-metro2, .button:hover div.view-block-metro2 {
    visibility: visible;
    font:10px Arial, Helvetica, sans-serif;
    text-align:left; color:#333; text-shadow:none;
    padding-top:70px; display:block;
}
```

Рисунок 6. Отображение блоков при наведении курсора с помощью свойств CSS

Тень и граница при активности ссылки достигаются выставлением значений для свойства *box-shadow* (тень элемента/блока/таблицы). С данным свойством работать сложней, так как у него проблемы с кроссбраузерностью, однако эта проблема решается добавлением к свойству вендорного префикса, обозначающего, что данное свойство будет работать только в определенном браузере (рис. 7). Эти префиксы также работают со свойствами *border-radius, background-size, background-clip,*

background-origin, radial-gradient и др. Как видно на рисунке, для корректного отображения данного свойства в разных браузерах были дописаны префиксы *-webkit-, -moz- и -khtml-*. Префикс *-webkit-* дает понять браузеру, что данное свойство будет работать только в браузерах Chrome и Safari, *-moz-* — для браузеров от Mozilla, а префикс *-khtml-* работает лишь в среде KDE UNIX-платформ. Стоит помнить, что свойство *box-shadow* некорректно работает в браузере Internet Explorer, из-за чего приходится применять специальные параметры, распознаваемые лишь этим браузером.

```
a.button, .button {
    -webkit-box-shadow:
        0 0 5px rgba(0, 0, 0, 0.28),
        inset 0 1px 0 rgba(255, 255, 255, 0.45),inset 0px -1px 0px rgba(255, 255, 255, 0.45),
        inset 1px 0px 0px rgba(255, 255, 255, 0.45), inset -1px 1px 0px rgba(255, 255, 255, 0.45);
    -moz-box-shadow:
        0 0 5px rgba(0, 0, 0, 0.28),
        inset 0 1px 0 rgba(255, 255, 255, 0.45), inset 0px -1px 0px rgba(255, 255, 255, 0.45),
        inset 1px 0px 0px rgba(255, 255, 255, 0.45), inset -1px 1px 0px rgba(255, 255, 255, 0.45);
    -khtml-box-shadow:
        0 0 5px rgba(0, 0, 0, 0.28),
        inset 0 1px 0 rgba(255, 255, 255, 0.45), inset 0px -1px 0px rgba(255, 255, 255, 0.45),
        inset 1px 0px 0px rgba(255, 255, 255, 0.45), inset -1px 1px 0px rgba(255, 255, 255, 0.45);
    box-shadow:
        0 0 5px rgba(0, 0, 0, 0.28),
        inset 0 1px 0 rgba(255, 255, 255, 0.45), inset 0px -1px 0px rgba(255, 255, 255, 0.45),
        inset 1px 0px 0px rgba(255, 255, 255, 0.45), inset -1px 1px 0px rgba(255, 255, 255, 0.45);
    filter: progid:DXImageTransform.Microsoft.Glow(color=#cccccc, strength=1);
    -ms-filter: "progid:DXImageTransform.Microsoft.Glow(color=#cccccc, strength=1)";
    -webkit-transition: all 0.4s; transition: all 0.4s;
    /* простые параметры */
    text-decoration: none;  color: #000 !important; text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.55);
    font: bold 16px/16px "klavika-web", "Helvetica Neue", Helvetica, Arial, Geneva, sans-serif;
    background:#eee; float: left; width:232px; height:160px; padding: 8px 15px; border: 1px solid #999;
}
a.button:hover, .button:hover {
    background: url(../fons/modernblockfon.png) no-repeat;
    filter: progid:DXImageTransform.Microsoft.Shadow(Color=#000000, Strength=1, Direction=150);
    -ms-filter: "progid:DXImageTransform.Microsoft.Shadow(Color=#000000, Strength=1, Direction=150)";
}
```

Рисунок 7. Разновидность вендорного префикса для правильного отображения одного и того же элемента в разных браузерах

В стиле используется свойство *filter*, как без, так и с вендорным префиксом, значения которого означают цвет, величину области и направленность теней соответственно тому, как показано на рисунке 7. Изменение фона при наведении также описано на рисунке 7, свойство *background* — с указанием фоновой картинки без повторения.

Независимость Modern-блока от содержимого текста или элемента достигается свойством *display* со значением *block*, что указывает на приоритет выставленных ширины и высоты блока и определяет, как элемент должен быть показан в документе. При выставлении данного свойства блок с шириной в пятьсот пикселей и высотой в двести пикселей будет отображаться точно с такими же значениями ширины и высоты, независимо от того, есть ли внутри элемента текст или вложенный элемент. Главная страница приобрела вид некой программы, в которой нет лишних картинок и кнопок, имеющей особую цветовую гамму, не превышающую использование более 3 основных и 3 дополнительных цветов. Наблюдается некая симметрия макета, отсутствие переполнения, что также является наиболее оптимальным видом для лучшего восприятия. Минимальное использование миниатюрных тематических картинок (иконок) способствует сосредоточенности и легкости использования, так как вместо указательных и тематических картинок используются символы мнемоники, что также облегчает использование страницы браузером, не теряя времени на загрузку дополнительных элементов макета. Для создания описанной структуры макета использовалось минимальное количество Java-скриптов, так как это было бы причиной долгой загрузки страниц, что является недоработанностью и акцентированием внимания автора не на содержании, а на изящности шаблона, что неприемлемо.

Список литературы

1 *Белицкий А.* Полный справочник по CSS и HTML. — Изд-во «Webformyself», 2012. — 452 с.
2 Интернет-портал JavaScript.ru // http://javascript.ru/book
3 *Вагнер Ричард, Вайк Аллен*. JavaScript. Энциклопедия пользователя. Изд-во «ДиаСофт», 2001. — 464 с.
4 *Шевчук Антон.* jQuery для всех // [ЭР]. Режим доступа: http://anton.shevchuk.name/jquery/

Р.И.Допира

Web-қосымшаның стилін құру

Web-қосымшаны құру кезінде парақтың интерфейсіне және дизайнына көп назар бөлінді. Автор эффектілер құру үшін қолданылатын тегтердің атрибуттарын және JavaScript оқиғаларын келтірді. Мақалада стильді құру кезендері кеңінен сипатталды.

R.I.Dopira

Development of style Web-application

When developing Web-based applications much attention is paid to design and interface pages. The author cites the tag attributes and events JavaScript, which were used to create effects. The article describes the steps for creating a style.

References

1 Belitsky A. *The Complete Reference CSS and HTML*, Publisher: «Webformyself», 2012, 452 p.
2 *Internet portal JavaScript.ru* // http://javascript.ru/book
3 Wagner Richard, Allen Vike. *JavaScript. Encyclopedia user*, Publisher: «DiaSoft», 2001, 464 p.
4 Shevchuk Anton. *jQuery for all* // [ER]. Access mode: http://anton.shevchuk.name/jquery/