# Лабораторная работа №3 «Каскадные таблицы стилей CSS»

**Каскадные таблицы стилей CSS** (Cascading Style Sheets) позволяют задавать параметры стилей для всех элементов web-документа.

Благодаря стилям целой группе html-элементов можно присвоить одинаковое форматирование. В отличие от HTML, форматирование с помощью CSS предоставляет пользователю намного больше возможностей.

CSS предоставляет собой мощную систему для разработчиков, расширяя возможности в области дизайна и вёрстки web-страниц.

## 1.1. Встраивание CSS в HTML-документ

Согласно общепринятым представлениям о web-разработке следует соблюдать разделение содержимого сайта от его оформления. HTML-код должен содержать только теги разметки и теги логического форматирования. Любое оформление и дизайн должны быть вынесены за пределы кода в отдельный файл стилей. Благодаря этому над проектом могут работать сразу несколько web-разработчиков.

Существует три способа встраивания CSS-стилей в html-документ:

1. С помощью атрибута STYLE (метод встраивания, *inline*). Применяет стиль только к тому тегу, внутри которого прописан. Например:

<p style=”font-size: 24px; color: #bbffcc”>Абзац</p>

Стиль применится только для данного абзаца.

1. С помощью парного тега <STYLE> (метод внедрения, *embedding*). Данный тег добавляется в секцию *<head>…</head>* и позволяет управлять стилями страницы целиком. В этом случае формат описания стилей выглядит следующим образом:

|  |  |
| --- | --- |
| **HTML** | **CSS** |
| <h1>Заголовок</h1>  <p>Абзац №1</p>  <p>Абзац №2</p>  <p>Абзац №3</p>  <a href="">Ссылка</a> | <style type=”text/css”>  p  {  font-size: 24px;  font-family: Courier New;  color: #800000;  font-weight: bold;  }  h1, a  {  font-size: 16px;  font-family: Arial;  color: #4B0082;  }  </style> |

1. *Связывание*. Самым грамотным способом встраивания CSS в документ является создание и подключение внешнего файла стилей (*\*.css*). Такой файл может быть подключен к нескольким web-страницам, и его стили будут одинаково применяться ко всем элементам, для которых они прописаны. Для этого необходимо прописать в секции *<head>…</head>* следующую строку:

<link rel=”stylesheet” href=”css/styles.css”>

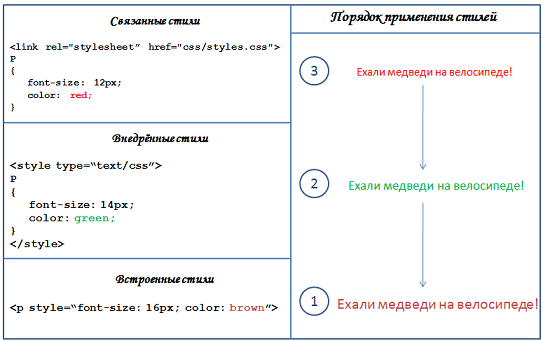
|  |
| --- |
| *Примечание.*  При копировании тегов из текста пособия меняйте кавычки вручную. Из-за особенностей шрифта *Courier New* копирование без замены кавычек не даст никакого результата. |

Атрибут *href=”css/styles.css”* говорит о том, что внешний файл со стилями *styles.css* хранится в папке **css** относительно текущей директории.

Так почему же таблицы стилей называются **каскадными**?

**Каскадность** заключается в том, что стили могут переопределяться. В зависимости от того, какими способами встраивания стилей вы пользуетесь, существует некоторая очерёдность переопределения (или применения) параметров стиля.

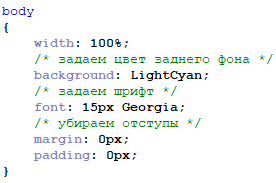
Если для одного и того же элемента стиль определен несколькими способами: внутри тега (метод встраивания), внутри страницы (метод внедрения) и во внешнем файле (метод связывания) порядок их применения к html-элементу строго определен.



|  |
| --- |
| *Задание.*  Создайте файл *styles.css* в папке **css** в своей рабочей директории и выполните его подключение в секции *<head>…</head>* файла *index.php*. |

## 1.2. Оформление «тела» документа

Первое, с чего следует начать оформление web-страницы – это задний фон и шрифты. С помощью селектора элемента **body** зададим стили для «тела» документа



Данные свойства будут распространяться на все элементы, находящиеся в «теле» документа.

