**Лекция 11: CSS. Работа с контейнерами**

*Дизайн* веб - сайта, в современных условиях веб - разработки, является одним из наиболее важных критериев оценки сайта в целом. Вспомните сколько раз вы уходили с тематического сайта при виде невзрачного оформления и "кривой" *навигации*.

Можно сколько угодно повторять про то, что *по* обложке не судят, но это не поможет увеличить число посещений сайта. Несмотря на то, что настоящий курс не ставит перед собой целью обучение основам дизайна, совсем обойти эту тематику нельзя.

**Понятие контейнерного дизайна**

Если подойти методологически к вопросу о дизайне веб - страниц, то можно выделить следующие группы (школы) дизайна:

* текстовый;
* фреймовый;
* табличный;
* контейнерный.

Четких границ между указанными школами, пожалуй, выделить нельзя. Подходы к дизайну, как и абсолютное большинство веб - технологий, стандартов и правил появлялись и развивались эволюционно.

*Текстовый* подход к дизайну сайтов является первым и простейшим. Суть крайне проста: основным содержанием веб - страницы является структурированный текст, приемы оформления, зачастую, ограничиваются простым форматированием текста, изменением шрифтов, *цветов текста* и т.д.

С учетом того, что к внешнему виду современной веб-страницы выдвигаются *определенные требования* (все мы привыкли к панели *навигации* слева, основному *меню* сайта в "шапке" и т.п.), обойтись только текстовой версткой крайне сложно. Фактически данным подходом в чистом виде никто не пользуется, даже несмотря на то, что время загрузки страниц при этом минимально и верстка проста.

Кроме того, общие элементы для всех веб-страниц сайта не подгружаются, а прописываются для каждой страницы. Т.е. каждый раз при открытии очередной страницы подгружаются все ее элементы, даже в случае, если меняется фактически, *по* сравнению с предыдущей страницей, только содержательная ее часть.

В качестве решения, ликвидирующего недостатки текстового подхода к дизайну сайтов, выступила идея использования *фреймов* – элементов веб - страниц, содержимое которых подгружается извне.

Иначе говоря, *фрейм* отображает содержимое другой веб - страницы, *адрес* которой прописывался в параметрах фрейма. Таким образом, веб - страница состояла из набора *фреймов*, при этом подгружается только требуемое содержимое. Структуру *фреймов*, правда, при этом поменять нельзя.

Как и текстовый *дизайн*, фреймовый в настоящее время практически не применяется.

Идея *табличной* верстки сайтов лежит на поверхности. На веб - странице размещается html - *таблица*, в ячейки которой и помещается *контент*. Соответственно, визуально веб - страницу можно разделить на необходимое количество блоков ("шапка" сайта, панель *навигации* и т.д.).

Табличный *дизайн* является весьма популярным и распространенным, особенно в среде начинающих разработчиков. Основным достоинством подхода является его *понятность* и возможность избежать проблем совместимости. С учетом того, что к каждой ячейке таблицы можно применить собственный *CSS* стиль, можно говорить о том, что подход до конца себя еще не изжил.

*Контейнерный* подход при верстке и дизайне сайтов заключается в использование элементов - контейнеров для размещения *контента*. Контейнеры (парный *тег* <div>) являются *блочными элементами*, к каждому контейнеру может быть применен индивидуальный стиль.

Также *контейнер* может отображать содержимое внешней, *по* отношению к нему веб - страницы, при этом, в отличие от *фреймов*, контейнеры стандартизованы консорциумом *W3C*. *По* сравнению же с табличным подходом, создаваемые страницы "легче" с точки зрения количества кода.

Рассмотрим подробнее приемы работы с контейнерами.

**Контейнеры**

Отметим, что <div> не является единственным элементом - контейнером. Но поскольку, в контексте данной лекции, мы говорим именно о контейнерном дизайне, то и рассматривать будем только его базовый элемент – *тег* <div>. С другими элементами контейнера предлагается ознакомиться самостоятельно.

Парный *тег* <div> используется для группировки элементов веб - страницы, как правило с целью управления внешним видом содержимого, при помощи *CSS* стилей.

**Работа со стилями контейнеров**

Для контейнеров применим тот же набор атрибутов стилей, что и для других элементов веб - страницы. Рассмотрим ключевые, с точки зрения дизайна, атрибуты стиля при работе с <div> контейнерами.

