**Понятие и проблемы кроссбраузерности в CSS. CSS-правила. Синтаксис CSS. Единицы измерения в CSS. Использование классов и идентификаторов в CSS.**

**CSS** (англ. Cascading Style Sheets — каскадные таблицы стилей) — технология описания внешнего вида документа, оформленного языком разметки.

*Преимущественно используется как средство оформления веб-страниц в формате HTML и XHTML, но может применяться с любыми видами документов в формате XML, включая SVG и XUL.*

*Каскадные таблицы стилей используются создателями веб-страниц для задания цветов, шрифтов, расположения и других аспектов представления веб-документа. Основной целью разработки CSS являлось разделение содержимого (написанного на HTML или другом языке разметки) и оформления документа (написанного на CSS). Это разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печать, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.*

**Каскадные таблицы стилей**(Cascading Style Sheets, CSS) — это стандарт, определяющий представление данных в браузере. Если HTML предоставляет информацию о структуре документа, то таблицы стилей сообщают как он должен выглядеть.

**Стиль** — это совокупность правил, применяемых к элементу гипертекста и определяющих способ его отображения. Стиль включает все типы элементов дизайна: шрифт, фон, текст, цвета ссылок, поля и расположение объектов на странице.

**Таблица стилей** — это совокупность стилей, применимых к гипертекстовому документу.

**Каскадирование**— это порядок применения различных стилей к веб-странице. Браузер, поддерживающий таблицы стилей, будет последовательно применять их в соответствии с приоритетом: сначала связанные, затем внедренные и, наконец, встроенные стили. Другой аспект каскадирования —*наследование (inheritance)*, — означает, что если не указано иное, то конкретный стиль будет применен ко всем дочерним элементами гипертекстового документа. *Например, если вы примените определенный цвет текста в теге <div>, то все теги внутри этого блока будут отображаться этим же цветом.*

*Если HTML используется для логического форматирования документа, то для управления его отображением на экране монитора или выводом на принтер применяются каскадные таблицы стилей. В CSS принята коробочная модель форматирования.*

*Использование каскадных таблиц дает возможность разделить содержимое и его представление и гибко управлять отображением гипертекстовых документов путем изменения стилей.*

Рассмотрим назначение каскадных таблиц стилей:

1) позволяют получить результаты, которые не могут быть получены средствами HTML. *Так, можно установить размер шрифта 40 pt, тогда как в HTML с помощью параметра size можно задать максимальный размер равный 7, что соответствует 36 pt*;

2) в сочетании с программами сценариев (JavaScript или VBScript) каскадные таблицы позволяют динамически изменять стиль отображения документа в окне браузера в зависимости от каких-либо действий пользователя (щелчок мышью, перемещение указателя и т. д.);

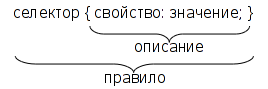
3) каскадные таблицы позволяют задать единый стиль оформления разных страниц документа и быстро поменять его, путем изменения нужного определения в таблице стилей.

*Таблицы стилей назвали каскадными потому, что в одном документе можно использовать несколько таблиц, а браузер выстроит их каскадом, в соответствии с приоритетами использования.*

**Синтаксис CSS**

Документ CSS – этот отдельный документ, который создается с помощью текстового редактора и сохраняется как стандартный текстовый файл, с расширением файла **.css** (например styles.css).

Таблицы стилей строятся в соответствии с определенным порядком (синтаксисом), в противном случае они не могут нормально работать. Таблицы стилей составляются из определенных частей и имеют следующий синтаксис:

**

- **селектор** (Selector) — это элемент, к которому будут применяться назначаемые стили. Это может быть тег, класс или идентификатор объекта гипертекстового документа;

- **свойство** (Property) определяет одну или несколько характеристик селектора. Свойства задают формат отображения селектора: отступы, шрифты, выравнивание, размеры и т.д.;

- **значения** (Value) — это фактические числовые или строковые константы, определяющие свойство селектора;

- **описание** (Declaration) – совокупность свойств и их значений;

- **правило** (Rule) – полное описание стиля (селектор + описание).

*Таким образом, таблица стилей — это набор правил, задающих значения свойств селекторов, перечисленных в этой таблице. Общий синтаксис описания правила выглядит так:*

**селектор[, селектор[, ...]] {свойство: значение;}**

Регистр символов значения не имеет, порядок перечисления селекторов в таблице и свойств в определении не регламентирован.

*Обратите внимание: в таблице стилей теги HTML не заключаются в круглые скобки.*

Свойства CSS должны находиться в фигурных скобках. Для каждого тега HTML можно указать не одно, а несколько свойств стиля.

**Единицы измерения в CSS**

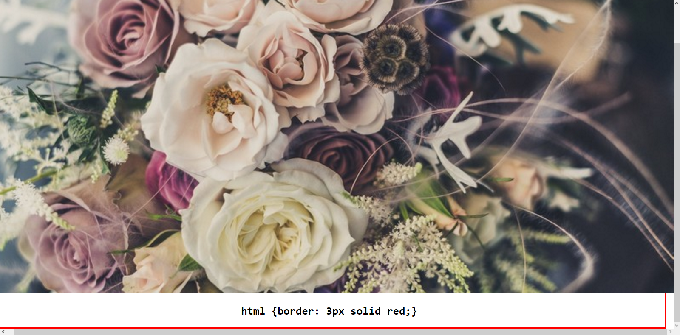
CSS поддерживает множество единиц измерений, включая абсолютные единицы измерения, такие как дюймы, сантиметры, пункты и т.д., а также относительные единицы измерения, такие как проценты, em и т.д. Эти значения необходимо использовать при указании различных измерений в ваших правилах стиля, например:

**border = "5px solid blue"**

Единица **rem** (происходит от **em**) ведет себя как **em**, за исключением того, что она всегда равна размеру основного шрифта по умолчанию **(font-size)**; унаследованные размеры шрифта на нее влияния не оказывают (как в случае с **em**).

В CSS3 были добавлены новые относительные единицы измерения, такие как **vh, vw, vmin, vmax**. Эти единицы вычисляются относительно размеров окна браузера. Для настольных компьютеров ширина окна браузера больше ширины области просмотра (добавляется ширина скроллбара), поэтому если для элемента установить ширину **100vw**, то он выйдет за пределы **html**.





|  |  |
| --- | --- |
| **Единица** | **Описание** |
| Vh | Эквивалентно 1% высоты окна браузера. |
| Vw | Эквивалентно 1% ширины окна браузера. |
| Vmin | Эквивалентно 1% меньшего размера окна браузера по высоте или ширине. |
| Vmax | Эквивалентно 1% большего размера окна браузера по высоте или ширине. |

Приёмы *отзывчивого веб-дизайна* базируются на использовании процентных значений. Но проценты далеко не лучшее решение для каждого случая, так как они вычисляются относительно размеров ближайшего родительского элемента. Поэтому, если вы хотите использовать высоту и ширину окна браузера, лучше воспользоваться единицами **vh** и **vw**. Например, если высота окна браузера равна **900px**, то **1vh** будет равен **9px**. Аналогично, если ширина окна браузера равна **1600px**, то **1vw** будет равен **16px**.

Так как ширина элемента, указанная с помощью **100vw** больше ширины области просмотра, то для создания полноэкранных фоновых изображений лучше использовать ширину **100%**, которая будет равна ширине корневого элемента **html**.

**.fullscreen-bg {**

**background: url(image.jpg);**

**background-position: center;**

**background-size: cover;**

**width: 100%;**

**height: 100vh;**

**}**

Чтобы сделать блок на всю высоту окна браузера, необходимо задать **height: 100%**; для трёх элементов — **html, body** и непосредственно для самого блока:

**html, body {**

**height: 100%;**

**}**

**section {**

**height: 100%;**

**}**

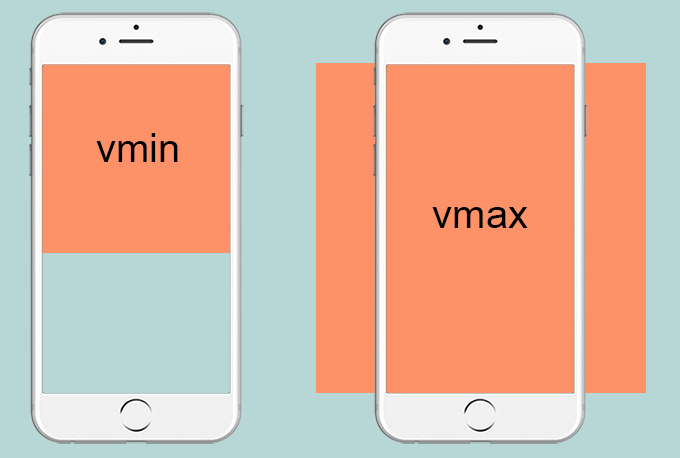
*Так как процентные размеры вычисляются относительно значений родительских элементов, то необходимо установить соответствующие значения для каждого элемента DOM. Единица измерения* ***vh*** *не требует установки значений по цепочке, так как её значение вычисляется напрямую относительно окна браузера:*

**section {**

**height: 100vh;**

**}**

*В то время, как единицы* ***vh*** *и* ***vw*** *всегда вычисляются относительно соответствующих размеров окна браузера,* ***vmin*** *и* ***vmax*** *определяются минимальным или максимальным значением высоты или ширины. Например, если ширина окна браузера равна* ***1200px****, а высота* ***700px****, то* ***vmin*** *будет равен* ***7px****, а* ***vmax*** *—* ***12px****.*

**

**Использование классов и идентификаторов в CSS (Виды селекторов)**

Итак, каскадная таблица стилей — это набор правил форматирования тегов HTML.

**Селекторы** представляют структуру веб-страницы. С их помощью создаются правила для форматирования элементов веб-страницы. Селекторами могут быть элементы (теги), их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

**1.** **Универсальный селектор с**оответствует любому HTML-элементу.

Например:

**\* {margin: 0;}**

обнулит внешние отступы для всех элементов сайта.

Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом:

**\*:after {CSS-стили}**

**\*:checked {CSS-стили}**

**2. Селектор элемента** – позволяет форматировать все элементы данного типа на всех страницах сайта.

Например:

**h1 {font-family: Lobster, cursive;}**

задаст общий стиль форматирования всех заголовков **h1**.

*Основной текст с выравниванием по ширине, абзацный отступ 30px, гарнитура (шрифт) — Serif, кегль (размер шрифта) — 14px:*

**p {**

**text-align: justify;**

**text-indent: 30px;**

**font-family: Serif;**

**font-size: 14px;**

**}**

*Следующее правило будет применено ко всем тегам <p>.*

**p {text-align: justify;**

**text-indent: 30px;**

**font-family: Serif;**

**font-size: 14px;**

**color: blue;}**

Одно и то же правило стиля можно применить сразу к нескольким различным тегам HTML-страницы.

*Синий цвет для заголовков с первого по третий уровень:*

**h1, h2, h3 {**

**color: blue; /\* тоже самое, что и #0000FF \*/**

**}**

*Таблицы и изображения выводить без обрамления:*

**table, img {border: none;}**

*Ссылки в элементах списков показывать без подчеркивания:*

**li a {text-decoration: none;}**

*Внутренние отступы слева и справа для блоков (<div>), заголовков таблиц и ячеек таблиц установить в 10px и залить фон желтым цветом:*

**div, th, td {**

**padding-left: 10px;**

**padding-right: 10px;**

**background-color: yellow;**

**}**

*Все ссылки в документе отображать черным цветом и полужирным шрифтом, а в основном тексте и списках — обычным, а также выделять их зеленым цветом и подчеркивать только при наведении курсора (в описании правил использован псевдоэлемент* ***a:hover****).*

**a {color: black; font-weight: bold;}**

**p a, li a {font-weight: normal; text-decoration: none;}**

**p a:hover, li a:hover {**

**color: #00FF00; text-decoration: underline;**

**}**

*Пример:*

**body,td,h1{**

**background:black;**

**color:white}**

*Стандарт CSS представляет возможности создания именованных стилей —* ***стилевых классов****. Это позволяет ответить на такой, например, вопрос: Как применить разные стили к одному и тому же селектору?*

**3. Селекторы класса** позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта.

Например, для создания заголовка с классом **headline** необходимо добавить атрибут **class** со значением **headline** в открывающий тег **<h1>** и задать стиль для указанного класса. Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.

**<h1 class="headline">Инструкция пользования персональным компьютером</h1>**

**.headline {**

**text-transform: uppercase;**

**color: lightblue;**

**}**

За стилевым классом в таблице стилей закрепляется имя. Это имя задается как обычное имя, но с точкой в качестве первого символа.

*Предположим, что в документе нужны два различных вида основного текста — один без отступа, второй — с левым отступом и шрифтом красного цвета. Для этого нужно создать правила для каждого из них, например так:*

**p {margin-left: 0;}**

**p.warn {margin-left: 40px; color: #FF00;}**

*Для применения созданного класса его имя нужно указать в атрибуте class для выбранных абзацев:*

**<p class=”warn”>Красный текст с отступом слева</p>**

*Общий синтаксис описания класса:*

**Селектор.имя\_класса {описание}**

*Если при создании класса селектор не указывать, тогда это правило можно применять к любому селектору, поддерживающему тот же набор свойств. Примеры:*

**Правило:**

**.solid\_blue {color: blue;}**

**Использование:**

**<p class=”solid\_blue”>Синий текст абзаца</p>**

**<li class=”solid\_blue”>Синий текст элемента списка</li>**

**Правило:**

**h1.bigsans {font-family: Sans; font-size: 1.5em;}**

**h1.smallserif {font-family: Serif; font-size: .84em;}**

**Использование:**

**<h1 class=”bigsans”>Большой, но рубленый</h1>**

**<h1 class=”smallserif”>Маленький, но с засечками</h1>**

**4. Селектор идентификатора** позволяет форматировать один конкретный элемент. Идентификатор **id** должен быть уникальным и на одной странице может встречаться только один раз.

