**Лабораторная работа 1 «Основы HTML, CSS в веб-дизайне»**

**Цель работы:** Научиться оформлять веб - страницы с использованием каскадных таблиц стилей CSS. 

**Теория**

1. **CSS-свойства: оформление таблиц**

**Таблицы**

Свойства CSS могут применяться к таблицам, их строкам и ячейкам для за­дания свойств текста и шрифта, управления фоном, полями, границами, раз­мерами и т.п.

Создадим таблицу и применим к ней CSS -стили. В таблицу внесем данные о популярности различных браузеров. Для заголовка таблицы используем

тег <th>...</th>.

**HTML-код:**

Без CSS-оформления таблица будет выглядеть так:

**Год\Браузер IE Firefox Safari Opera**

2010 61.43% 24.40% 4.55% 2.37%

2009 69.13% 22.67% 3.58% 2.18%

2008 77.83% 16.86% 2.65% 1.84%

2007 79.38% 14.35% 4.70% 0.50%

По умолчанию содержимое заголовочных ячеек отображается жирным шрифтом с выравниванием по центру.

Добавим в тег

<head>...</head> тег <style>...</style>, а к тегу <table>...</table> атрибут id="browser\_stats".

Запишем CSS-правила для таблицы.

Для заголовочных ячеек установим серый фон и отступ содержимого от границ (padding) в половину высоты строки, для ячеек с данными - выравнивание по правому краю и padding три десятых от высоты строки. Вокруг таблицы зададим двойную рамку, а для ячеек - обычную одинарную.

<style>

/\* стиль таблицы \*/

TABLE#browser\_stats {

border: Зрх double black; }

/\* стиль заголовочных ячеек \*/

TABLE#browser\_stats

TH{

border: 1px solid black;

background-color: gray;

padding: 0.5em; }

/\* стиль ячеек с данными \*/

TABLE#browser\_stats

TD{

border: 1px solid black;

padding: 0.3em;

text-align: right; }

</style>

В браузере:

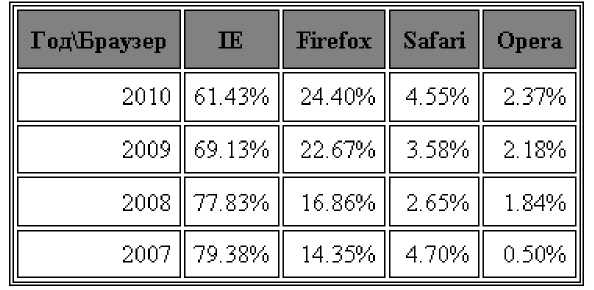


Рисунок 1. Отображение таблицы с заданными CSS-стилями

Виден существенный недостаток: у каждой ячейки появилась собственная рамка. Чтобы этого не происходило, необходимо указать в правилах для таб­лицы свойство **border**-**collapse** со значением **collapse**.

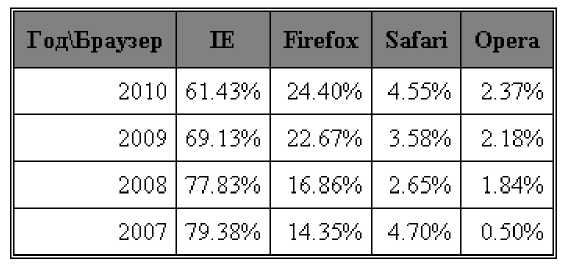
:

Рисунок 2. Эффект слияния границ соседних ячеек

Теперь применим к той же таблице другое форматирование. Разделим табли­цу двумя линиями на 3 части: названия браузеров, годы и процентные дан­ные. Названия браузеров и процентные доли выровняем по центру, годы - по правому краю. Зададим одинаковую ширину для столбцов с информацией по браузерам.