**Управление размерами контейнеров**

<div> – *контейнер* представляет собой прямоугольную область. Значения высоты и ширины данной области определяются такими стандартными атрибутами стилей, как:

* высота
  + *min*-*height* – задает минимальную высоту элемента;
  + *height* – задает высоту элемента;
  + max-*height* – задает максимально возможную высоту элемента;
* ширина
  + *min*-*width* – задает минимальную ширину элемента;
  + *width* – задает ширину элемента;
  + max-*width* – задает максимально возможную ширину элемента.

Каждый из указанных атрибутов стиля может принимать в качестве значений любые единицы длины, поддерживаемые в *CSS*, например:

width: 200px (в пикселях)

width: 2in (в дюймах)

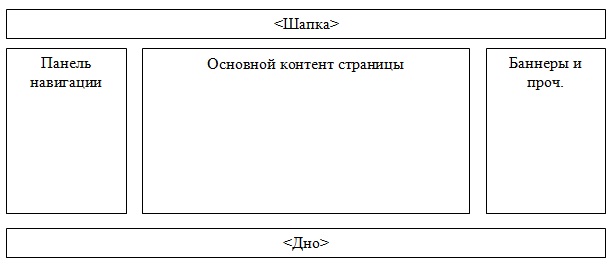
width: 20% (в зависимости от значения аналогичного атрибута родителя)

Также допустимо использование следующих значений атрибутов управления размерами:

* *auto* – размер устанавливается в зависимости от размеров *контента*;
* *inherit* – значение наследуется от родителя.

**Управление размещением контейнеров**

Как и любой *блочный элемент*, *контент* тега <div> начнется с новой строки. Содержимое нескольких *тегов* <div> будет расположено вертикально, друг под другом. Допустим, что мы хотим получить веб - страницу следующего вида:



**Рис. 8.1.**Макет веб - страницы

Добавим следующий *html* - код для размещения пяти контейеров:

<div id="top">top</div>

<div id="left">left </div>

<div id="content">content</div>

<div id="right">right</div>

<div id="bottom">bottom</div>

В *таблицу стилей* добавим соответствующие стили для каждого контейнера:

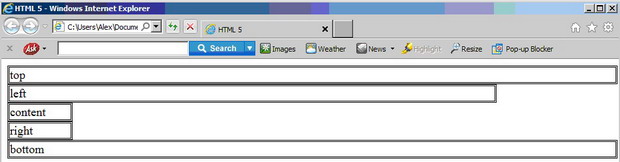
#top { height: 20px; width:412px; border: 3px double black}

#left { height: 200px; width:50px; border: 3px double black; }

#content { height: 200px; width:300px; border: 3px double black; }

#bottom { height: 20px; width:412px; border: 3px double black; }

Для наглядности, для каждого контейнера указаны параметры отрисовки его границ. В результате получим следующее:



**Рис. 8.2.**Результат размещения контейнеров веб - страницы по - умолчанию

Очевидно, что *размещение* контейнеров, используемое *по* умолчанию, не подходит для наших целей.

Для управления размещением элементов используется *атрибут* стиля *float*, принимающий следующие значения:

* left – элемент выравнивается по левому краю родителя, остальные элементы "обтекают" указанный по правой стороне;
* right – элемент выравнивается по правому краю родителя, остальные элементы "обтекают" указанный по левой стороне;
* none – обтекание элемента не указано;
* *inherit* – значение наследуется от родителя.

Для того чтобы элементы расположились *по* горизонтали один за другим, необходимо задать одно и то же *значение* *float* для следующих друг за другом элементов.

Для размещения блоков ниже выровненных *по* горизонтали, необходимо использовать *атрибут* стиля *clear*, явно указывающий на то, что данный блок должен располагаться ниже предшествующих ему контейнеров.

*Атрибут* *clear* может принимать следующие значения:

* left – элемент будет расположен ниже всех элементов, значение атрибута *float* у которых равно left;
* right – элемент будет расположен ниже всех элементов, значение атрибута *float* у которых равно right;
* both – элемент будет расположен ниже всех элементов, значение атрибута *float* у которых равно left или right;
* none – отмена свойств атрибута *clear*;
* *inherit* –значение наследуется от родителя.