С помощью свойств *margin* и *padding* настраиваются отступы элементов.

|  |
| --- |
| *Задание.*  Добавьте в файл *styles.css* стили для тела документа и проверьте, что они применились. |

Перед тем как приступить к оформлению приложения, следует разобраться с методом блочной модели построения приложений.

## 1.3. Блочная модель (box model)

Фундаментом блочной вёрстки служит универсальный парный тег **<div>…</div>**. Благодаря этому тегу всё пространство web-страницы разбивается на блоки. При этом код становится более компактным, чем при табличной вёрстке, и лучше индексируется поисковыми системами.

|  |
| --- |
| *Примечание.*  Элемент *<div>* 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. |

Суть блочной вёрстки состоит в следующем: всё содержимое сайта (шапка, логотип, меню, контент) распределяется по блокам *<div>*. В каждый блок помещается соответствующее содержимое – html-элементы. Далее каждый блок получает свой идентификатор и форматируется средствами CSS-разметки. Всё оформление хранится в отдельном css-файле.

Блочную модель можно представить следующим образом (см. рис. 1.1).

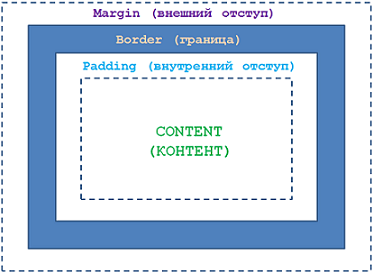


Рисунок 1.1. Структура блочной модели

Блок включает в себя следующие составляющие: внешние отступы (*margin*), границы (*border*), внешние отступы (*margin*) и основной контент (*content*).

Структура web-страницы при использовании блочной модели представляет собой набор контейнеров *<div>*, каждый из которых содержит собственный набор элементов.

Перейдём к оформлению главной страницы приложения с использованием блочной модели.

## 1.4. Оформление главной страницы приложения

Разделим содержимое главной страницы на четыре <div>-блока (см. рис. 1.2).

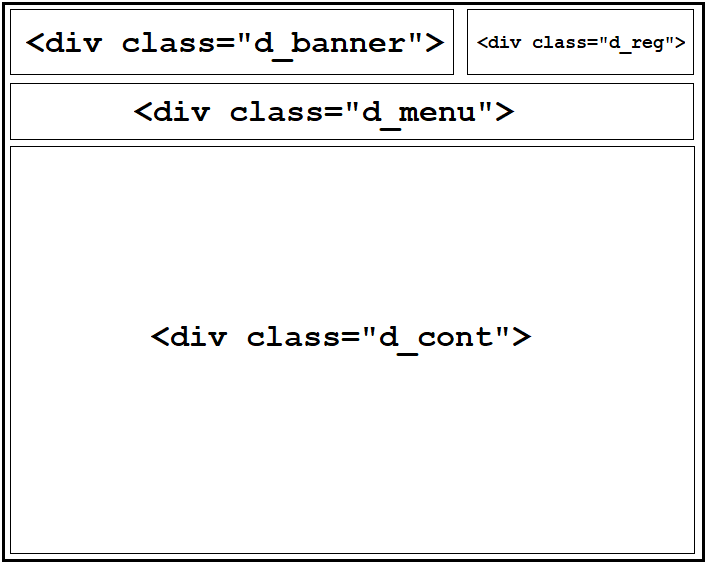


Рисунок 1.2. Представление главной страницы приложения в виде div-контейнеров

Блок с классом *class=”d\_banner”* служит для расположения в нём изображения-баннера.

Блок с классом *class=”d\_reg”* служит для расположения в нём ссылок на регистрацию или авторизацию пользователей в приложении.

Блок с классом *class=”d\_menu”* служит для расположения в нём главного меню приложения.

Блок с классом *class=”d\_cont”* служит для расположения содержимого страницы.

На всех страницах приложения два верхних блока должны повторяться.