Идентификатор элемента гипертекста указывается в атрибуте id. Для назначения стилей таким элементам используется синтаксис, аналогичный описанию классов, но вместо точки ставится знак # (“решетка”).

**<div id="sidebar"></div>**

**#sidebar {**

**width: 300px;**

**float: left;**

**}**

*Пример:*

**div#content {**

**position: absolute;**

**top: 10px;**

**left: 10%;**

**right: 10%;**

**border: solid 1px silver;**

**}**

**...**

**<div id="content">Текст</div>**

*Следует помнить, что идентификаторы элементов должны быть уникальны в пределах документа.*

**5. Селектор потомка** применяет стили к элементам, расположенным внутри элемента-контейнера.

Например:

**ul li {text-transform: uppercase;} — выберет все элементы** li**, являющиеся потомками всех элементов** ul**.**

Если нужно отформатировать потомки определенного элемента, этому элементу нужно задать стилевой класс:

**p.first a {color: green;} — данный стиль применится ко всем ссылкам, потомкам абзаца с классом first;**

**p .first a {color: green;} — если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого тега класса .first, который является потомком элемента <p>;**

**.first a {color: green;} — данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .first.**

**6. Дочерний селектор**. Дочерний элемент является прямым потомком содержащего его элемента. У одного элемента может быть несколько дочерних элементов, а родительский элемент у каждого элемента может быть только один. Дочерний селектор позволяет применить стили только если дочерний элемент идёт сразу за родительским элементом и между ними нет других элементов, то есть дочерний элемент больше ни во что не вложен.

Например:

**p > strong — выберет все элементы strong, являющиеся дочерними по отношению к элементу p.**

**7. Сестринский селектор.** Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня.

**h1 + p — выберет все первые абзацы, идущие непосредственно за любым тегом <h1>, не затрагивая остальные абзацы;**

**h1 ~ p — выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него.**

**8. Селектор атрибута** выбирает элементы на основе имени атрибута или значения атрибута:

**[атрибут] — все элементы, содержащие указанный атрибут**

**[alt] — все элементы, для которых задан атрибут alt;**

**селектор[атрибут] — элементы данного типа, содержащие указанный атрибут**

**img[alt] — только картинки, для которых задан атрибут alt;**

**селектор[атрибут="значение"] — элементы данного типа, содержащие указанный атрибут с конкретным значением**

**img[title="flower"] — все картинки, название которых содержит слово flower;**

**селектор[атрибут~="значение"] — элементы частично содержащие данное значение, например, если для элемента задано несколько классов через пробел**

**p[class~="feature"] — абзацы, имя класса которых содержит feature;**

**селектор[атрибут|="значение"] — элементы, список значений атрибута которых начинается с указанного слова**

**p[class|="feature"] — абзацы, имя класса которых feature или начинается на feature;**

**селектор[атрибут^="значение"] — элементы, значение атрибута которых начинается с указанного значения**

**a[href^="http://"] — все ссылки, начинающиеся на http://;**

**селектор[атрибут$="значение"] — элементы, значение атрибута которых заканчивается указанным значением**

**img[src$=".png"] — все картинки в формате png;**

**селектор[атрибут\*="значение"] — элементы, значение атрибута которых содержит в любом месте указанное слово**

**a[href\*="book"] — все ссылки, название которых содержит book.**

**9. Селектор псевдокласса**. Псевдоклассы – это классы, фактически не прикрепленные к HTML-тегам. Они позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу. Псевдоклассы характеризуют элементы со следующими свойствами:

**:link — не посещенная ссылка;**

**:visited — посещенная ссылка;**

**:hover — любой элемент, по которому проводят курсором мыши;**

**:focus — интерактивный элемент, к которому перешли с помощью клавиатуры или активировали посредством мыши;**

**:active — элемент, который был активизирован пользователем;**

**:valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;**

**:invalid — поля формы, содержимое которых не соответствует указанному типу данных;**

**:enabled — все активные поля форм;**

**:disabled — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;**

**:in-range — поля формы, значения которых находятся в заданном диапазоне;**

**:out-of-range — поля формы, значения которых не входят в установленный диапазон;**

**:lang() — элементы с текстом на указанном языке;**

**:not(селектор) — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — :not([type="submit"]);**

**:target — элемент с символом #, на который ссылаются в документе;**

**:checked — выделенные (выбранные пользователем) элементы формы.**

**10. Селектор структурных псевдоклассов**. Структурные псевдоклассы отбирают дочерние элементы в соответствии с параметром, указанным в круглых скобках:

**:nth-child(odd) — нечётные дочерние элементы;**

**:nth-child(even) — чётные дочерние элементы;**

**:nth-child(3n) — каждый третий элемент среди дочерних;**

**:nth-child(3n+2) — выбирает каждый третий элемент, начиная со второго дочернего элемента (+2);**

**:nth-child(n+2) — выбирает все элементы, начиная со второго;**

**:nth-child(3) — выбирает третий дочерний элемент;**

**:nth-last-child() — в списке дочерних элементов выбирает элемент с указанным местоположением, аналогично с :nth-child(), но начиная с последнего, в обратную сторону;**

**:first-child — позволяет оформить только самый первый дочерний элемент тега;**

**:last-child — позволяет форматировать последний дочерний элемент тега;**

**:only-child — выбирает элемент, являющийся единственным дочерним элементом;**

**:empty — выбирает элементы, у которых нет дочерних элементов;**

**:root — выбирает элемент, являющийся корневым в документе — элемент html.**

**11. Селектор структурных псевдоклассов** типа указывает на конкретный тип дочернего тега:

**:nth-of-type() — выбирает элементы по аналогии с :nth-child(), при этом берёт во внимание только тип элемента;**

**:first-of-type — выбирает первый дочерний элемент данного типа;**

**:last-of-type — выбирает последний элемент данного типа;**

**:nth-last-of-type() — выбирает элемент заданного типа в списке элементов в соответствии с указанным местоположением, начиная с конца;**

**:only-of-type — выбирает единственный элемент указанного типа среди дочерних элементов родительского элемента.**

**12. Селектор псевдоэлемента**. Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства **content**:

**:first-letter — выбирает первую букву каждого абзаца, применяется только к блочным элементам;**

**:first-line — выбирает первую строку текста элемента, применяется только к блочным элементам;**

**:before — вставляет генерируемое содержимое перед элементом;**

**:after — добавляет генерируемое содержимое после элемента.**

***Комбинация селекторов.*** *Для более точного отбора элементов для форматирования можно использовать комбинации селекторов:*

**a[href][title] — выберет все ссылки, для которых заданы атрибуты href и title;**

**img[alt\*="css"]:nth-of-type(even) — выберет все четные картинки, альтернативный текст которых содержит слово css.**

***Группировка селекторов.*** *Один и тот же стиль можно одновременно применить к нескольким элементам. Для этого необходимо в левой части объявления перечислить через запятую нужные селекторы:*

**h1,**

**h2,**

**p,**

**span {**

**color: tomato;**

**background: white;**

**}**

***Группировка свойств.*** *Группировка (grouping) состоит в объединении значений родственных свойств. При этом таблица стилей становится более компактной, но предъявляются более жесткие требования к описанию правил.*

*Ниже приведен пример обычного стиля, задающего отступы:*

**div {**

**margin-left: 10px;**

**margin-top: 5px;**

**margin-right: 40px;**

**margin-bottom: 15px;**

**}**

*Это же правило можно переписать с группировкой в следующем виде:*

**div {margin: 5px 40px 15px 10px;} /\*порядок: top right bottom left\*/**

*Оба стиля будут отображаться одинаково.*

Группировка может применяться для таких свойств, как padding, font, border, background и еще некоторых (см. документацию CSS).

При группировке всегда требуется правильный синтаксис выражения.

**Наследование и каскад**

*Наследование и каскад — два фундаментальных понятия в CSS, которые тесно связаны между собой.*

*Наследование заключается в том, что элементы наследуют свойства от своего родителя (элемента, их содержащего).*

*Каскад проявляется в том, как разные виды таблиц стилей применяются к документу, и как конфликтующие правила переопределяют друг друга.*

**1. Наследование**

Наследование является механизмом, с помощью которого определенные свойства передаются от предка к его потомкам. Спецификацией CSS предусмотрено наследование свойств, относящихся к текстовому содержимому страницы, таких как:

**color, font, letter-spacing, line-height, list-style, text-align, text-indent, text-transform, visibility, white-space, word-spacing**

*Во многих случаях это удобно, так как не нужно задавать размер шрифта и семейство шрифтов для каждого элемента веб-страницы.*

Свойства, относящиеся к форматированию блоков, не наследуются.

**background, border, display, float** и **clear, height** и **width, margin, min-max-height** и **-width, outline, overflow, padding, position, text-decoration, vertical-align** и **z-index.**

Принудительное наследование осуществляется с помощью ключевого слова **inherit**. Можно принудить элемент наследовать любое значение свойства родительского элемента. Это работает даже для тех свойств, которые не наследуются по умолчанию.

*CSS-стили задаются и работают следующим образом:*

*1) стили могут наследоваться от родительского элемента (наследуемые свойства или с помощью значения* ***inherit****);*

*2) стили, расположенные в таблице стилей ниже, отменяют стили, расположенные в таблице выше;*

*3) к одному элементу могут применяться стили из разных источников. Проверить, какие стили применяются, можно в режиме разработчика браузера. Для этого над элементом нужно щёлкнуть правой кнопкой мыши и выбрать пункт «Посмотреть код» (или что-то аналогичное). В правом столбце будут перечислены все свойства, которые заданы для этого элемента или наследуются от родительского элемента, а также файлы стилей, в которых они указаны, и порядковый номер строки кода.*

*4) при определении стиля можно использовать любую комбинацию селекторов — селектор элемента, псевдокласса элемента, класса или идентификатора элемента:*

**<div id="wrap" class="box clear"></div>**

**div {border: 1px solid #eee;}**

**#wrap {width: 500px;}**

**.box {float: left;}**

**.clear {clear: both;}**

Стили, которые используются на странице задаются внутри элемента **<style>**. Для задания стилей в HTML-документе используется язык CSS. Таких элементов на странице может быть несколько.

Для элемента доступны атрибуты **media, scoped, type**, а также [‎глобальные атрибуты](https://html5book.ru/html-attributes/).

<https://html5book.ru/osnovy-html/#part6>

<https://habr.com/post/86770/>

<https://learn.javascript.ru/css-format>

https://html5book.ru/osnovy-css/#part2

**Понятие и проблемы кроссбраузерности в CSS**

***https://web-standards.ru/articles/css-architecture/***

**CSS-правила**

***https://learn.javascript.ru/css-format***

**Использование в веб-страницах**

Существует три способа применения таблицы стилей к документу HTML:

**1.Встраивание (Inline).** Этот метод позволяет применить стиль к заданному тегу HTML.

**2. Внедрение (Embedded).** Внедрение позволяет управлять стилями страницы целиком.

**3. Связывание (Linked или External).** Связанная таблица стилей позволяет вынести описание стилей во внешний файл, ссылаясь на который можно контролировать отображение всех страниц сайта.

**1. Встроенные стили**

Встраивание стилей предоставляет максимальный контроль над всеми элементами веб-страницы. Встроенный стиль применяется к любому тегу HTML с помощью атрибута ***style*** следующим образом:

**<p style="font: 12pt Courier">Это текст с кеглем 12 точек и гарнитурой Courier</P>**

Mожно добавлять встроенный стиль в любой тег HTML, в котором эта операция будет иметь смысл. Среди таких тегов можно назвать абзацы, заголовки, горизонтальные линии, якоря и ячейки таблиц. Ко всем этим элементам логично применять встроенные стили.

Существуют два тега, которые помогают применять встроенные стили к разделам страницы. Это теги **<div>** (division - раздел) и **<span>** (промежуток).

Эти теги определяют диапазон текста, так что все, находящееся между ними, будет оформлено с помощью нужного стиля. Единственное различие между **<div>** и **<span>** состоит в том, что **<div>** создает принудительный разрыв строки, a **<span>** - нет.

*Следовательно, нужно использовать <span> для изменения стиля любой части текста, меньшей абзаца.*

*Пример:*

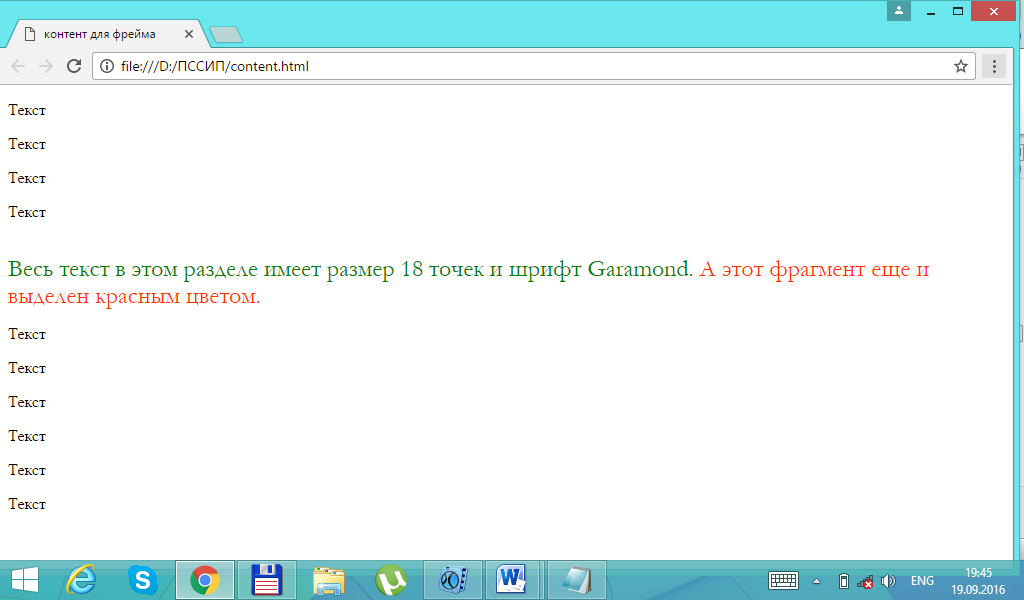
**<div style="font-family: Garamond; color:green; font-size:18pt">**

**Весь текст в этом разделе имеет размер 18 точек и шрифт Garamond.**

**<span style="color:#ff3300;">**

**А этот фрагмент еще и выделен красным цветом.</span>**

**</div>**



Встроенные стили полезны, когда необходима тонкая настройка отображения некоторого элемента страницы или небольшой веб-страницы.