Изменим стили контейнеров из предыдущего примера следующим образом:

#top { height: 20px; width:412px; border: 3px double black}

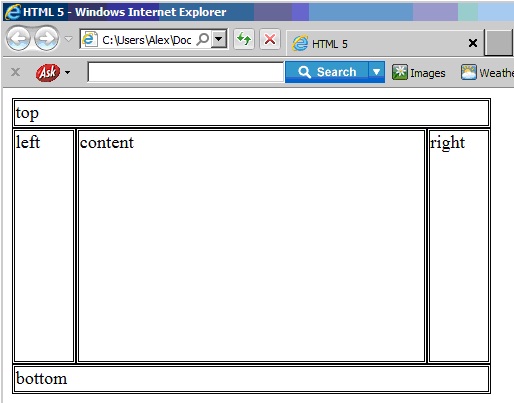
#left { height: 200px; width:50px; border: 3px double black; float: left}

#content { height: 200px; width:300px; border: 3px double black; float:left}

#right { height: 200px; width:50px; border: 3px double black; float:left}

#bottom { height: 20px; width:412px; border: 3px double black; clear: left}

Этого достаточно для того, чтобы получить желаемый результат:



**Рис. 8.3.**Результат размещения контейнеров при помощи атрибутов стилей float и clear

**Переполнение контейнеров**

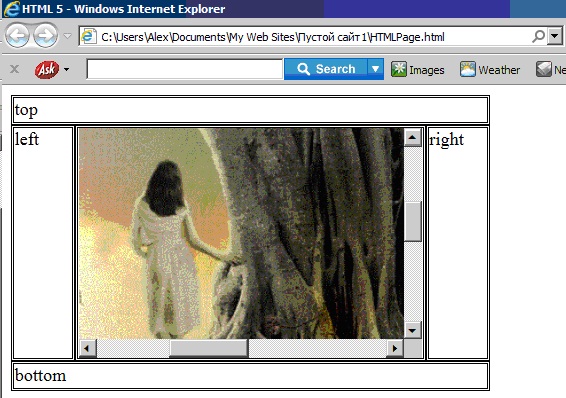
В случае если размер контейнера указан явно, то может возникнуть ситуация, когда размер *контента* превышает границы размеров контейнера.

Для управления отображением содержимого контейнера в таких случаях используется *атрибут* стиля *overflow*, принимающий следующие значения:

* *visible* – отображается весь контент даже за пределами контейнера;
* *hidden* – отображается только область внутри контейнера, оставшаяся часть *контента* скрывается;
* *scroll* – добавление *полос прокрутки* контейнеру, полосы будут отображаться даже если в них нет необходимости;
* *auto* – полосы прокрутки появятся только в случае необходимости.

Существует возможность управления отображением содержания контейнера отдельно *по* горизонтали и вертикали, соответственно при помощи атрибутов *overflow*-x и *overflow*-y, значения которых аналогичны значениям *overflow*.

Следующий рисунок демонстрирует как будет отображаться в рамках контейнера изображение, размер которого значительно превышает размеры контейнера.



**Рис. 8.4.**Пример управления отображением содержимого контейнеров

В рамках последующих практических занятий будут более детально рассмотрены примеры с использованием контейнерного дизайна.

**Ключевые термины и определения**

**Блочный элемент** – элемент, имеющий форму прямоугольника, *по* умолчанию занимает всю доступную ширину, *высота* элемента регулируется его содержимым.

**html-контейнер** – элемент, внутри которого могут содержаться другие элементы.

**Краткие итоги**

Были рассмотрены различные подходы к дизайну веб-страниц. Текстовый и фреймовый подходы являются устаревшими. В настоящий момент времени широко используются табличный и контейнерный виды дизайна. Несмотря на то, что большинство источников рекомендует для целей верстки сайтов пользоваться только контейнерами, из-за простоты и понятности можно встретить большое количество сайтов, *дизайн* которых основан на табличном подходе.

Также были описаны ключевые моменты управления внешним видом контейнеров при помощи каскадных таблиц стилей.

Работа *CSS* будет также затронута в рамках последующих практических занятий.

**Список материалов для самостоятельного изучения**

* <http://elearn.oknemuan.ru/>
* <http://www.i2r.ru/static/479/out_19595.shtml>
* <http://itfn.ru/articles/webdesign/frame-site/>
* <http://marvic.ru/zapiski/verstka/chto-luchshe-div-ili-table-moi-otvet>
* <http://dev.by/blog/7824>
* <http://ru.html.net/tutorials/css/lesson8.php>
* <http://www.codenet.ru/webmast/js/javascript/glava1_2.php>
* <http://www.codenet.ru/webmast/js/javascript/glava1_2_1.php>
* <http://htmlbook.ru/html/attr/common>
* <http://htmlbook.ru/css/clear>
* <http://htmlbook.ru/css/float>