Код для создания блока выглядит следующим образом:



То есть, созданный нами ранее код для вставки баннера-ссылки, заносится в div-контейнер с указанием класса. Дополнительно с помощью тега *align* осуществляется расположение содержимого блока по центру.

|  |
| --- |
| *Задание.*  Создайте div-контейнеры для блоков *d\_reg*, *d\_menu* и *d\_cont* самостоятельно. Свойство атрибута *align=”center”* для блоков *d\_reg* и *d\_menu* указывать не требуется. |

После условного разделения страницы на блоки, зададим каждому блоку индивидуальные стили. Для этого необходимо открыть файл **styles.css**, обратиться к классу, указанному внутри div-контейнера, и прописать необходимые стили.

|  |  |
| --- | --- |
| **Блок** | **Стили** |
| *.d\_banner* | .d\_banner  {  position: relative;  float: left;  width: 75%;  height: 90px;  }  /\* масштабирование изображения в блоке div \*/  .d\_banner img  {  max-width: 100%;  height: auto;  max-height:100%;  } |
| *.d\_reg* | .d\_reg  {  position: relative;  left: 30px;  margin-top: 10px;  bottom: 5px;  float: left;  border: 2px outset CornflowerBlue;  background: LightCyan;  border-radius: 10px;  padding: 5px 5px 5px 5px;  box-shadow: 0 0 10px rgba(0,0,0,0.5);  } |
| *.d\_menu* | .d\_menu  {  /\* Задаём ширину блока \*/  width: 100%;  /\* Задаём отступ \*/  margin: 4px;  float: none;  clear: both;  } |
| *.d\_cont* | .d\_cont  {  /\* Выравниваем блок по центру и задаём соответствующие отступы \*/  padding-top: 10px;  margin:0 15% 0 15%;  width:70%;  height: 700px;  position: relative;  } |

|  |
| --- |
| *Задание.*  Пропишите в файле *styles.css* указанные выше стили и проверьте результат. |

## 1.5. Оформление главного меню приложения

Меню приложения будет представлять собой выпадающий список. Для этого требуется соответствующим образом задать стили для элементов списка. Зададим отдельные классы для списков первого уровня и второго уровней. Списки второго уровня – это списки, вложенные в элементы списка первого уровня (см. рис. 1.3).

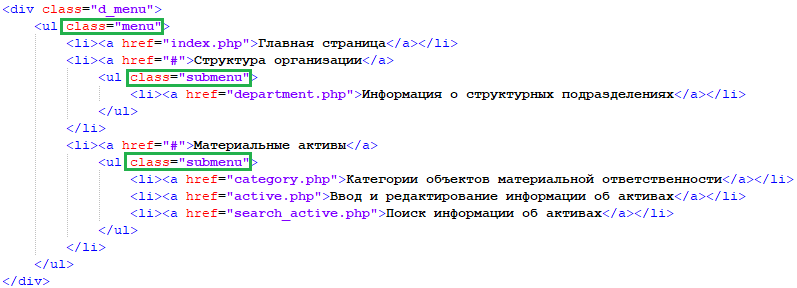


Рисунок 1.3. Указание классов для элементов списка

Здесь в списке первого уровня задан класс **menu** (*class=”menu”*). Для списка второго уровня задан класс **submenu** (*class=”submenu”*). Стили CSS требуется задать таким образом, чтобы получился выпадающий список, в котором списки второго уровня изначально скрыты и появляются только при наведении. Перейдём к описанию стилей.