**2. Внедренные стили**

Внедренные стили используют тег <style>, который обычно размещают в заголовке HTML-документа (<head>...</head>) и <bodу> в стандартном документе HTML:

**<html>**

**<head>**

**...**

**<style>**

**правила CSS**

**</style>**

**...**

**</head>**

**<body>**

**...**

*Пример:*

**<!DOCTYPE HTML>**

**<html>**

**<head>**

**<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">**

**<style type="text/css">**

**p {text-align: justify;**

**text-indent: 30px;**

**font-family: Serif;**

**font-size: 14px;**

**color: blue;}**

**p.warn {margin-left: 40px; color:red;}**

**a {text-decoration: none;}**

**</style>**

**<title>Ссылка в новом окне</title>**

**</head>**

**<body>**

**<p><a href="#top1">Ссылки</a> <a href="#top2">Якоря</a></p>**

**<p><a href="./site/1.html" title="Статья о Менделееве">Научная статья</a></p>**

**<p><h1><a href="./site/2.html">Как создать сайт?</a></h1></p>**

**<p><a href="./site/1.html" target="\_blank">Открыть в новом окне</a></p>**

**Использование графики в HTML**

**<p class="warn"> Возможность использования графики в HTML трудно переоценить.</p>**

**<p> Правильно подобранная и размещенная на Web-странице графика делает её</p> <p>визуально привлекательной и, что самое главное, передаёт одну из основных</p>**

**<p>идей документа. Для того чтобы вставить в Web-страницу изображение,</p> <p>необходимо либо нарисовать его, либо взять уже готовое. В любой программе</p>**

**<p> рисования можно создать простое изображение и сохранить его в нужном</p> <p>формате. Самыми распространенными графическими форматами в Web являются</p> <p>GIF,JPG и JPEG.</p>**

**<p>Теперь рассмотрим как вставить изображение в Web-страницу.</p>**

**<p>Для того чтобы вставить изображение в Web-страничку существует тег НТМL,</p>**

**<p> который заставляет браузер выводить изображение,им является тегIMG с</p> <p>обязательным атрибутом SRC (SouRCe, источник). Имя источника представляет</p>**

**<p> собой имя выводимого графического файла. Замыкающего тега не требуется.</p>**

**<p><h3>Пример вставки изображения</h3></p>**

**<p><center><img src="./site/picture.jpg" alt="Бабочка" title="Бабочка"> </center></p>**

**<A href="./site/picture.jpg"><img border="20" src="./site/picture.jpg" width="300" height="168" ALT="Использование графики в HTML"></A>**

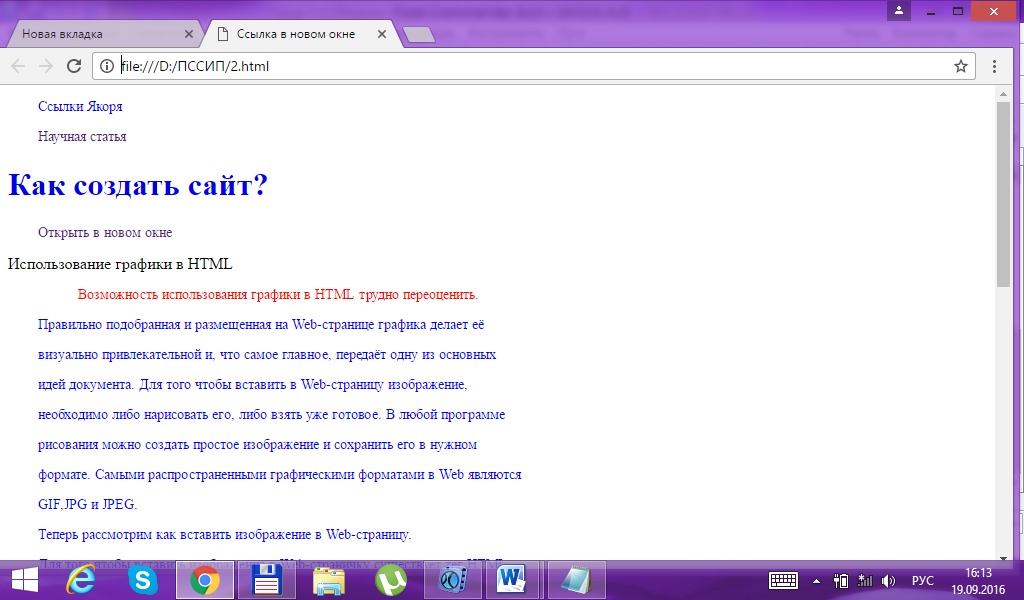
**<p><a name="top1"></a></p>**

**<p>Ссылки являются основой гипертекстовых документов и позволяют переходить с одной веб-страницы на другую. Особенность их состоит в том, что сама ссылка может вести не только на HTML-файлы, но и на файл любого типа, причем этот файл может размещаться совсем на другом сайте. Главное, чтобы к документу, на который делается ссылка, был доступ.</p>**

**<p><a name="top2"></a></p>**

**<hr align="center" color="#666666" width="800">**

**<p>Якорем называется закладка с уникальным именем на определенном месте веб-страницы, предназначенная для создания перехода к ней по ссылке. Якоря удобно применять в документах большого объема, чтобы можно было быстро переходить к нужному разделу.</p>**

**</body>**

**</html>**

*Пример:*

**<html>**

**<head>**

**<title>Блоки (div)</title>**

**<style> <!-- см. внедренные стили -->**

**body {margin: 10px;}**

**div {border: solid 1px black;}**

**.top {position: relative; height: 100px; width: 100%;}**

**.left {position: absolute; top: 114px; left: 10px; width: 200px; }**

**.main {position: absolute; top: 114px; left: 214px; margin-right:8px;}**

**</style>**

**</head>**

**<body>**

**<div class="top">TOP</div>**

**<div class="left">LEFT</div>**

**<div class="main">CONTENT</div>**

**</body>**

**</html>**

**3. Связанные таблицы стилей**

Связанные (linked), или внешние (external) таблицы стилей — наиболее удобное решение, когда речь идет об оформлении целого сайта, они, по сути, являются расширением понятия внедренных стилей. Используется тот же самый код, что и для внедренной таблицы стилей, только описание правил помещается в отдельный файл (обычно, но не обязательно, с расширением *.css*). С помощью тега <link> выполняется связывание этой таблицы стилей с каждой страницей, где ее необходимо применить, *например так:*

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

Любая страница, содержащая такую связь, будет оформлена в соответствии со стилями, указанными в файле sample.css. Следует отметить, что файл со стилями физически может находиться на другом веб-сервере, тогда в href нужно указать абсолютный путь к нему.

*Пример связанной таблицы стилей будет выглядеть следующим образом:*

*файл style-l.css, который находится в папке таблиц стилей style:*

**<style>**

**BODY{**

**background: #ffffcc;**

**color: #000000;**

**}**

**P {**

**font-family : sans-serif;**

**font-style : italic;**

**font-size : 16pt;**

**color: #006633;**

**}**

**H1{**

**font-family: Arial, Helvetica, sans-serif;**

**font-size: 24pt;**

**color: #996633;**

**}**

**</style>**

*С этим документом можно связать любое количество страниц HTML. Для этого нужно использовать между тегами </title> и </head> следующую конструкцию:*

**<html>**

**<head>**

**<title>Пример использования связанной таблицы стилей</title>**

**<link href="style/style-1.css" rel=stylesheet type="text/css">**

**</head>**

**<body>**

**<h1>Старинная студенческая песня</h1>**

**<P>**

**Поднявши меч на наш союз <br>**

**достоин будет худшей кары,<br>**

**и я за жизнь его тогда<br>**

**не дам и ломанной гитары.<br>**

**Как вожделенно жаждет век<br>**

**нащупать брешь у нас в цепочке...<br>**

**Возьмемся за руки друзья,<br>**

**чтоб не пропасть поодиночке.<br>**

**</P>**

**<p>**

**Среди совсем чужих пиров<br>**

**и слишком ненадежных истин,<br>**

**не дожидаясь похвалы, <br>**

**мы перья белые почистим.<br>**

**Пока безумный наш султан<br>**

**сулит дорогу нам к острогу,<br>**

**возьмемся за руки друзья,<br>**

**возьмемся за руки, ей-богу.<br>**

**</p>**

**<p>**

**Когда ж придет дележки час,<br>**

**не нас калач ржаной поманит,<br>**

**и рай настанет не для нас,<br>**

**зато Офелия помянет.<br>**

**</p>**

**<p>**

**Пока ж не грянула пора<br>**

**нам отправляться понемногу,<br>**

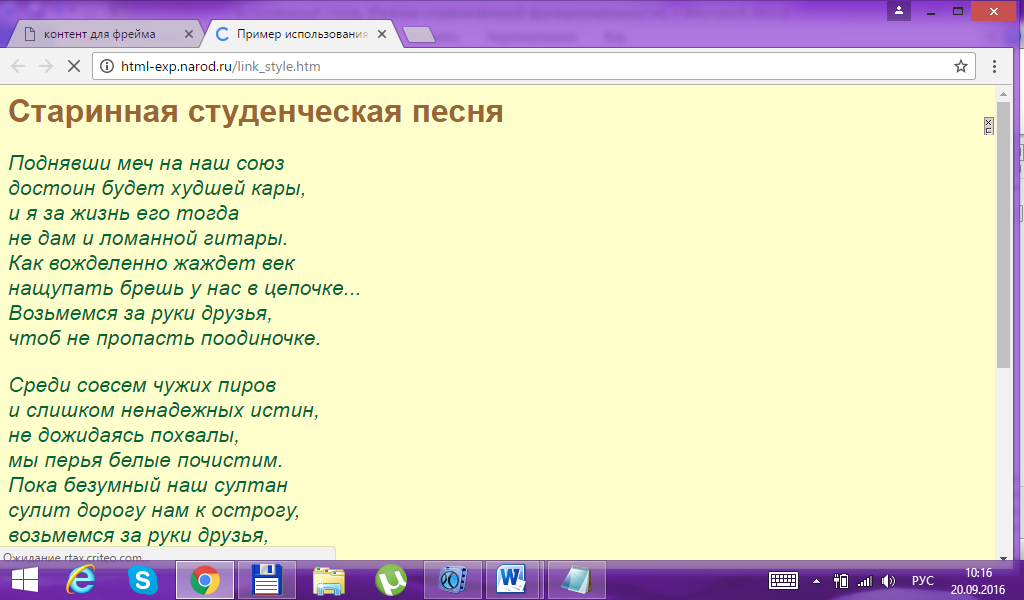
**возьмемся за руки друзья,<br>**

**возьмемся за руки, ей-богу.<br>**

**</p>**

**Это стихотворение Булата Окуджавы, ставшее уже классикой.<br>**

**<a href="css.htm#метка\_51">Вернемся? </a>**

****

**</body>**

**</html>**

*Далее к коду был добавлен внедренный стиль.*

**<html>**

**<head>**

**<title>Пример использования связанной таблицы стилей</title>**

**<link href="style/style-1.css" rel=stylesheet type="text/css">**

**<style type="text/css">**

**p.warn {margin-left: 40px; color:red;}**

**</style>**

**</head>**

**<body>**

**<h1>Старинная студенческая песня</h1>**

**<P>**

**Поднявши меч на наш союз <br>**

**достоин будет худшей кары,<br>**

**и я за жизнь его тогда<br>**

**не дам и ломанной гитары.<br>**

**Как вожделенно жаждет век<br>**

**нащупать брешь у нас в цепочке...<br></P>**

**<p class="warn"> Возьмемся за руки друзья, <br></p>**

**<P> чтоб не пропасть поодиночке.<br>**

**</P>**

**<p>**

**Среди совсем чужих пиров<br>**

**и слишком ненадежных истин,<br>**

**не дожидаясь похвалы, <br>**

**мы перья белые почистим.<br>**

**Пока безумный наш султан<br>**

**сулит дорогу нам к острогу,<br>**

**возьмемся за руки друзья,<br>**

**возьмемся за руки, ей-богу.<br>**

**</p>**

**<p>**

**Когда ж придет дележки час,<br>**

**не нас калач ржаной поманит,<br>**

**и рай настанет не для нас,<br>**

**зато Офелия помянет.<br>**

**</p>**

**<p>**

**Пока ж не грянула пора<br>**

**нам отправляться понемногу,<br>**

**возьмемся за руки друзья,<br>**

**возьмемся за руки, ей-богу.<br>**

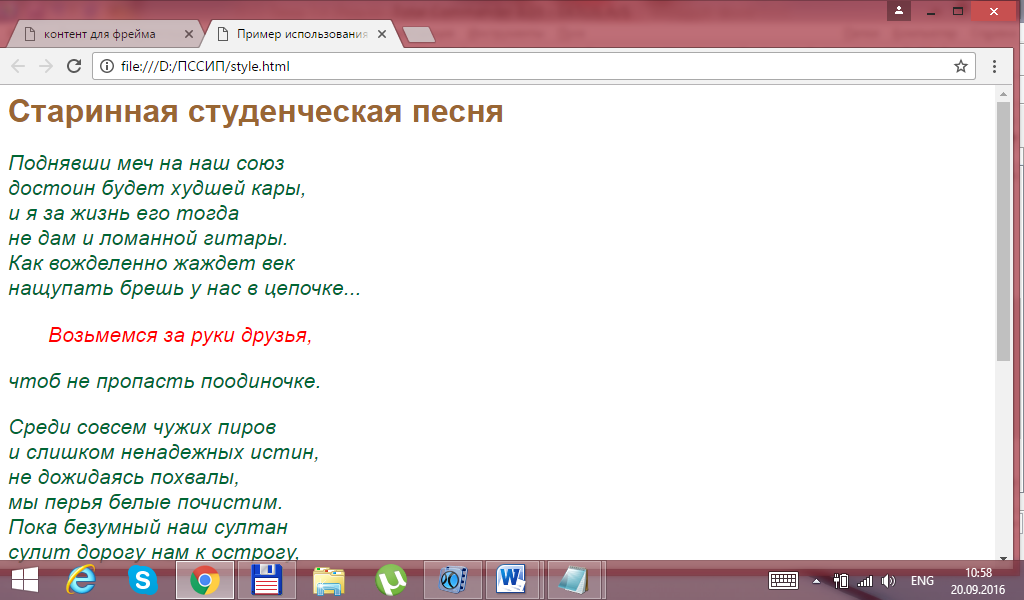
**</p>**

**Это стихотворение Булата Окуджавы, ставшее уже классикой.<br>**

**<a href="css.htm#метка\_51">Вернемся? </a>**

**</body>**

**</html>**



*Встроенный стиль применяется поверх связанного и внедренного стилей.*

*Обязательно просматривайте страницы с таблицами стилей в различных браузерах. Это связано с тем, что разные браузеры могут по разному интерпретировать одно и то же правило, а некоторые свойства и/или значения и вовсе не поддерживать. Следует также тестировать страницы с отключенными стилями (например, в текстовых браузерах), чтобы убедиться, что страница читабельна.*