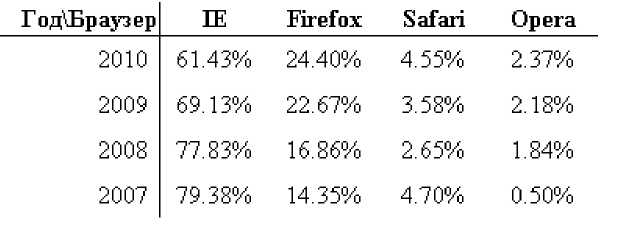


Рисунок 3 Оформление таблицы с двумя разделительными линиями

Чтобы применить правила CSS к левой колонке (годы), нам придется задать новый класс 1с и прописать атрибут class="lc" вовсе ячейки левой колонки.

Горизонтальная линия создается путем указания свойства border-bottom для ячеек тн, вертикальная - border-left для ячеек класса 1с

Код-страницы:

<html>

<head>

<title>Популярность браузеров в Mиpe</title>

<style>

TABLE#browser\_stats {

border-collapse: collapse;

}

TABLE#browser\_stats TH{ border-bottom: 1px solid black; }

TABLE#browser\_stats TD{ padding: 0.3em; text-align: center; width: 7 0px; }

TABLE#browser\_stats .lc{ text-align: right; border-right: 1px solid black; width: 100px; }

</style> </head> <body>

<table id="browser\_stats"> <tr>

<th class="lc">rofl\Bpaysep</th>

<th>IE</th>

<th>Firefox</th>

<th>Safari</th>

<th>Opera</th> </tr> <tr>

<td class="lc">2010</td>

<td>61.43%</td>

<td>24.40%</td>

<td>4.55%</td>

<td>2.37%</td> </tr> <tr>

<td class="lc">2009</td>

<td>69.13%</td>

<td>22.67%</td>

<td>3.58%</td>

<td>2.18%</td> </tr> <tr>

<td class="lc">2008</td>

<td>77.83%</td>

<td>16.86%</td>

<td>2.65%</td>

<td>1.84%</td> </tr> <tr>

<td class="lc">2007</td>

<td>79.38%</td>

<td>14.35%</td>

<td>4.70%</td>

<td>0.50%</td>

</tr>

</table>

</body>

</html>

**Псевдоклассы**

Способы привязки правил оформления CSS к элементам документа HTML могут быть: по названию тега, по имени класса, по ID и т.п. В CSS также существует несколько **псевдоклассов**. С помощью псевдоклассов можно задать стиль в зависимости от состояния элемента или его положения в документе.

Для ссылок определено 4 псевдокласса:

**link** - ссылки, которые не посещались пользователем;

**visited** - посещенные ссылки;

**active** - активная (нажатая) ссылка;

**hover** - ссылка, на которую наведен курсор.

Пример:

<html>

<head>

<title>IIpимep</title>

<style>

A:link, A:visited {

color: black;

font-family: Verdana, sans-serif;

text-decoration: none;

}

A:hover {

color: #de7300;

text-decoration: underline;

}

</style>

</head>

<body>

<a href="index.html">Главная</а><br>

<a href="hobby.html">Moe хобби</а><br>

<a href="photo.html">Фотоальбом</а><br>

</body>

</html>

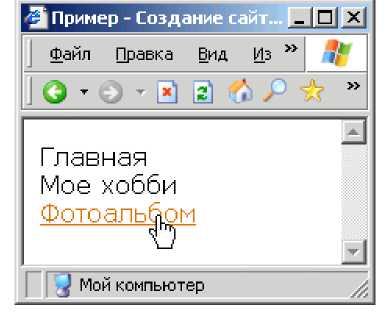


Рисунок 4. Пример: меню сайта

**Позиционирование**

С помощью CSS можно точно задать положение элемента на странице. Ре­жимом позиционирования управляет свойство position:

<html>

<head>

<title>Position: absolute</title>

<style>

DIV {

width: 100px;

height: 100px;

border: 3px double black;

padding: 5px;

position: absolute;

}

DIV#first {

background-color: #c0dcc0; top: 40px; left: 40px; }

DIV#second {

background-color: #c0c0dc; top: 80px; left: 100px; }

</style>

</head>

<body>

<div id="first">1</div>

<div id="second">2</div>

</body>

</html>

Для блоков задается отступ от верхнего и левого края свойствами top и left. Так как второй блок объявлен в HTML-коде позже, он перекрывает первый блок на странице.