|  |  |
| --- | --- |
| **Код** | **Описание** |
| ul {  /\* Убираем маркеры у списка \*/  list-style: none;  /\* Делаем элементы блочными \*/  display: block;  /\* Убираем отступы \*/  margin: 0px;  padding: 0px;  /\* Задаём ширину блока \*/  width: 100%;  } | Задаём стили для любого списка, который находится в теге <ul>…</ul>. |
| ul:after {  /\* Делаем элементы блочными \*/  display: block;  /\* Убираем выравнивание \*/  float: none;  content: ' ';  /\* Отменяем обтекание элемента одновременно с правого и левого края \*/  clear: both;  } | Псевдо-элемент :after создаёт виртуальный элемент, добавляет в него содержимое и затем добавляет этот элемент в качестве дочернего элемента после уже существующего содержимого выбранного элемента. Используется только совместно со свойством content, которое указывает, какое именно содержимое будет иметь псевдо-элемент. |
| ul.menu > li {  /\* Задаём выравнивание и позиционирование \*/  float: left;  /\* Считаем координаты относительно исходного места \*/  position: relative;  /\* Задаём ширину элемента \*/  width: 33%;  } | Задаём стили для элементов списка первого уровня (class=”menu”). |
| ul.menu > li > a {  /\* Делаем элементы блочными \*/  display: block;  /\* Задаем белый цвет \*/  color: #fff;  /\* Задаем отступ \*/  padding: 10px;  /\* Убираем форматирование \*/  text-decoration: none;  /\* Задаем цвет заднего фона \*/  background-color: SlateBlue;  /\* Выравниваем текст по центру \*/  text-align: center;  } | Задаёт стили для ссылок, которые являются элементами списка первого уровня. |
| ul.menu > li > a:hover {  /\* Задаем цвет заднего фона при наведении \*/  background-color: MediumPurple;  } | Задаёт стили для ссылок, которые являются элементами списка первого уровня. Стили применяются при наведении на элемент списка. |
| ul.submenu {  /\* Элемент абсолютно позиционирован \*/  position: absolute;  /\* Определяем расстояние от верхнего и левого края родительского элемента до верхнего края дочернего элемента \*/  top: 37px;  left: 0px;  /\* Делаем подменю скрытыми \*/  display: none;  /\* Цвет заднего фона — белый \*/  background-color: white;  /\* Выравниваем текст по центру \*/  text-align: center;  z-index: 999;  } | Задаём стили для списка второго уровня (подменю). По умолчанию элементы этого меню скрыты.  Свойство *z-index* предназначено для позиционирования элементов по оси Z. Чем больше значение свойства z-index, тем ближе элемент расположен к нам, и наоборот, чем меньше значение, тем дальше расположен элемент от нас. |
| ul.submenu > li {  /\* Блочное расположение элементов \*/  display: block;  } | Задаём стили для элементов списка подменю. |
| ul.submenu > li > a {  /\* Делаем элементы блочными \*/  display: block;  /\* Убираем форматирование \*/  text-decoration: none;  /\* Задаем отступ \*/  padding: 10px;  /\* Задаем цвет \*/  color: #ffffff;  /\* Задаем цвет заднего фона \*/  background-color: SlateBlue;  } | Задаёт стили для ссылок, которые являются элементами списка второго уровня (подменю). |
| ul.submenu > li > a:hover {  /\* Цвет заднего фона при наведении \*/  background-color: MediumPurple;  /\* Убираем подчеркивание при наведении \*/  text-decoration: none;  } | Задаёт стили для ссылок, которые являются элементами списка второго уровня (подменю). Стили применяются при наведении на элемент списка. |
| ul.menu > li:hover > ul.submenu {  /\* Делаем элементы блочными \*/  display: block;  } | Задает стили для элементов списка второго уровня при наведении на элементы списка первого уровня. |

В результате после указания описанных выше стилей выпадающее меню будет выглядеть следующим образом (см. рис. 1.4).

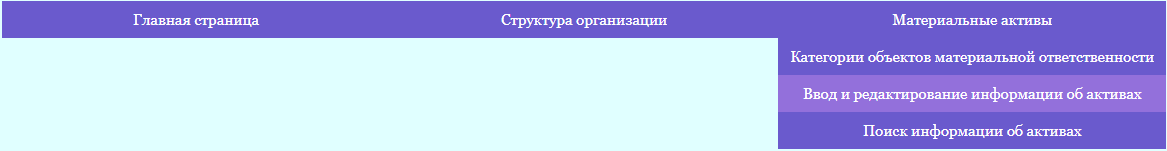


Рисунок 1.4. Внешний вид выпадающего меню в браузере

|  |
| --- |
| *Задание.*  Внесите в файл *styles.css* соответствующие изменения и проверьте результат. |

Результирующий вид главной страницы приложения после добавления всех стилей представлен на рис. 1.5.



Рисунок 1.5. Вид главной страницы приложения после внесения в файл *styles.css* соответствующих стилей

В процессе разработки приложения ещё не раз потребуется прописывать стили для различных элементов. Каждая страница приложения будет ссылаться на файл *styles.css* и к ней будут применяться соответствующие стили.

Следующим шагом при создании приложения является разработка базы данных и организация доступа к ней.

|  |
| --- |
| *Задание.*  Убедитесь, что ваша главная страница содержит все представленные выше изменения и выглядит как представленная на рис. 1.5. |