*Если вам нужна сотня-другая-третья страниц HTML — используйте внешнюю, глобальную, таблицу стилей. Если некоторые из этих страниц требуют корректировки общего оформления — используйте внедренный стиль. А если на странице нужно явно изменить оформление одного-двух элементов, то применяйте встроенные стили. Именно* **в таком порядке происходит перекрытие стилей при каскадировании, схематично это можно представить так: *связанные стили -> внедренные стили -> встроенные стили***

**Аппаратно-зависимые стили**

Таблицы стилей могут применяться для управления отображением содержимого в зависимости от используемого устройства вывода (монитор, проектор, устройство печати, звуковой синтезатор и т.п.). Для этого в описание стилей включить тип устройства, например так:

**@media print {/\* печатающее устройство \*/**

**BODY { font-size: 10pt; }**

**}**

**@media screen { /\* монитор \*/**

**BODY { font-size: 12pt; }**

**}**

**@media screen, print {**

**BODY { line-height: 1.2; }**

**}**

**@media all {**

**BODY { margin: 1pt; }**

**}**

*Как видно из примера, вся таблица разбивается на секции, каждая из которых начинается со слова* ***@media****, за которым следует название класса устройств и далее, в фигурных скобках, непосредственно описание стилей.*

*Можно разделить таблицы стилей иначе, указав тип устройства в теге <link>:*

**<link rel=stylesheet href="sample.css" type="text/css" media=”screen”>**

**Свойства CSS**

В таблице перечислены некоторые часто используемые свойства элементов CSS и их назначение.

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значения** | **Описание** |
| background | [background-color || background-image || background-repeat || background-attachment || background-position] | inherit | Управление фоном элемента |
| background-color | <color> | transparent | inherit | Цвет фона |
| background-image | <uri> | none | inherit | Фоновое изображение |
| background-position | [ [<percentage> | <length> ]{1,2} | [ [top | center | bottom] || [left | center | right] ] ] | inherit | Положение фо новой картинки |
| background-repeat | repeat | repeat-x | repeat-y | no-repeat | inherit | Повторение фоновой картинки |
| Border | [ border-width || border-style || <color> ] | inherit | Границы элемента |
| border-collapse | collapse | separate | inherit | Объединение/разделение смежных границ |
| border-color | <color>{1,4} | transparent | inherit | Цвет границы |
| border-style | <border-style>{1,4} | inherit | Стиль линии границы |
| border-top border-right border-bottom border-left | [ border-top-width || border-style || <color> ] | inherit | Управление стилем заданной границы |
| border-width | <border-width>{1,4} | inherit | Толщина линии границы |
| Bottom | <length> | <percentage> | auto | inherit | Низ элемента |
| Clear | none | left | right | both | inherit | Запрет заполнения свободного пространства рядом с элементом |
| Clip | <shape> | auto | inherit | Обрезка содержимого элемента |
| Color | <color> | inherit | Цвет содержимого |
| Cursor | [ [<uri> ,]\* [ auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize| text | wait | help ] ] | inherit | Форма курсора |
| Display | inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit | Способ отображения элемента |
| empty-cells | show | hide | inherit | Отображение пустых ячеек таблицы |
| Float | left | right | none | inherit | Свободное размещение элемента |
| Font | [ [ font-style || font-variant || font-weight ]? font-size [ / line-height ]? font-family ] | caption | icon | menu | message-box | small-caption | status-bar | inherit | Управление шрифтом |
| font-family | [[ <family-name> | <generic-family> ],]\* [<family-name> | <generic-family>] | inherit | Гарнитура |
| font-size | <absolute-size> | <relative-size> | <length> | <percentage> | inherit | Кегль |
| font-style | normal | italic | oblique | inherit | Стиль шрифта |
| font-variant | normal | small-caps | inherit | Варианты отображения шрифта |
| font-weight | normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit | Толщина шрифта |
| Height | <length> | <percentage> | auto | inherit | Ширина элемента |
| Left | <length> | <percentage> | auto | inherit | Положение левой границы элемента |
| line-height | normal | <number> | <length> | <percentage> | inherit | Высота строки |
| list-style | [ list-style-type || list-style-position || list-style-image ] | inherit | Стиль списка |
| Margin | <margin-width>{1,4} | inherit | Внешний отступ |
| margin-top margin-right margin-bottom margin-left | <margin-width> | inherit | Внешний отступ по заданной стороне |
| Padding | <padding-width>{1,4} | inherit | Внутренний отступ |
| padding-top padding-right padding-bottom padding-left | <padding-width> | inherit | Внутренний отступ по заданной стороне |
| Position | static | relative | absolute | fixed | inherit | Позиционирование элемента |
| Right | <length> | <percentage> | auto | inherit | Положение правой границы |
| text-align | left | right | center | justify | <string> | inherit | Выравнивание текстового блока |
| text-decoration | none | [ underline || overline || line-through || blink ] | inherit | Текстовые эффекты |
| text-indent | <length> | <percentage> | inherit | Абзацный отступ |
| text-transform | capitalize | uppercase | lowercase | none | inherit | Начертание текста |
| Top | <length> | <percentage> | auto | inherit | Положение верхней границы элемента |
| vertical-align | baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage> | <length> | inherit | Вертикальное выравнивание в пределах блока |
| Visibility | visible | hidden | collapse | inherit | Управление видимостью элемента |
| white-space | normal | pre | nowrap | inherit | Управление пробелами между словами |
| Width | <length> | <percentage> | auto | inherit | Ширина элемента |
| z-index | auto | <integer> | inherit | Порядок перехода по клавише Tab |

**Псевдостили**

Для управления отображением гиперссылок используются так называемые псевдостили. Они позволяют определить вид непосещенной ссылки (a:link); ссылки, по которой пользователь уже совершал переход (a:visited), а так же вида ссылки, над которой находится курсор (a:hover) и активной ссылки (a:active).

*Пример:*

**a:link {**

**font-family: Geneva, Arial, Helvetica, sans-serif;**

**color: #CC0000;**

**text-decoration: underline;**

**background-color: #CCFFFF;**

**}**

**a:hover {**

**font-family: "Times New Roman", Times, serif;**

**font-size: 22px;**

**font-style: italic;**

**font-weight: bold;**

**color: #FFFF00;**

**text-decoration: blink;**

**}**

**a:visited {**

**color: #660000;**

**}**

**a:active {**

**font-size: 18px;**

**color: #000099;**

**text-decoration: blink;**

**}**

**Поддержка шрифтов в таблицах стилей**

*Один из наиболее привлекательных аспектов таблиц стиля - это возможность применения различных шрифтов к определенной странице без необходимости использовать многочисленные теги <font>.*

Таблицы стилей позволяют выбирать ряд шрифтов и применять их к конкретным разделам страницы типа номера заголовка, абзаца или другого фрагмента. Однако вместо стандартного тега HTML <font> используется атрибут таблицы стилей ***font-family***. В этот тег можно добавлять многие атрибуты либо использовать классы и группировку, чтобы реализовать всю мощь средств работы со шрифтами с помощью таблиц стилей.

*Практическая реализация поддержки шрифтов в таблицах стилей аналогична проблемам, с которыми пользователи сталкиваются при использовании тегов****<font>****и соответствующих атрибутов. Конкретный шрифт должен быть установлен на том компьютере, где происходит просмотр страницы. Как и тег****<font>****, таблицы стилей позволяют указывать любое число шрифтов, чтобы увеличить вероятность того, что браузер клиента подберет нужный шрифт из списка. Например, если у них не будет шрифта Century Schoolbook, то, возможно, найдется Garamond и т. д. Хотя сами эти шрифты различаются довольно значительно, их семейства достаточно похожи, чтобы считать, что они выполняют сходные дизайнерские задачи.*

Таблицы стилей распознают пять семейств шрифтов: *Serif* (С засечками) , *Sans Serif* (Рубленые),*Script*(Рукописные), *Monospace* (Моноширинные), *Decorative* (Декоративные).

Можно указывать несколько названий шрифтов в любом теге HTML, используя встроенные, внедренные и связанные таблицы стилей.

*Например, на этой странице используется внедренная таблица стилей, и для заголовка первого уровня задан:*

**H1{font-family:Arial,Halvetica,Verdana,sans-serif;**

**font-weight:bold; color:#333399;}**

*Если браузер не сможет найти на компьютере клиента первые три шрифта, он поставит вместо них первый из доступных рубленных шрифтов семейства sans-serif и будет его использовать.*

Задание свойств и значений шрифта

*У шрифтов есть очень много свойств, которые можно изменять, и очень много значений, которые могут принимать эти свойства.*

*Как и для стандартных шрифтов HTML, можно задавать свойства для управления размером и цветом. Однако для шрифтов HTML нельзя изменять насыщенность и тип шрифта, а также высоту строки или интерлиньяж (расстояние между отдельными строками текста). Кроме того, доступные в таблицах стилей методы управления размером шрифта намного превосходят уровень, предлагаемый стандартами HTML.*

Цвет

Таблицы стилей используют стандартные методы работы с цветами браузера.

*Другими словами, для получения оптимальных результатов используйте шестнадцатеричный (и лучше всего, поддерживаемый браузерами) цвет. Можно добавлять цвет, как и другие свойства стиля, в любой подходящий тег HTML для встроенных, внедренных и связанных стилей.*

*Например, при описании цвета заголовка первого уровня указано значения****color:#333399****, в результате мы видим заголовок "Каскадные таблицы стилей" темно-синего цвета.*

Насыщенность

Насыщенность (weight) показывает толщину шрифта.

*Например, для гарнитуры Arial существуют такие разновидности: жирный (black), полужирный (bold), светлый (light) и т. д.*

*Для шрифтов существуют различные начертания. Если вы не уверены на сто процентов, что на компьютере клиента установлен определенный шрифт, разумнее будет применять начертание, доступное для всех шрифтов. Существует только одна (не считая нормальной) стандартная насыщенность, которая доступна для большинства шрифтов. Это - полужирное начертание!*

*Вот пример применения встроенного начертания:*

**<p style="font-family: arial, helvetica, sans-serif; font-weight: bold;**

**color=#cc99ff;">**

**We are an intelligent species and the use of our**

**intelligence quite properly gives us pleasure. In this respect the**

**brain is like a muscle. When it is in use we feel very good.**

**Understanding is joyous.**

**</p >**

*Полужирное начертание применяется к абзацу.*

Размер

Размер шрифта в таблицах стилей можно определять, используя пункты ( points), пиксели (pixels), дюймы (inches), сантиметры (centimeters), миллиметры (millimeters) и пики (picas), а также используя [новые единицы измерения.](http://html-exp.narod.ru/css.htm#метка_41)

*Для Web-дизайнеров естественно выбирать пункты или пиксели, хотя все зависит от вкусов. Однако, использование единиц, отличных от пунктов, может вызывать серьезные проблемы. Если вместо пунктов используются пиксели, то хотя текст, размер которого задан в пикселях, будет виден в броузерах с поддержкой таблиц стилей (типа Internet Explorer 3.0 и выше или Netscape 4.0 и выше), он может не выводиться на печать!*

Другие параметры шрифта

***Text-decoration.*** Этот параметр полезен для отключения подчеркивания ссылок. Для этого нужно просто задать для параметра ***text-decoration*** значение ***none*** (отсутствует). Другие возможные значения -***underline*** (подчеркивание), ***italic*** (курсив) и ***strikethrough*** (перечеркивание).

***Line-height.*** Это то же самое, что интерлиньяж (leading). Данный параметр определяет высоту каждой строки текста - по существу, расстояние между строками.

***Background.*** Этот параметр помещает текст на цвет или изображение. Здесь используются атрибуты ***color***или ***url***(адрес), где адрес указывает расположение файла изображения.

*Внимание! Этот параметр может быть назначен не только тегу****<body>****, но и любому тегу или фрагменту текста, чтобы выделить нужную область на странице.*

**Краткий обзор способов размещения элементов страницы**

Таблицы стилей могут помогать в создании макета страницы, предоставляя широкие возможности выравнивания элементов страницы и управления полями.

