**Лекция 9. Основы CSS. Особенности CSS.**

источников попросту пересказывают друг друга, если не дублируют полностью. К примеру, набрав в поисковике *CSS*, можно очень быстро узнать, что это:

1. каскадные таблицы стилей;
2. используется для управления внешним видом информации, предоставляемой сайтом.

Несколько сложнее узнать, как именно соотносится *CSS* с *HTML*, и еще сложнее – почему для оформления *HTML* документа нельзя обойтись собственными возможностями *HTML*, тем более, что соответствующие атрибуты присутствуют в большинстве *тегов*. В качестве введения к данной лекции, мы постараемся внести некоторую *ясность* в эти вопросы.

**Возникновение CSS**

Во второй лекции нашего курса приведена краткая история возникновения *HTML*. Частично написанное сейчас будет пересекаться с прошлым материалом.

Первый стандарт *HTML* не содержал структур, позволяющих отображать текст каким-либо особенным образом, то есть не было никаких средств управления внешним видом информации. Первоначальной задачей было обеспечение доступности информации для любых устройств.

С дальнейшим распространением *HTML* такие компании, как *Microsoft* и *Netscape* стали внедрять свои собственные *теги* или "улучшать" имеющиеся именно с целью управления оформлением *html* - документов. Часть внесенных изменений прижилась и "ушла" в массы.

После случилось логически неизбежное, а именно, *HTML* стал представлять собой набор несовместимых между собой *тегов* и расширений. Результатом исправления ситуации стал стандарт *HTML* 3.2, "узаконивший" ряд изменений и устранивший проблемы совместимости.

Таким образом, если несколько упростить, то можно сказать, что все приемы и *теги*, направленные на управление внешним видом предоставляемой информации, по своей сути, являются атавизмом предыдущих версий *HTML*.

Стандарт *HTML* 3.2 лишь исправил ряд наиболее серьезных недостатков предыдущих версий. Настоящим же решением проблемы явился стандарт *HTML* 4.0, в рамках которого было предложено отделить описание структуры *html* документа от его оформления.

Если уж совсем "рыться" в истории, то можно заметить, что и этот подход не нов. SGML, на котором и основывалась первая версия HTML, в числе прочего предполагал наличие отдельного "файла стилей" документа.

Таким образом, возникновение *каскадных таблиц стилей* является закономерным результатом эволюционного развития стандартов *HTML*.

**Суть и преимущества CSS**

*CSS* (*Cascading Style Sheets* – каскадные таблицы стилей) – язык описания внешнего вида документа, созданного при помощи *языка разметки*.

Концепция *каскадных таблиц стилей* была предложена Хоконом Виум Ли – норвежским ученым и специалистом в области *информационных технологий*, работавшим в то время на консорциум *W3C*.

Как правило, *CSS* применяется при работе с *HTML* и *XHTML* языками, и используется для задания цветов, параметров шрифтов, расположения блоков и иных элементов представления веб - страниц.

К преимуществам использования *CSS* относятся:

* *централизованное управление* *отображением множества* документов при помощи одной таблицы стилей;
* упрощенный контроль внешнего вида веб - страниц;
* наличие разработанных дизайнерских техник;
* возможность использования различных стилей для одного документа, в зависимости от устройства, при помощи которого осуществляется доступ к веб - странице.

Стандарт *HTML* 4.0 помимо *CSS* утвердил и объектную модель *браузера* (*Browser Object Model* – *BOM*), о которой следует сказать отдельно.

*Объектная модель* *браузера* описывает содержимое веб - документа, т.е. сама модель является набором объектов, описывающих указанное содержимое. Поскольку *BOM* уникальна для каждого *браузера*, возникали проблемы с межплатформенными приложениями. В конечном итоге, на *место* *объектной модели* *браузера* пришла *объектная модель документа* (*Document Object Model – DOM*), описывающая стандарт представления веб - страниц в виде набора объектов.