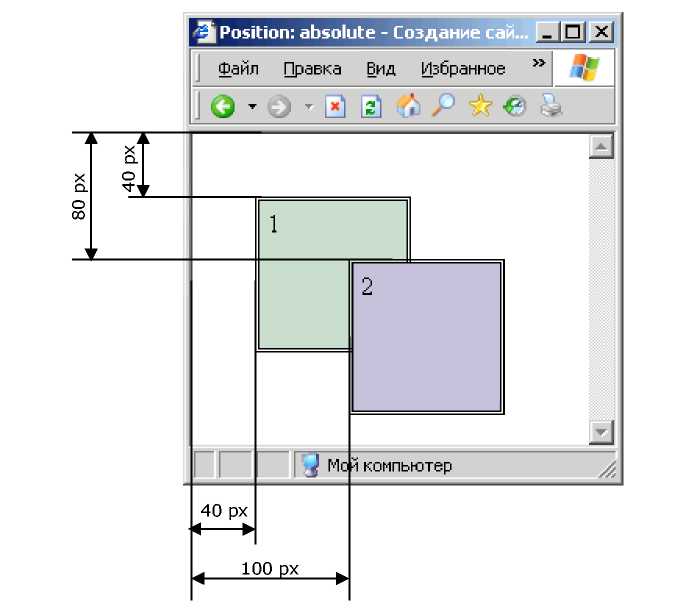


Рисунок 5. Использование абсолютного позиционирования.

Для управления порядком наложения элементов друг на друга необходимо использовать свойство z-index.

**Плавающие элементы**

По умолчанию блочные элементы идут строго друг под другом. Изменить этот порядок можно сделав элементы «плавающими». Для этого служит CSS атрибут float. Он задает, по какой стороне будет выравниваться элемент: левой (left) или правой (right). Плавающий элемент будет стремиться к левой или правой стороне родитель­ского элемента, а с других сторон он может обтекаться текстом или другими элементами.

Создадим пример с несколькими плавающими блоками. Зададим основной контейнер с фиксированной шириной, а в него поместим пять плавающих блоков с выравниванием по левому краю.

<html>

<head>

<title>Плавающие элементы</title>

<style>

DIV#main {

border: double black 3px;

width: 150px;

padding: 5px;

}

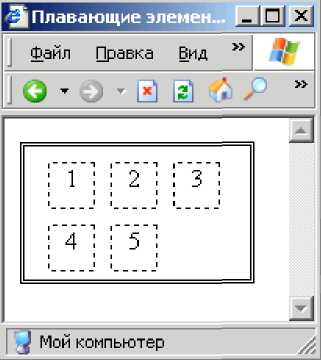


Рисунок 6. Пример: несколько плавающих блоков

Первый блок выравнивается по левому краю родительского контейнера. Вто­рой блок тоже стремится к левому краю, но так как место уже занято первым блоком, второй блок становится (обтекает) справа от первого. Аналогично поступает третий блок. Четвертый блок уже не может встать справа от третьего, поэтому он помещается ниже остальных и выравнивается по левому краю. И наконец, пятый блок обтекает четвертый справа.

Можно одновременно использовать блоки с выравниванием по левому и пра­вому краю.

<div style="float: left">&larr; Haлево</div> <div style="float: right">направо &rarr;</div>

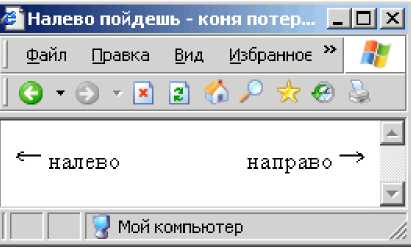


Рисунок 7. Пример: блоки с выравниванием по разным краям.