Выравнивание текста выполняется с помощью свойства **text-align**. Возможные значения **left**, **right**, **center** и **justify**.

Для управления полями Web-страницы используются четыре свойства: **margin-top**, **margin-bottom**, **margin-left** и **margin-right**.

Как и все свойства CSS, они могут применяться к любому логическому объекту. Обычно для управления полями используются объекты BODY и Р (paragraph - абзац). Для указания значений полей можно использовать разные единицы измерения: пиксели, пункты, сантиметры, дюймы и проценты.

*Для справки: стандартные макеты страниц HTML обычно измеряются в пикселях. CSS разрешают использовать отрицательные значения для полей. Это дает возможность дизайнеру создавать перекрывающиеся области страницы, добиваясь интересного эффекта.*

**Пример создания 3-D эффекта**

*Рассмотрим пример, создающий эффект, который без задания стиля можно было бы достичь только с помощью графики. В процессе создания Web-страницы наложим один текст на другой. Эта возможность частичного перекрытия текста часто используется в дизайне реальных страниц. Сначала*[*посмотрим*](http://html-exp.narod.ru/ex_css_1.htm)*на результат, а затем обсудим как он получен.*

*HTML-код этой страницы выглядит так:*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">**

**<html>**

**<head>**

**<META http-equiv="Content-Type"content="text/html; charset=windows-1251">**

**<META http-equiv="Content-Language" content="ru">**

**<title>Пример создания 3-D эффект с помощью CSS</title>**

**<STYLE type=text/css>BODY {color: #000080; font-size : 16px;**

**font-family : Arial, Helvetica, sans-serif; }**

**.тень{**

**color: #DBDBDB;**

**text-align : right;**

**font : lighter "Times New Roman", Times, serif;**

**margin-top : 30px;**

**font-size : 270px;**

**line-height : 270px;**

**}**

**.основа{**

**color: red;**

**font : 900 220px "Times New Roman", Times, serif;**

**margin-top : -230px;**

**line-height : 250px;**

**}**

**.слой1{**

**color: black;**

**margin-top : -130px;**

**font-weight : normal;**

**font-size : 65px;**

**line-height : 65px;**

**font-family : Arial, Helvetica, sans-serif;**

**}**

**.слой2{**

**color: green;**

**margin-top : 30px;**

**font-weight : normal;**

**font-size : 35px;**

**line-height : 45px;**

**font-family : Arial, Helvetica, sans-serif;**

**}**

**</STYLE>**

**</head>**

**<body>**

**<p>Пример наложения текстов друг на друга, реализованный с помощью тега**

**<div> и CSS. В результате получаем 3-D эффект.**

**</p>**

**<center>**

**<table width="500" border="0" cellspacing="0" cellpadding="0">**

**<tr>**

**<td align="center" valign="top">**

**<div class="тень">Мы</div>**

**<div class="основа">Мы</div>**

**<div class="слой1">сделали это без всякой графики</div>**

**<div class="слой2">используя только каскадные таблицы стилей</div>**

**</td>**

**</tr>**

**</table>**

**</center>**

**<br>**

**</body>**

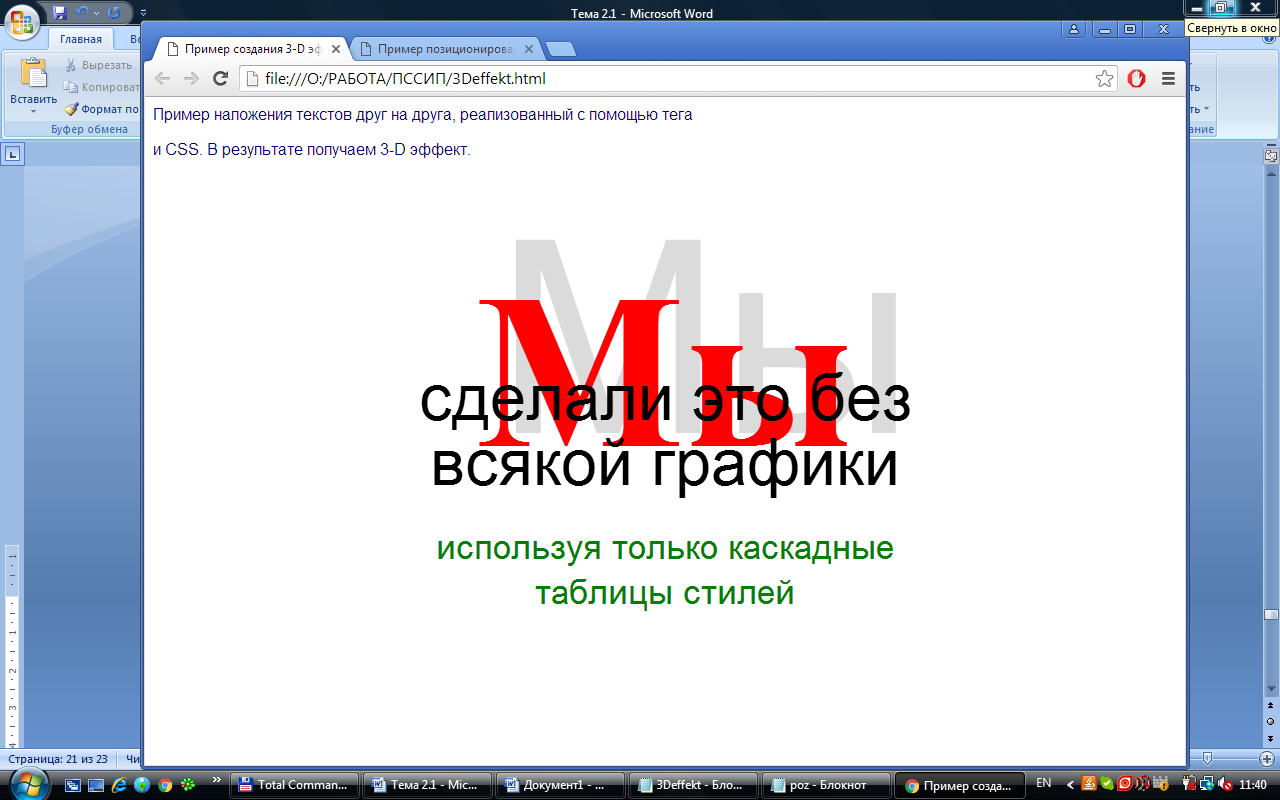
**</html>**

*В данном примере тег <div> применяется для выделения фрагмента HTML-документа. Он ничего не форматирует, а лишь помечает фрагмент текста, который рассматривается как единый объект.*

*Атрибут class этого тега позволяет сослаться на таблицу стилей, и таким образом задать стиль представления текста, расположенного между тегами <div class> и </div>.*

*В этом примере тексты определяются как бы в слоях, которые накладываются друг на друга. Сначала выводится слой****тень****, на него накладывается слой****основа****, а затем****слой1****и****слой2****. Порядок, в котором слои накладываются друг на друга, задается порядком следования фрагментов текста, помеченных тегом <div>.*

*Перекрытие слоев обеспечивается применением отрицательных значений свойста****margin-top****(отступ сверху).*

**

*Запомните - используя отрицательные значения свойста****margin-top****, можно наложить один текст на другой.*

**Позиционирование элементов**

*Рассмотрим вопрос позиционирования элементов на странице.*

*Мы уже рассматривали вопрос о расположении элементов на странице путем использования*[*фреймов*](http://html-exp.narod.ru/frame.htm)*и* [*таблиц*](http://html-exp.narod.ru/table.htm#метка_3)*. Кроме того, для перекрытия элементов приходилось применять особые ухищрения, как в*[*примере создания 3-D эффекта*](http://html-exp.narod.ru/css.htm#метка_9)*.*

*Вместе с тем, существуют специальные свойства для позиционирования, применяя которые можно легко установить элементы в нужные места страницы. При этом элементы могут накладываться друг на друга.*

Для позиционирования элементов на странице используются три свойства:

**left** - для задания расстояния в пикселах от левого края окна (х-координата);

**top** - для задания расстояния в пикселах от верхнего края окна (у-координата);

**z-index** - для указания порядка, в котором элементы будут перекрывать друг друга; это новое измерение, элементы с большим z-индексом будут появляться над элементами с меньшим z-индексом.

*Конечно, при использовании этих трех свойств не создается эффект трехмерного пространства, но это уже нечто большее, чем двухмерная плоскость.*

Кроме свойств-координат, для позиционирования элементов понадобится свойство **position**. Это свойство в сочетании со свойствами left и top позволяет устанавливать элементы в определенные позиции окна.

Свойство position может принимать три значения:

**absolute** - заданные свойства left и top поставят элемент в место с координатами х и у относительно верхнего левого угла контейнера (объекта, содержащего данный элемент). Если они определены для элемента вне контейнера, то началом отсчета координат будет верхний левый угол страницы.

*Заметим, что положение элемента не зависит от положения его тэга внутри HTML-документа.*

**relative** - элемент будет установлен в соответствии с тем, в каком месте исходного текста он находится; это значение установлено по умолчанию.

*Например, если элемент находится в трех строках от начала его выделения в тексте документа, то по умолчанию считается, что свойство позиционирования имеет значение relative, а свойства координат leftи top - нулевые значения.   
Ненулевые значения свойств left и top осуществляют сдвиг элемента относительно исходного положения.*

**static** - элемент будет поставлен в некоторое положение относительно фона и не будет перемещаться при прокручивании страницы.

*П[ример](http://html-exp.narod.ru/ex_css_3.htm) помещения текста на определенное место на странице и назначение ему красного цвета:*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">**

**<html>**

**<head>**

**<META http-equiv="Content-Type"content="text/html; charset=windows-1251">**

**<META http-equiv="Content-Language" content="ru">**

**<title>Пример позиционирования текста</title>**

**</head>**

**<body>**

**<H1 STYLE="position:absolute;**

**top:120;left:100;width:400;height:100;color:red">**

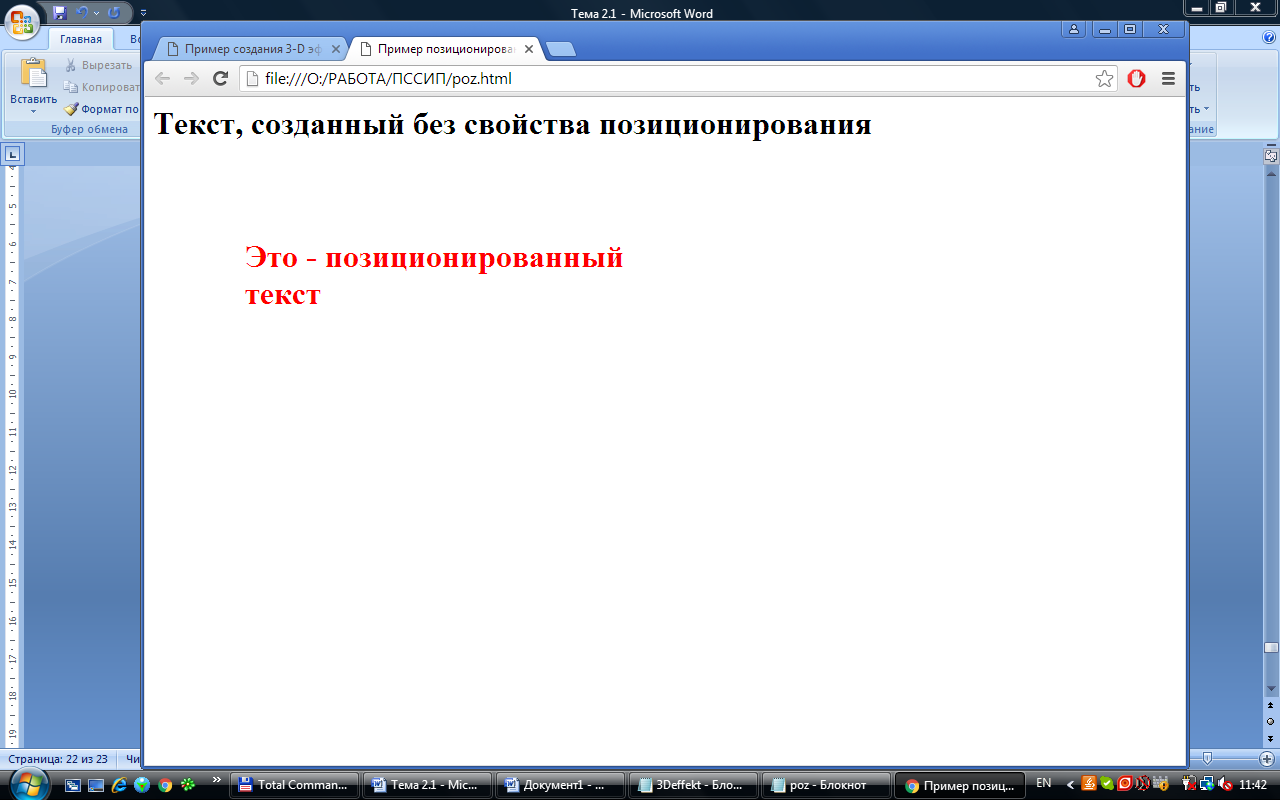
**Это - позиционированный текст**

**</H1>**

**<h1>Текст, созданный без свойства позиционирования</h1>**

**</body>**

**</html>**

**

*Верхний текст не позиционирован, а нижний - позиционирован.*

*Дело в том, что в первом случае использования тега <H1> элемент позиционировался индивидуально с помощью атрибута STYLE, а во втором случае специального позиционирования не осуществлялось и броузер расположил элемент <H1> так, как он это должен был сделать по умолчанию.*

**Статические фильтры**

*Рассматриваемые вопросы имеют чётко выраженную специфику: они относятся только к одному браузеру, браузеру фирмы Microsoft Internet Explorer.*

**Фильтр** - это некоторый алгоритм, преобразующий визуальное отображение элемента в окне браузера *(изменения внешнего вида текста и графики на странице)*.