**Краткая история CSS**

Описание возникновения *CSS* будет неполным, без краткой истории версий *каскадных таблиц стилей*.

|  |
| --- |
| Таблица 6.1. |
| **Версия** | **Дата принятия** | **Предоставляемые возможности** |
| *CSS1* | 17.01.1996 | * Управление способом отображения элемента на странице.
* Задание настроек обтекания *элемента текстом*.
* Управление размерами элемента.
* Управление внешними и *внутренними отступами* элемента.
* Управление вертикальным выравниванием в таблицах.
* Управление стилями границ элементов.
* Управление форматированием списков.
* Управление *цветами текста* и фона.
* Управление параметрами шрифтов.
* Управление свойствами текста.
* Управление междустрочными *интервалами*.
 |
| *CSS2* | 12.05.1998 | Все возможности *CSS1* и ряд новых:* Управление направлением текста.
* Управление *позиционированием* элементов.
* *Управление видимыми* областями элементов.
* Управление отображением элементов, выходящих за границы заданных размеров.
* Управление внешним видом курсора.
* Управление расположением элементов по оси z (слои).
* Задание минимально и максимально возможных размеров элемента.
* Управление расстоянием между ячейками таблицы.
* Управление стилями границ элемента (каждой границы в отдельности).
* Управление размерами элементов таблиц.
* Управление стилями кавычек.
* Управление *контентом* при его печати.
* Управление звуковым оформлением *контента* (громкость, паузы и т.п.).
 |
| *CSS2.1* | 8.09.2009 | * Исправлен ряд ошибок *CSS2*.
* Изменились некоторые моменты, реализация которых в подавляющем большинстве *браузеров* отличается от спецификации *CSS2*.
* Убрали особенности *CSS2*, которые, в силу того, что не были реализованы, были отвергнуты *CSS* сообществом.
* Удалили фрагменты *CSS2*, которые будут устаревшими в CSS3.
* Добавили некоторые новые значения свойств.
 |
| CSS3 | разрабатывается | * Поддержка закругленных углов.
* Поддержка градиентных границ.
* Управление тенями элементов.
* Поддержка возможности использования изображений, в качестве границ элементов.
* Управление тенью текста.
* Поддержка нестандартных шрифтов.
* Возможность изменения размеров блоков пользователем.
* Возможность разбиения текста на несколько колонок.
* Управление фоновыми изображениями.
 |

В настоящее время стандарт CSS3 только разрабатывается. В связи с этим нами перечислен только ряд его новшеств. Подробнее с ними, на данном этапе, предлагается ознакомиться самостоятельно.

**Отношения между множественными вложенными элементами**

В *html* - документе элементы (*теги*) могут находиться в рамках других элементов. Отношения между вложенными элементами могут быть *родительскими*, *дочерними* и *братскими* (в ряде литературы также встречается название *сестринские*). Поясним эти и иные термины, относящиеся к структуре *html* - документа:

*Дерево документа* – воображаемая древовидная структура элементов в *html* - документе, *синоним* понятия *объектная модель документа* (*DOM*).

*Родительский* элемент – элемент, содержащий в себе рассматриваемый элемент. В записи вида <p> <strong> ... </strong> </p>, элемент <p> является родительским по отношению к <strong>.

*Предок* – элемент на несколько уровней выше и содержащий в себе рассматриваемый элемент. Т.е. в записи вида <*body*>... <p> <strong> ... </strong> </p>... </*body*>, <*body*> является предком strong.

*Дочерний* элемент – элемент, находящийся внутри рассматриваемого документа. В записи вида <p> <strong> ... </strong> </p>, элемент <strong> является дочерним по отношению к <p>.

*Потомок* – элемент, находящийся внутри рассматриваемого элемента и находящийся на несколько уровней ниже. В записи вида <*body*>... <p> <strong> ... </strong> </p>... </*body*>, <strong> является потомком <*body*>.

*Братский* элемент – элемент, имеющий общий родительский элемент с рассматриваемым. Т.е. в записи <p> <strong> ... </strong> ... <*img* ...></p>, элементы <*img*> и <strong> являются братскими.