Еще одним свойством, связанным с плавающими элементами, является clear, Clear запрещает обтекание элемента с левой (left), правой (right) или с обеих сторон (both). По умолчанию значение - попе - обтекание раз­решено. Рассмотрим пример:

<html> <head>

<style>

DIV {

border: solid black 1px;

width: 75px;

}

DIV.floating { float: left; }

</style>

</head>

<body>

<div class="floating">Блок 1</div>

<div class="floating">Блок 2</div>

<div style="clear: both">Блок с запретом обтекания</div>

<div class="floating"><Блок 3</div>

<div class="floating">Блок 4</div>

<div class="floating">Блок 5</div>

</body>

</html>

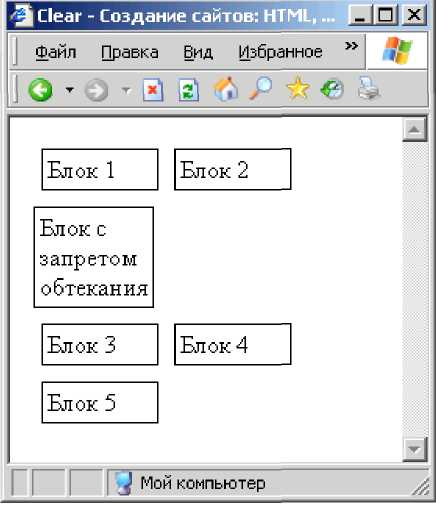


Рисунок 8 Использование правила clear.

При создании сайтов плавающие элементы, свойства float и clear часто используются для создания «каркаса» страниц сайта.

**Формы**

Формы были созданы и используются в WWW для получения отклика пользователя на предоставленную информацию и сбора данных о пользователе. После заполнения пользователем формы и запуска процесса ее обработки информация из нее попадает к программе, работающей на сервере. Простота использования тега <MAILTO:> в формах позволяет даже владельцам небольших страниц получать отклик от своих читателей. Для обработки большого количества откликов используются программы, поддерживающие Common Gateway Interface (CGI) и расположенные на сервере, в адрес которого поступают отклики. Таким образом пользователь может интерактивно взаимодействовать с Web-сервером через Internet.

**Задание формы — элемент FORM**

Элемент FORM обозначает документ как форму и определяет границы использования других тегов, размещаемых в форме. Тег <FORM> определяется последовательностью тегов <INPUT>, размещенных внутри пары <FORM> и </FORM>. В форме используется как метод (method), так и действие (action) для описания обработки данных, вводимых пользователем в форму. Метод (GET или POST) определяет, как должны обрабатываться входные данные из формы, а действие указывает на URI программы, ответственной за обработку этих данных.

<FORM METHOD="post" ACTION="mailto:yourname@your.email.address">

**Определение элементов управления формы — тег <INPUT>**

Данный тег используют для определения области внутри формы, куда вводятся данные. Он формирует поле для ввода информации пользователем. Это может быть текстовое поле, опция, изображение или кнопка. Вид поля ввода определяется значением атрибута TYPE.

**Атрибут TYPE=text**

Когда пользователю необходимо ввести небольшое количество текста (одну или несколько строк), используется тег <INPUT>, и атрибут TYPE устанавливается в значение text. Это значение принято по умолчанию и указывать его необязательно. Кроме того, задается атрибут NAME для определения наименования переменной поля.

Ваше имя <INPUT NAME="Name" SIZE="35">

Имеется еще три дополнительных атрибута, которые можно использовать. Первый называется MAXLENGTH, он ограничивает число символов, вводимых пользователем в текущее поле. По умолчанию данное число не ограничено. Вторым атрибутом является SIZE, определяющий размер видимой на экране области, занимаемой текущим полем. Значение по умолчанию определяется типом браузера. Если значение MAXLENGTH больше, чем SIZE, браузер будет прокручивать данные в окне. Последним из дополнительных атрибутов является атрибут VALUE, обеспечивающий начальное значение поля ввода.