*Иначе, фильтр - это трансформация исходного изображения по определенным правилам.*

Существуют статические и динамические фильтры:

- статические фильтры просто изменяют внешний вид элемента до его отображения;

- динамические фильтры воздействуют во времени на визуальное отображение элемента, меняя его непосредственно на HTML-странице, что приводит к эффекту анимации. Динамический фильтр еще называют переходом из одного состояния отображения в другое. Позволяют изменить графический элемент со скоростью, задаваемой пользователем. Работа динамических фильтров основана на [сценариях](http://html-exp.narod.ru/dhtml.htm#метка_20).

Фильтры применяются не ко всем HTML-элементам, а только к тем, которые могут определять правильный прямоугольный блок при отображении в окне браузера и не являются окнами, как, например, элемент <IFRAME>.

Элементы, к которым можно применять фильтры: BODY, BUTTON, DIV (если заданы ширина (свойство width), высота (свойство height) или элемент абсолютно позицирован), IMG, INPUT, MARQUEE, SPAN (если заданы ширина (свойство width), высота (свойство height) или элемент абсолютно позицирован), TABLE, TD, TEXTAREA, TH.

Статический фильтр задается как свойство в таблице стилей.

**filter: название\_фильтра (параметр1,параметр2,...,параметр№)**

**<Element STYLE="filter:filtername(sProperties)">**

**Element** **– имя элемента разметки, например, DIV, SPAN , TABLE , IMG;**

**filtername – наименование фильтра, например, alpha, blur, chroma;**

**sProperties – строка параметров фильтра, формат которой таков:**

**duration=5, transition=12;**

**freq=1, phase=0, strength=1;**

**color=Red, offX=5, offY=-5, positive=1.**

**filter: progid:DXImageTransform.Microsoft.Фильтр(свойства)**

Параметры не обязательны.

Статические фильтры (без параметров):

[**alpha**](http://htmlbook.ru/css/filter/alpha) - настраивает прозрачность объекта;

[**basicImage**](http://htmlbook.ru/css/filter/basicimage) - устанавливает параметры цвета, поворота изображения или прозрачности;

[**blur**](http://htmlbook.ru/css/filter/blur) - размывает содержимое;

[**hroma**](http://htmlbook.ru/css/filter/chroma) - показывает определенные цвета прозрачными;

[**dropShadow**](http://htmlbook.ru/css/filter/dropshadow) - отображает тень;

[**emboss**](http://htmlbook.ru/css/filter/emboss) - показывает содержимое объекта в виде барельефа;

[**engrave**](http://htmlbook.ru/css/filter/engrave) - показывает содержимое объекта в виде черно-белой гравюры;

**fliph** - горизонтальное отражение картинки или текста;

**flipv** - вертикальное отражение картинки или текста;

[**glow**](http://htmlbook.ru/css/filter/glow)- добавляет свечение вокруг краев;

[**gradient**](http://htmlbook.ru/css/filter/gradient) - создаёт линейный градиент;

[**ICMFilter**](http://htmlbook.ru/css/filter/icmfilter) - преобразует цвета содержимого на основе профиля системы управления цветом (Image Color Management, ICM);

[**light**](http://htmlbook.ru/css/filter/light) - создает эффект лучей света;

[**maskFilter**](http://htmlbook.ru/css/filter/maskfilter) - показывает прозрачные пикселы как цветную маску, а непрозрачные пикселы наоборот, прозрачными.

[**matrix**](http://htmlbook.ru/css/filter/matrix) - изменяет размер, поворачивает или отражает объект на основе матричных преобразований;

[**motionBlur**](http://htmlbook.ru/css/filter/motionblur) - размывает объект так, словно он быстро движется.

[**shadow**](http://htmlbook.ru/css/filter/shadow)- добавляет тень;

[**wave**](http://htmlbook.ru/css/filter/wave) - вносит волнообразные искажения;

**xray** - изображение только контура объекта.

*[Пример использования статического фильтра](http://html-exp.narod.ru/ex_css_61.htm" \t "_self):*

**<HTML>**

**<HEAD><TITLE>Filter: blur</TITLE>**

**<STYLE>**

**<!--**

**BODY {**

**background-color: lightyellow;**

**}**

**TABLE {**

**font-family: serif;**

**font-size: 10mm;**

**color: blue;**

**background-color: LavenderBlush;**

**margin: 10px;**

**padding: 10px;**

**border-style: solid;**

**border-color: Brown;**

**border-width: 1px;**

**text-align: left;**

**}**

**-->**

**</STYLE>**

**<BODY>**

**<CENTER>**

**<TABLE>**

**<TR><TD STYLE="position: relative; left: 0; top: 0;**

**filter: blur(add=100, direction=5, strength= 10);">**

**Представление о статических фильтрах можно получить,<BR>**

**рассмотрев действие такого фильтра как blur<BR>**

**(обеспечение размытости элемента в заранее заданных пределах).**

**</TD></TR>**

**</TABLE>**

**<BR>**

**<TABLE>**

**<TR><TD STYLE="position: relative; left: 0; top: 0;**

**filter: blur(add=0, direction=45, strength= 10);">**

**Представление о статических фильтрах можно получить,<BR>**

**рассмотрев действие такого фильтра как blur<BR>**

**(обеспечение размытости элемента в заранее заданных пределах).**

**</TD></TR>**

**</TABLE>**

**<BR>**

**<TABLE>**

**<TR><TD>**

**<img src="site/picture.jpg" border="0"**

**STYLE="position: relative; left: 0; top: 0;**

**filter: blur(add=10, direction=45, strength= 30);">**

**</TD></TR>**

**</TABLE>**

**<BR>**

**<TABLE>**

**<TR><TD>**

**<img src="site/picture.jpg" border="0"**

**STYLE="position: relative; left: 0; top: 0;**

**filter: blur(add=0, direction=45, strength= 30);">**

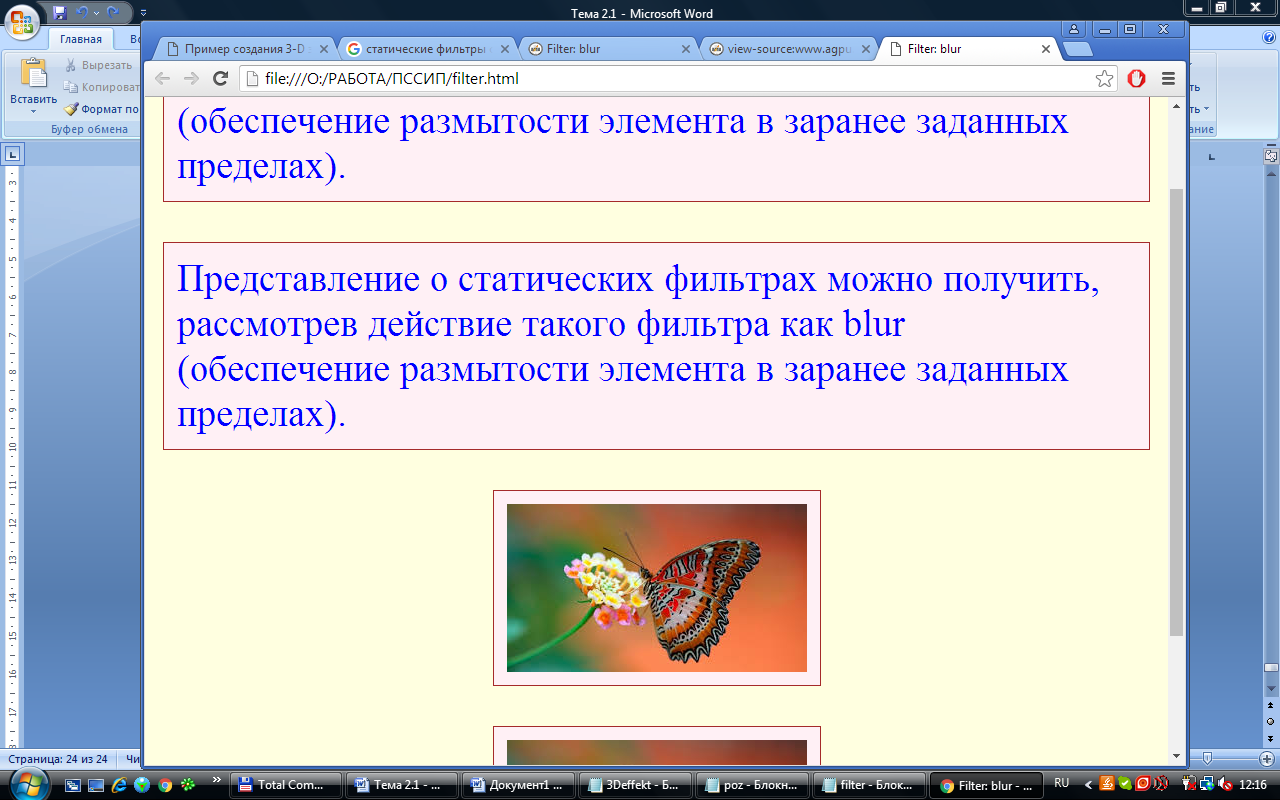
**</TD></TR>**

**</TABLE>**

**<CENTER>**

**</BODY>**

**</HTML>**

****

[*Пример использования статического фильтра*](http://html-exp.narod.ru/ex_css_61.htm) *Alpha:*

Фильтр Alpha обеспечивает прозрачность элемента страницы. Прозрачность элемента может быть как равномерной, так и меняющейся. В случае меняющейся прозрачности указываются значения прозрачности в начальной и конечной точках её изменения.

Прозрачностью управляют 3 параметра фильтра opacity, finishOpacity, style:

**opacity** - степень прозрачности в начальной точке изменения прозрачности;

**finishOpacity** - степень прозрачности в конечной точке изменения прозрачности;

**style** - вариант изменения прозрачности:

**0** — степень прозрачности не меняется от начальной к конечной точке;

**1** — степень прозрачности меняется линейно;

**2** — степень прозрачности меняется радиальным образом;

**3** — степень прозрачности меняется прямоугольным образом.

**<TABLE>**

**<TR><TD>**

**<img src="vlnt.jpg" border="0"**

**STYLE="position: relative;**

**left: 0;**

**top: 0;**

**filter: alpha(style=1,**

**opacity=0,**

**finishOpacity= 50);">**

**</TD></TR>**

**</TABLE>**

*Образцы применения фильтра Alpha (синтаксис версии 5.5) приводятся ниже :*

*Фрагмент HTML-документа:*

**<TABLE>**

**<TR><TD>**

**<img src="vlnt.jpg" border="0"**

**STYLE="position: relative; left: 0; top: 0;**

**filter:**

**progid:DXImageTransform.Microsoft.alpha(finishOpacity=50,**

**Opacity=0,**

**style=1);">**

**</TD></TR>   
</TABLE>**

**Динамические фильтры**

*Работает только в Internet Explorer!*

Динамические фильтры, в отличие от статических, обязательно требуют использования несложных скриптов. Динамические фильтры позволяют организовать постепенное появление или исчезновение изображения, трансформацию одного графического объекта в другой, а также имитацию освещения.

*Как уже отмечалось, для их применения необходимы сценарии. Рассмотрим простейшие сценарии, связанные с применением динамических фильтров.*

Трансформация (revealtrans)

Суть трансформации графического объекта заключается в том, что сначала необходимо зафиксировать первое изображение, затем заменить его другим (или изменить параметры того же самого изображения), а после этого выполнить собственно трансформацию. Все эти действия реализует сценарий.

Фиксацию и трансформацию изображения осуществляют с помощью специальных методов (функций) фильтра — **apply()** и **play()** соответственно. Для остановки процесса преобразования предназначен метод **stop()**. Для трансформации объектов служат два фильтра: **revealtrans** и **blendtrans**. По существу, второй из них является частным случаем первого.

Фильтр **revealtrans** служит для трансформации изображения — постепенного появления или перехода от одного изображения к другому. Кроме параметра **enabled** (которым обладают все фильтры), он имеет следующие параметры:

- duration — длительность преобразования в секундах (число с плавающей точкой);

- transition — тип преобразования (целое число от 0 до 23):

0 — Box In (стягивающийся прямоугольник);

1 — Box Out (расширяющийся прямоугольник);

2 — Circle In (стягивающийся круг);

3 — Circle Out (расширяющийся круг);

4 — Wipe Up (стирание вверх);

5 — Wipe Down (стирание вниз);

6 — Wipe Right (стирание вправо);

7 — Wipe Left (стирание влево);

8 — Vertical Blinds (вертикальные жалюзи);

9 — Horisontal Blinds (горизонтальные жалюзи);

10 — Checkerboard Across (сужающиеся клетки шахматной доски);

11 — Checkerboard Down (закрывающаяся шахматная доска);

12 — Random Dissolve (случайный наплыв);

13 — Split Vertical In (вертикальное деление внутрь);

14 — Split Vertical Out (вертикальное деление наружу);

15 — Split Horisontal In (горизонтальное деление внутрь);

16 — Split Horisontal Out (горизонтальное деление наружу);

17 — Strips Left Down (стирание влево вниз);

18 — Strips Left Up (стирание влево вверх);

19 — Strips Right Down (стирание вправо вниз);

20 — Strips Right Up (стирание вправо вверх);

21 — Random Bars Horisontal (случайные горизонтальные полосы);

22 — Random Bars Vertical (случайные вертикальные полосы);

23 — Random selection of (0—22) (случайный выбор из предыдущих вариантов).