**Создание каскадных таблиц стилей**

Нами было достаточно сказано о причинах возникновения *CSS* и его возможностях. Рассмотрим формат *каскадных таблиц стилей*, правила их создания и способах установления связей с *html* - документом.

В общем виде *CSS* задается следующим образом:

<селектор> {

 <атрибут стиля №1>: <значение атрибута>;

 <атрибут стиля №2>: <значение атрибута>;

...

 <атрибут стиля №N>: <значение атрибута>;

 }

**Селектором** называется формальное описание элемента, или их группы, к которому должны быть применены описанные правила стиля.

**Атрибутом\_стиля** называется один из параметров элемента веб - страницы (в частных случаях схожи с атрибутами *тегов*, но следует помнить, что это разные вещи).

*CSS* стили отделяются друг от друга пробелами, либо символами переноса строк. **Не должно быть символов**';' **между различными стилями!**

В описании селекторов и имен стилей не должно быть пробелов или *переводов строк*.

Рассмотрим возможные виды селекторов, способы их описания и *представление* в *html* - документе.

|  |
| --- |
| Таблица 6.2. |
| **Селектор** | **Оформление в*****CSS*** | **Оформление элемента, к которому применяется стиль в HTML** | **Пояснения** |
| Селектор элементов (*переопределение* *тега*) | P {*color*: *red* } | <p>...</p> | Стиль будет применен ко всем абзацам документа. Т.е. весь текст, находящийся в рамках парного *тега* <p> будет выделен красным цветом. |
| Селектор классов (стилевой класс) | .classname {*color*: *red*} | <p class="classname">...</p> | Стиль может быть применен к любому *тегу*, содержащему атрибут class, значение которого совпадает с наименованием селектора класса. |
| Селектор идентификаторов (именованный стиль) | #clrRed { *color*: *red* } | <p id="clrRed">...</p> | Стиль будет применен к любому *тегу*, содержащему атрибут id, значение которого совпадает с наименованием селектора стиля. В рамках одной веб страницы значение атрибута id должно быть уникальным. Т.е., фактически, данный стиль может быть применен единовременно только к одному элементу веб - страницы. |
| *Селектор дочерних элементов* | #clrRed>strong { *color*: *red*} | <p id="clrRed"> <strong>...</strong> </p> | Стиль будет применен ко всем элементам <strong>, находящимся в дочерних отношениях с <p id="clrRed"> и только к ним. |
| *Контекстный селектор* (комбинированный стиль) | p strong {*color* : *red*} | <p><strong>...</strong></p> | Стиль будет применен к тексту, в рамках *тега* <strong> следующего за *тегом* <p>. Фактически стиль привязывается к *тегу* <strong>. |
| p.classname {*color*: *red*} | <p class="classname">...</p> | Стиль будет применен ко всем тегам <p>, содержащим атрибут class, значение которого равно classname. |
| p.classname <strong> { *color*: *red*} | <p class = "classname"> <strong>...</strong></p> | Стиль будет применен к содержимому *тега* <strong>, находящемуся внутри *тега* <p>. Атрибут class *тега* <p>, при этом, должен принимать значение classname. |

Допускается создание одинаковых стилей, путем последовательного перечисления их селекторов. К примеру:

p, .classname, td strong { color: red }

В приведенном примере создается три одинаковых стиля: *переопределение* *тега* <p>, стилевой *класс* classname и комбинированный стиль для *тега* <strong>, находящегося в рамках *тега* <td>.

**Связь CSS и HTML**

Как уже упоминалось, *CSS* *файл* является внешним, по отношению к *html* - документу. Существует несколько способов "привязки" определенного файла стилей к конкретному документу.

Посредством *тега* <*link*>, находящимся в рамках парного *тега* <*head*>:

<head>

...........

<link rel="stylesheet" type="text/css" href="style.css">

</head>

*Атрибут* rel указывает на тип подключаемого файла, type – указывает *MIME* *тип файла*, *href* – указывает *путь* до файла стилей.

Посредством *директивы* @import, находящейся в рамках парного *тега* <*style*>:

<head>

.............