**Атрибут TYPE=checkbox**

При создании форм часто требуется получить ответ пользователя на вопрос типа "Да/Нет". Для создания независимых кнопок в формах HTML используется тег <INPUT> с атрибутом TYPE=checkbox. В зависимости от содержания формы пользователь может отметить несколько флагов. Когда форма использует тег <INPUT> с атрибутом CHECKBOX, в нем должны присутствовать и атрибуты NAME, и VALUE. Атрибут NAME указывает на наименование данного поля (флага) ввода. В атрибуте VALUE будет содержаться значение поля.

Россия<INPUT NAME="Страна" TYPE=checkbox VALUE="Россия"> %

Страны СНГ<INPUT NAME="Страна" TYPE=checkbox VALUE="СНГ">

В некоторых случаях необходимо инициализировать данный флаг, как уже отмеченный. В таких случаях тег <INPUT> должен содержать атрибут CHECKED.

**Атрибут TYPE=radio**

В некоторых случаях требуется организовать выбор одного из нескольких возможных значений. Для создания формы ввода при выборе пользователем одного значения из нескольких возможных необходимо использовать тег <INPUT> с атрибутом TYPE=radio. Когда в форме применяется данный атрибут, в теге <INPUT> должны быть указаны атрибуты NAME и VALUE. Атрибут NAME указывает наименование соответствующего поля (кнопки). Атрибут VALUE содержит значение поля.

Пол мужской<INPUT NAME="Пол" TYPE=radio VALUE="Мужской"> %

Пол женский<INPUT NAME="Пол" TYPE=radio VALUE="Женский">

**Атрибут TYPE=image**

В зависимости от содержимого формы может случиться так, что пользователю потребуется щелкнуть мышью на изображении, чтобы завершить работу с формой. Для этого программисты используют тег <INPUT> с атрибутом TYPE=image. Когда пользователь щелкает мышью по изображению, браузер сохраняет координаты соответствующей точки экрана. Далее он "обрабатывает" введенную в форму информацию. Когда форма использует атрибут image, тег <INPUT> должен содержать также атрибуты NAME и SRC. NAME указывает наименование поля ввода формы. Атрибут SRC содержит URI файла — источника изображения. Атрибут ALIGN является дополнительным и используется аналогично тому же атрибуту тега <IMG>.

<BR>Выберите точку<INPUT TYPE=image NAME=point SRC="image.gif">

**Атрибут TYPE=password**

Если в форме требуется организовать ввод пароля, то атрибут TYPE можно установить в значение password (TYPE=password). Используя данный тип, можно организовать ввод пароля без вывода на экран составляющих его символов. При этом следует помнить, что введенные данные передаются по незащищенным каналам связи и могут быть перехвачены.

Подпись<INPUT NAME="login"> %

Пароль <INPUT TYPE=password NAME="pass">

**Атрибут TYPE=reset**

Когда пользователь заполняет форму, ему может потребоваться начать все сначала. На такой случай существует кнопка Reset, по которой пользователь может щелкнуть мышью, чтобы вернуться к первоначальным значениям полей. Когда пользователь выбирает данную кнопку, форма восстанавливает первоначальные значения всех элементов, в которых присутствует атрибут TYPE=reset. Для создания кнопки Reset используется тег <INPUT> с атрибутом TYPE=reset. Браузер в свою очередь будет выводить изображение данной кнопки. Если в форме используется атрибут reset, тег <INPUT> может дополнительно содержать атрибут VALUE. Данный атрибут определяет надпись на изображении кнопки.