*Сначала рассмотрим применение фильтра revealtrans для создания эффекта появления изображения, в котором изображение постепенно появляется:*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 transitional//EN"**

**"http://www.w3.org/TR/HTML4.01/loose.dtd">**

**<html>**

**<head><title>Фильтр revealtrans</title></head>**

**<body>**

**<img id="myimg" src="site/picture.jpg"**

**style="visibility:hidden;**

**filter:revealtrans(duration=2, transition =12)"/>**

**</body>**

**<script type="text/javascript">**

**/\* фиксируем исходное состояние изображения: \*/**

**document.getElementById("myimg").filters["revealtrans"].apply();**

**/\* делаем изображение видимым: \*/**

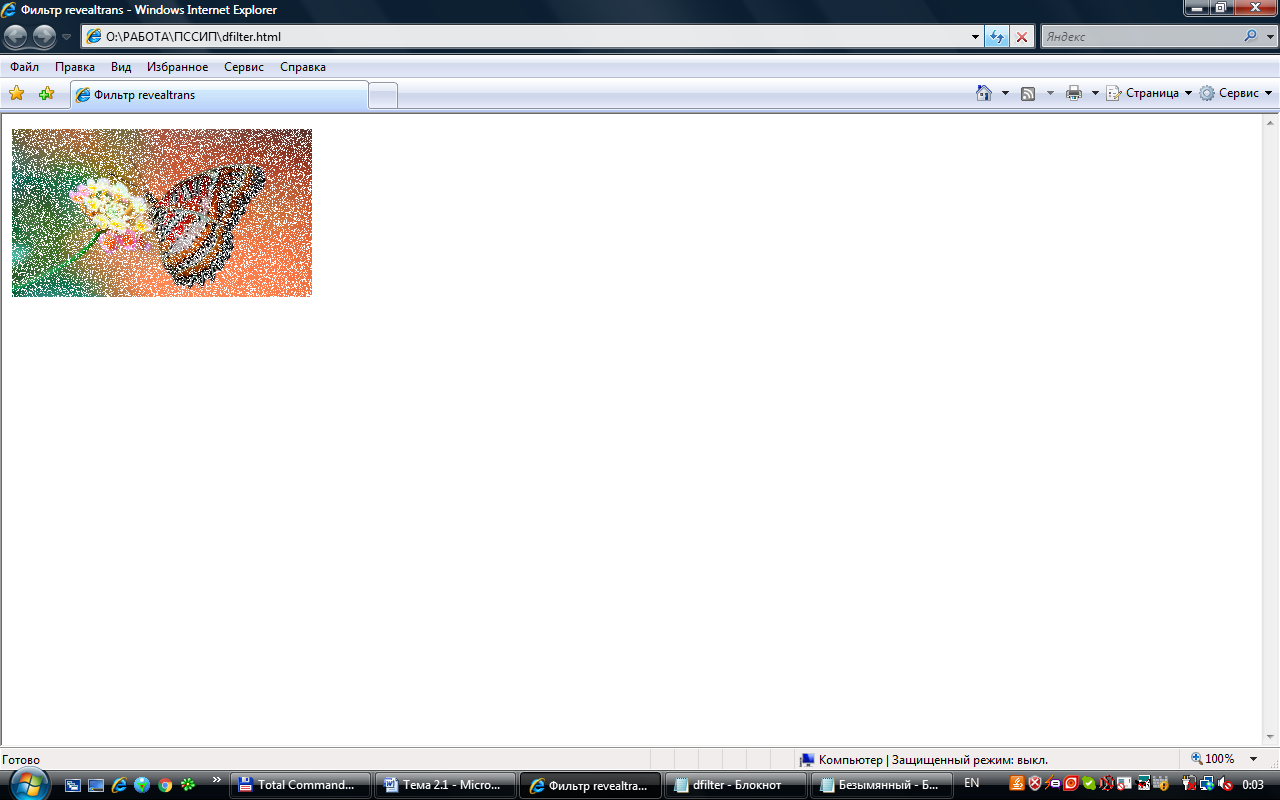
**document.getElementById("myimg").style.visibility="visible";**

**/\* выполняем преобразование: \*/**

**document.getElementById("myimg").filters["revealtrans"].play()**

**</script>**

**</html>**



*Здесь в таблице стилей для изображения устанавливается его невидимость* ***(visibility:hidden)*** *и фильтр c указанием его имени и параметров. В секции сценария с помощью метода* ***applay()*** *сначала фиксируется исходное невидимое состояние, затем изображение делается видимым и, наконец, с помощью метода* ***play()*** *выполняется переход от невидимого состояния к видимому.*

*Доступ к методам фильтра* ***apply()****,* ***play()*** *и* ***stop()*** *осуществляется через объект* ***filters*** *с указанием идентификатора (значения атрибута id, в примере — myimg) соответствующего элемента. Это делается с помощью выражения:*

**document.getElementById("myimg").filters["revealtrans"].apply()**

*Очевидно, чтобы добиться исчезновения изображения, необходимо поступить наоборот: сначала сделать его видимым, а затем невидимым.*

Теперь рассмотрим более общий случай: трансформацию одного графического объекта в другой. В отличие от рассмотренного ранее случая, эта задача сводится к установке разных изображений (начального и конечного), а не различных состояний одного и того же изображения. Это делается путем присвоения нужных значений свойству **src** объекта, значением которого является URL-адрес файла с изображением.

*Пример перехода между различными изображениями:*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 transitional//EN"**

**"http://www.w3.org/TR/HTML4.01/loose.dtd">**

**<html>**

**<head><title>Фильтр revealtrans</title></head>**

**<body>**

**<img id="myimg" src="site/picture.jpg"**

**style="visibility:visible;**

**filter:revealtrans(duration=2, transition =12)"/>**

**</body>**

**<script type="text/javascript">**

**/\* фиксируем исходное изображение: \*/**

**document.getElementById("myimg").filters["revealtrans"].apply();**

**/\* заменяем изображение: \*/**

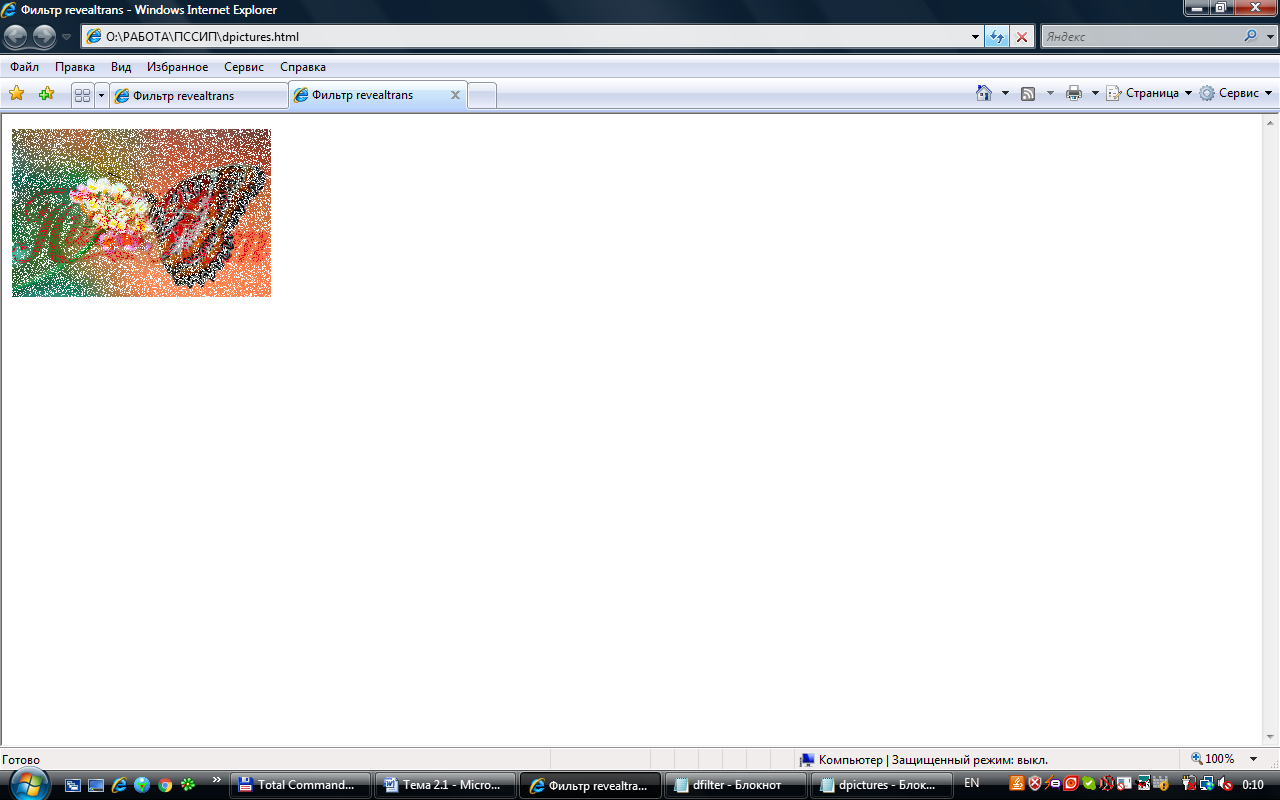
**document.getElementById("myimg").src="site/images.jpg";**

**/\* выполняем преобразование: \*/**

**document.getElementById("myimg").filters["revealtrans"].play()**

**</script>**

**</html>**



*Переход между различными изображениями или различными состояниями одного и того же изображения можно выполнять и с помощью фильтра* ***blendtrans****, который принимает единственный параметр* ***duration****, определяющий длительность преобразования в секундах, и обладает такими же методами. В отличие от revealtrans, он осуществляет преобразование единственным способом: одно изображение постепенно исчезает, а другое — появляется*.

Освещение (light)

Создать эффект освещения графического объекта источником света можно с помощью фильтра **light**, который имеет единственный параметр **enabled**, принимающий значения **true** (разрешено применение) и **false** (запрещено применение). *Значение по умолчанию — true.* Фильтром управляют из сценариев с помощью нескольких методов. *Методы фильтра задают тип источника освещения и параметры собственно освещения.*

Методы фильтра light:

**addAmbient(r, g, b, s)** — добавляет источник рассеянного (ненаправленного) света, параллельные лучи которого направлены перпендикулярно плоскости страницы. Первые три параметра метода задают яркости соответственно красной, зеленой и синей составляющих света. Последний параметр определяет интенсивность источника света. Значения параметров — целые числа в диапазоне от 0 до 255;

**addCone**(x1, y1, z, x2, y2, r, g, b, s, a) — добавляет источник направленного света. Принимает следующие параметры:

x1, y1 — координаты источника света в пикселах;

z — номер слоя (z-index) источника света;

x2, y2 — координаты точки, в которую направлен свет;

r, g, b — яркости соответственно красной, зеленой и синей составляющих света (0—255);

s — интенсивность источника света (0—255);

a — угол конуса света (0—360). Прямая, проходящая через точки источника и падения света, является биссектрисой этого угла;

**addPoint(x1, y1, z, r, g, b, s)** — добавляет точечный источник света. Принимает параметры:

x1, y1 — координаты источника света в пикселах;

z — номер слоя (z-index) источника света;

r, g, b — яркости соответственно красной, зеленой и синей составляющих света (0—255);

s — интенсивность источника света (0—255).

*При использовании только направленного источника света часть изображения, находящаяся вне светового конуса, не видна. Чтобы сделать ее видимой, необходимо добавить источник рассеянного света.*

*Пример применения фильтра light c двумя источниками света — направленным и рассеянным:*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 transitional//EN"**

**"http://www.w3.org/TR/HTML4.01/loose.dtd">**

**<html>**

**<head><title>Фильтр light</title></head>**

**<body>**

**<img id="myimg" src="picture.jpg"**

**style="filter:light"/>**

**</body>**

**<script type="text/javascript">**

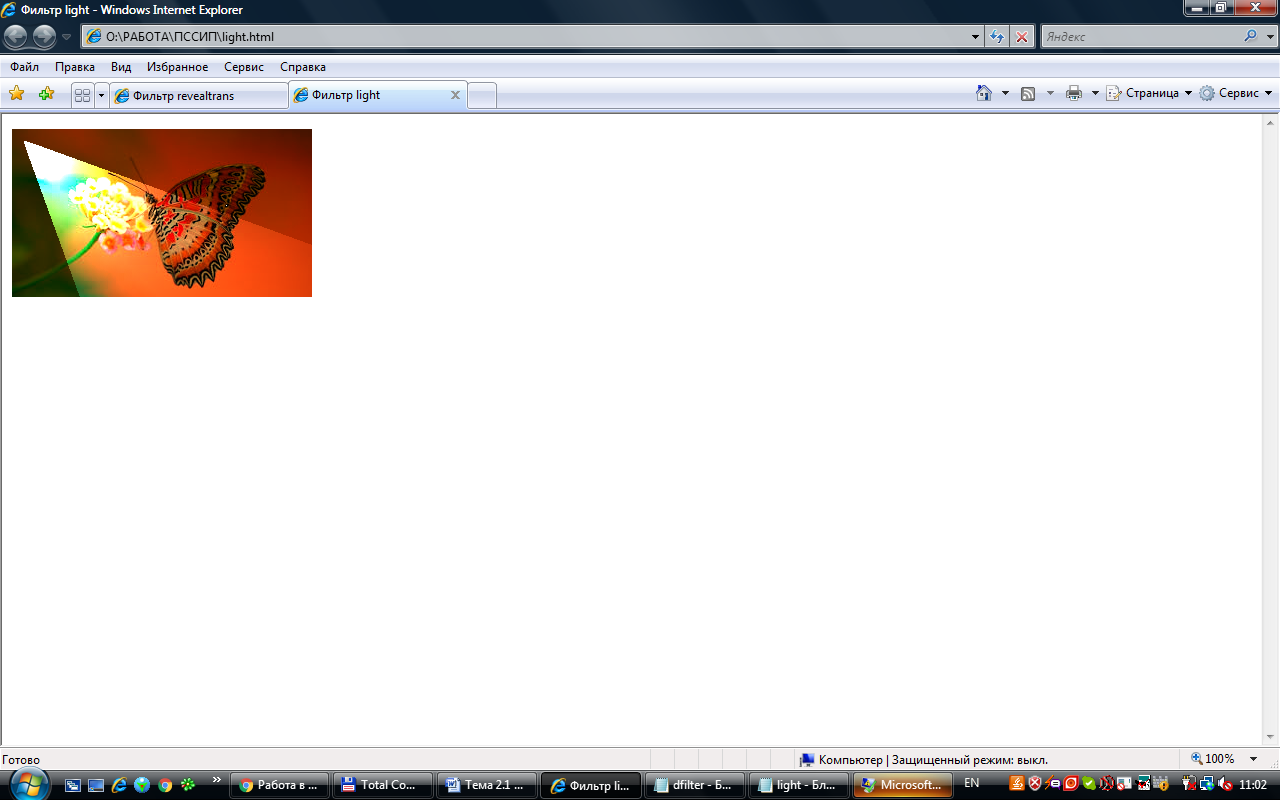
**/\* добавляем направленный свет: \*/**