<style type = "text/css">

@import url(style.css)

</style>

</head>

Указанные способы позволяют подключить *внешнюю* таблицу стилей. Несмотря на наличие явной рекомендации по размещению в отдельных файлах, имеется возможность использования *таблиц стилей* в рамках самого *html* - документа:

*Таблица стилей* располагается в рамках парного *тега* <*style*>:

<head>

.............

<style type = "text/css">

p { color: red}

..............

</style>

</head>

*Таблица стилей* располагается в теле отдельного *тега*:

<p style="color: red"> ........ </p>

В этих случаях *таблица стилей* называется *внутренней*.

Внутренние таблицы полезнее тем, что являются неотъемлемой частью самого *html* - документа, т.е. веб - страница будет выглядеть, как должна, вне зависимости от доступности внешней таблицы стилей. У данного подхода есть и недостатки:

* внутренний стиль распространяется только на один веб - документ;
* применение внутренних стилей противоречит положению о необходимости отделения структуры документа от его представления.

**Правила и каскадность CSS**

Итак, существует несколько способов задания связи между *CSS* и *html* - документом. Кроме того, к одному и тому же элементу веб - страницы могут быть назначены несколько стилей (к примеру, в рамках самого элемента и во внешней таблице стилей). При этом фактическое *отображение* элемента регулируется правилами каскадности:

1. Внешняя *таблица стилей*, ссылка на которую встречается в html - документе позже, имеет приоритет по отношению к внешней таблице стилей, ссылка на которую встречается раньше.
2. Внутренние таблицы стилей приоритетнее внешних.
3. Таблицы стилей, располагающиеся в рамках самого элемента (<p *style*="...">), имеют приоритет, по отношению ко всем остальным стилям.
4. Более конкретные стили имеют приоритет перед менее конкретными (к примеру, p.classname {...} приоритетнее p {..}), т.е. стилевой класс приоритетнее переопределения *тега*, комбинированный класс приоритетнее стилевого.
5. В случае привязки к *тегу* нескольких стилевых классов, приоритетными считаются те, что указаны правее.
6. Атрибуты стиля, объявленные как !important, имеют приоритет перед всеми другими значениями. Таким образом, стиль p {*color*: *red* !important} сделает весь текст в рамках *тегов* <p> красным вне зависимости от любых других переопределений стиля для <p>.

**Ключевые термины и определения**

**CSS** – каскадная *таблица стилей*, язык описания внешнего вида *html* - документа.

**Объектная модель браузера** – уникальная для каждого *браузера* модель представления содержимого веб - документа.

**Объектная модель документа** – стандарт, регламентирующий *представление* содержимого веб - документа.

**CSS - селектор** – описание элемента или их группы, к которым должны быть применены правила стилей.

**Внешняя таблица стилей** – *таблица стилей*, располагающаяся во внешнем файле по отношению к *html* - документу.

**Внутренняя таблица стилей** – *таблица стилей*, являющаяся частью *html* - документа.

**Краткие итоги**

Основной *вывод*, который нужно вынести из данной лекции, заключается в следующем: *html* - разметка документа отвечает за структуру информации (блоки, абзацы, цитаты и т.д.), за управление внешним видом документа отвечает *CSS* *файл* (цвет, параметры шрифта, рамки, фон, обтекание и т.д.).

Нами были рассмотрены основы *CSS*, в рамках последующих лекций мы рассмотрим непосредственно приемы оформления отдельных элементов при помощи создания *каскадных таблиц стилей*.

**Список материалов для самостоятельного изучения**

1. <http://citforum.ru/internet/webd/article_31.shtml>
2. <http://design.originweb.info/css/rules.html>
3. <http://www.ruled.ru/css.html>
4. <http://severus.name/css3-novye-vozmozhnosti/>
5. <http://vremenno.net/html-css/css3-review/>
6. <http://www.web-palette.ru/sub/useful/8i/8j>
7. <http://www.ruled.ru/sintacs-css.html>
8. <http://www.03www.su/ex-0010.html>
9. <http://allxml.h1.ru/articles/dom.html>
10. <http://www.structuralist.narod.ru/it/internet/dom.htm>