<INPUT TYPE=reset VALUE="Очистить форму">

**Атрибут TYPE=submit**

Используя форму HTML для ввода информации от пользователя, необходимо обеспечить пользователю возможность завершить ввод данных. Для этого используется тег <INPUT> с атрибутом TYPE=submit. Браузер, в свою очередь, выводит данный элемент, как кнопку, по которой пользователь может щелкнуть, чтобы завершить процесс редактирования. Когда в форме используется тег <INPUT> с атрибутом submit, данный элемент может содержать два дополнительных атрибута: NAME и VALUE. Атрибут NAME хранит значение переменной поля в вашей форме. Атрибут VALUE — указывает наименование кнопки Submit.

<BR><INPUT TYPE=submit VALUE="Отправить сообщение"> %

**Атрибут TYPE=hidden**

Скрытые поля. Добавление в тег INPUT атрибута TYPE=hidden позволит включить в отправляемую форму значения атрибутов NAME и VALUE, которые пользователь изменить не может. Такие метки полезны при наличии нескольких форм для дальнейшей обработки данных.

**Создание многострочных областей ввода текста — тег <TEXTAREA>**

В зависимости от типа формы может потребоваться организовать ввод большого количества текста. В таких случаях используется тег <TEXTAREA> для создания текстового поля из нескольких строк. Данный тег использует три атрибута: COLS, NAME и ROWS.

**Атрибут COLS** Указывает (число символов) число колонок, содержащихся в текстовой области.

**Атрибут NAME** Определяет наименование поля.

**Атрибут ROWS** Задает количество видимых строк текстовой области.

<TEXTAREA NAME=тема COLS=38 ROWS=3> </TEXTAREA>

**Использование списков в форме — тег <SELECT>**

Когда формы HTML становятся более сложными, в них часто включают списки с прокруткой и выпадающие меню. Для этого используют тег SELECT с атрибутом TYPE=select. Для определения списка пунктов используют тег <OPTION>. Тег <SELECT> поддерживает три необязательных атрибута: MULTIPLE, NAME и SIZE.

**Атрибут MULTIPLE** Позволяет выбрать более чем одно наименование.

**Атрибут NAME** Определяет наименование объекта.

**Атрибут SIZE** Определяет число видимых пользователю пунктов списка. Если в форме установлено значение атрибута SIZE=1, то браузер выводит на экран список в виде выпадающего меню. В случае SIZE > 1 браузер представляет на экране обычный список.

В форме может использоваться тег <OPTION> только внутри тега <SELECT>. Эти теги поддерживают два дополнительных атрибута: SELECTED и VALUE.

**Атрибут SELECTED** Используется для первоначального выбора значения элемента по умолчанию.

**Атрибут VALUE** Указывает на значение, возвращаемое формой после выбора пользователем данного пункта. По умолчанию значение поля равно значению тега <OPTION> (открыть).

Выбор <SELECT NAME="Выбор"> %

<OPTION>Вариант 1 </OPTION>%

<OPTION>Вариант 2 </OPTION>%

<OPTION VALUE="Вариант 3">Вариант 3 </OPTION>%

<OPTION SELECTED>Вариант 4 </OPTION>%

</SELECT>

**Практическое задание**

Создать HTML-кода Вашего резюме и форму обратной связи

**1. Обязательные параметры резюме:** использование списков,

* фотография соискателя;
* имя соискателя и желаемая должность;
* образование;
* опыт работы;
* интересы;
* контакты.
* В поле «контакты» обязательно предусматривать поле email, телефон.
* Поместить рядом с каждым контактом соответствующую картинку (например, рядом с электронным адресом можно поместить конверт).

**2. Создать ссылку «Связаться со мной» на страницу с формой обратной связи**

* В форме предусмотреть поля: ФИО, телефон, email, Сообщение
* Форма должна быть внутри таблицы, одна колонка названия строк, вторая колонка поля.

**3. Обязательные параметры стилевых таблиц:**

* Заголовок (задать цвет, размер шрифта),
* Пункты «Опыт работы», «Образование», «Интересы» должны содержать не менее 3 предложений. Для всего текста должны быть заданы цвет, выравнивание и размер шрифта (название шрифта можно указать, используя свойство face тега font).