**document.getElementById("myimg").filters["light"].addCone(**

**10,10,0,100,100,255,255,255,255,25);**

**/\* добавляем рассеянный свет: \*/**

**document.getElementById("myimg").filters["light"].addAmbient(**

**200,100,0,100);**

**</script>**

**</html>**

*Пример управления лучом света:*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 transitional//EN"**

**"http://www.w3.org/TR/HTML4.01/loose.dtd">**

**<html>**

**<head><title>Управление лучом света</title>**

**<script type="text/javascript">**

**/\* Для всех браузеров \*/**

**function init(){return null}**

**</script>**

**<!--[if IE]>**

**<style type="text/css">**

**#myimg {filter:light}**

**</style>**

**<script type="text/javascript">**

**function init() { // исходная установка освещения**

**document.getElementById("myimg").filters["light"].addCone(**

**10,10,0,100,100,255,255,255,255,25);**

**document.getElementById("myimg").filters["light"].addAmbient(**

**200,100,0,100);**

**/\* назначаем обработчик щелчка на изображении: \*/**

**document.getElementById("myimg").onclick=lightmove**

**}**

**function lightmove(){ // перемещение луча**

**/\* удаляем старый фильтр и снова устанавливаем: \*/**

**document.getElementById("myimg").style.filter=null;**

**document.getElementById("myimg").style.filter="light";**

**/\* устанавливаем луч света: \*/**

**document.getElementById("myimg").filters["light"].addCone(**

**10,10,0,**

**event.clientX, event.clientY,**

**255,255,255,255,25);**

**/\* рассеянный свет: \*/**

**document.getElementById("myimg").filters["light"].addAmbient(**

**200,100,0,100);**

**}**

**</script>**

**<![endif]-->**

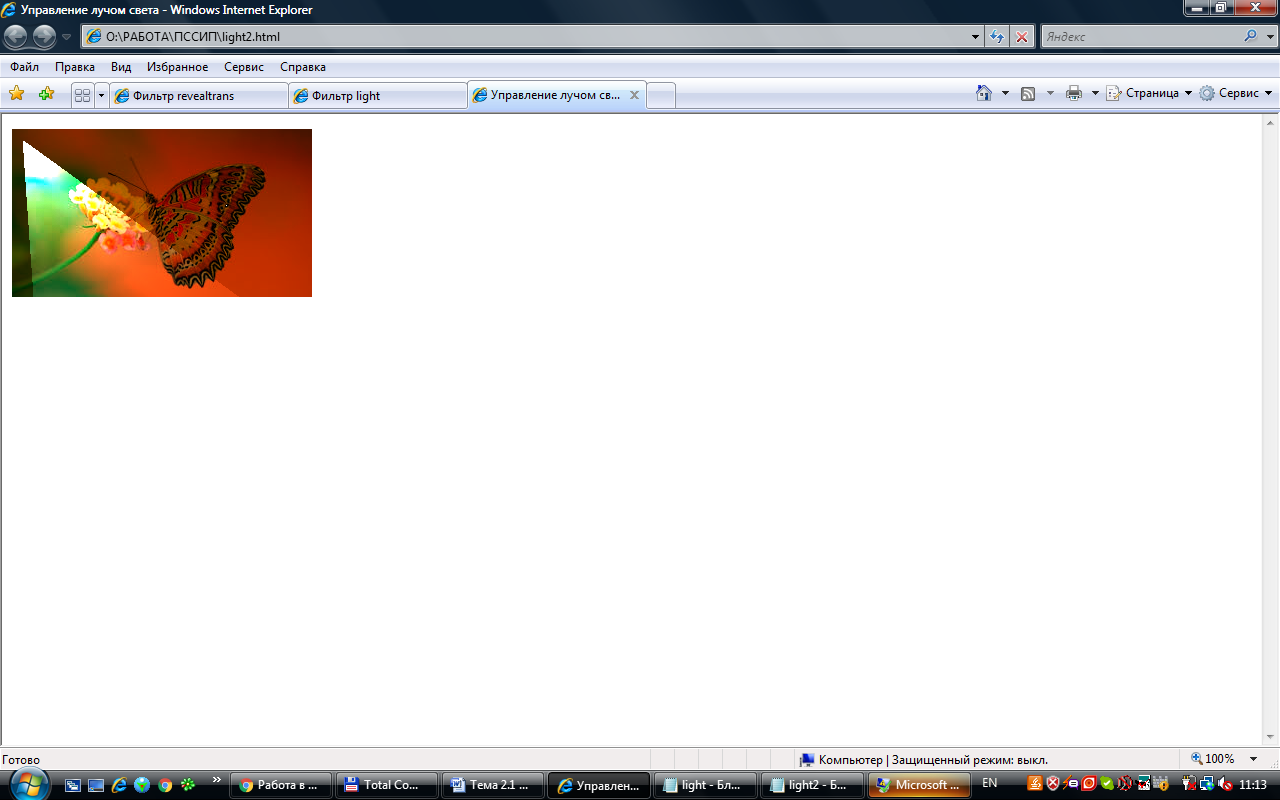
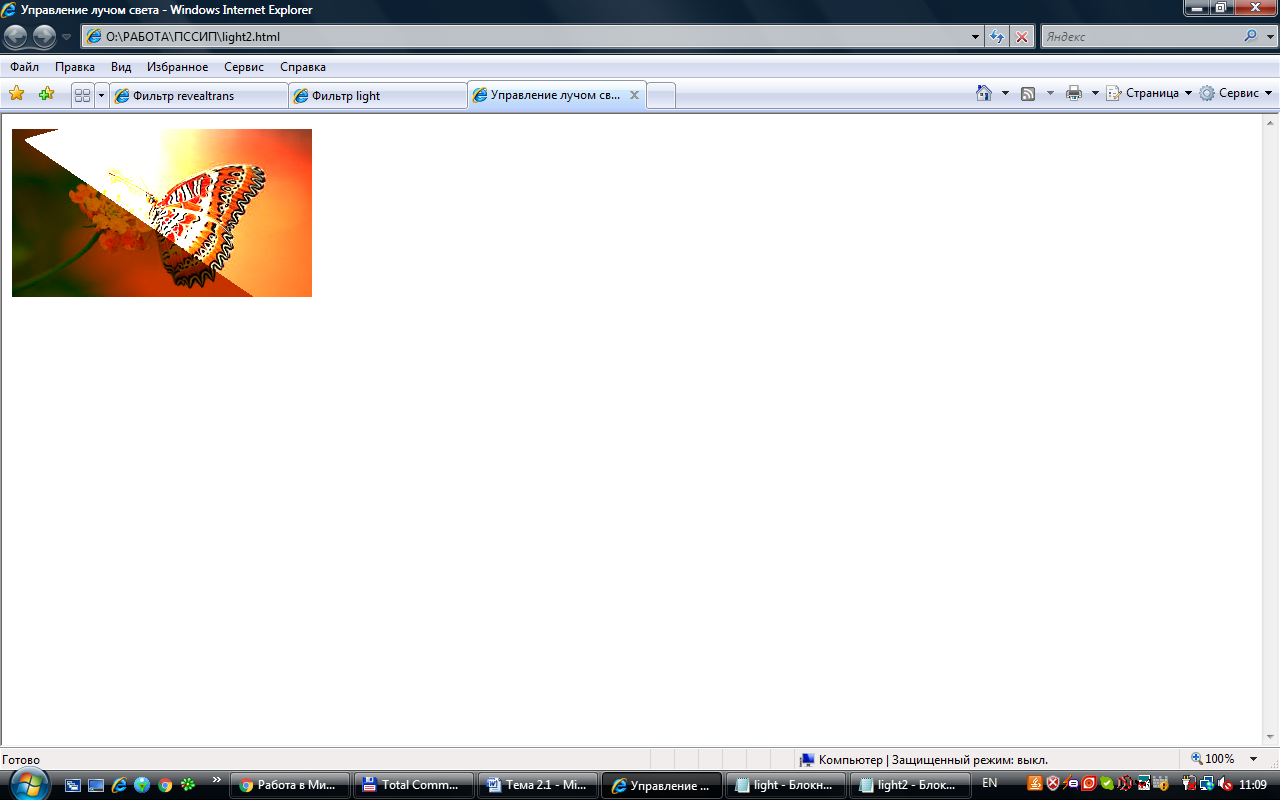
**</head>**

**<body onload="init()">**

**<img id="myimg" src="picture.jpg" />**

**</body>**

**</html>**



*В данном примере луч света изменяет направление в сторону щелчка мышью на графическом изображении. Координаты щелчка определяются свойствами client и clientY объекта события event. Приведенный код работает лишь в браузере Internet Explorer. Другие браузеры отображают только картинку без фильтра. Все, что заключено между дескрипторами* ***<!--[if IE]>*** *и* ***<!endif-->****, воспримет только Internet Explorer.*

Применение нескольких фильтров одновременно

К одному и тому же элементу можно применить несколько фильтров одновременно, достигая при этом сложных и интересных визуальных эффектов. Для этого все используемые фильтры перечисляют через пробел или запятую в правиле CSS вслед за ключевым словом filter.

*Пример применению двух фильтров к текстовому содержимому тега <p> shadow (тень) и alpha (прозрачность):*

**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 transitional//EN"**

**"http://www.w3.org/TR/HTML4.01/loose.dtd">**

**<html>**

**<head><title>Несколько фильтров</title>**

**<style type="text/css">**

**body {background: url("site/images.jpg")}**

**p {**

**font-size:135px;color:#606060;**

**position:absolute;top:-50px;left:400px;**

**filter: shadow(direction=225, color:#ffffff) alpha(opacity=70);**

**}**

**</style>**

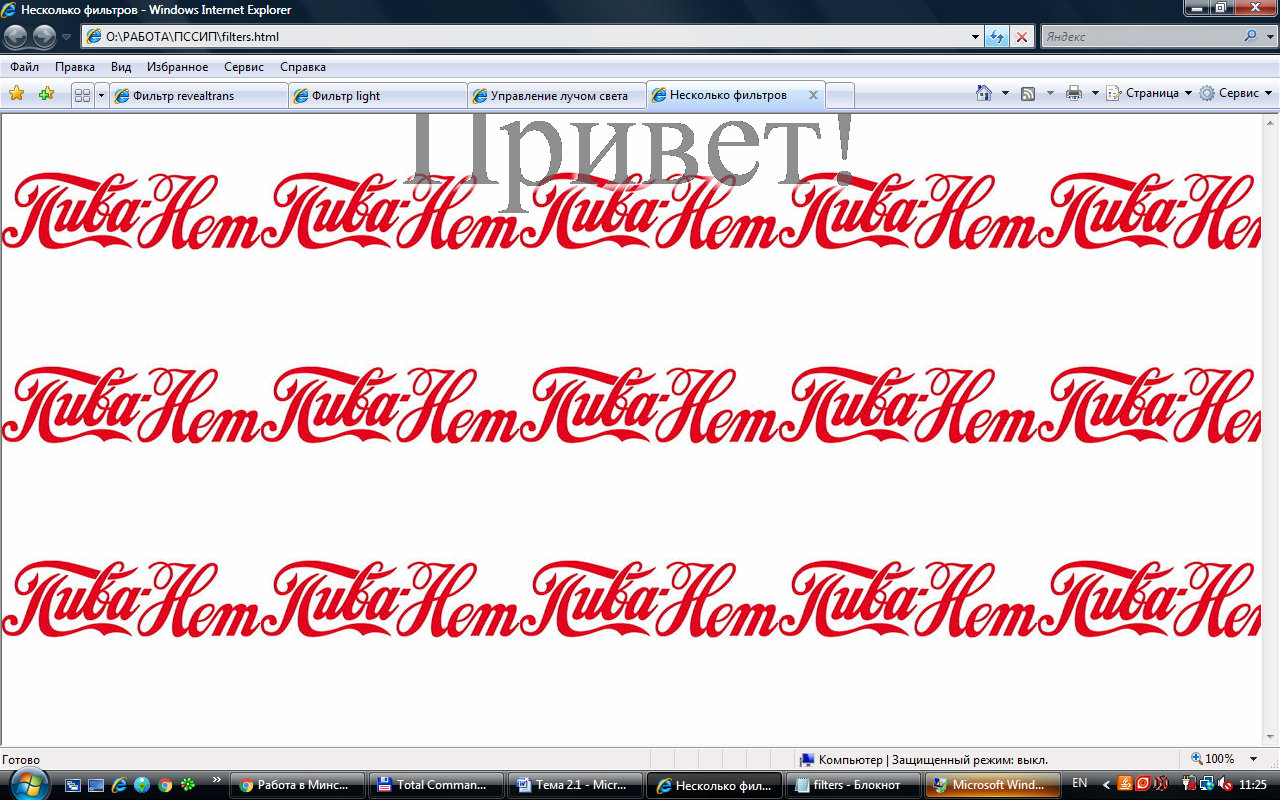
**</head>**

**<body>**

**<p>Привет!</p>**

**</body>**

**</html>**



*В результате полупрозрачный подсвеченный текст отображается на фоне графического изображения.*