Лекция №3 **СОЗДАНИЕ HTML СТРАНИЦ**

1 Основы CSS

17 декабря 1996 года была принята первая версия CSS как рекомендация *W3C (World Wide Web Consortium (Консорциума Всемирной Паутины))*. 18 мая 1998 года принята вторая версия CSS, CSS2. 8 сентября 2009 года утверждена CSS2.1, как исправленная версия CSS2. В настоящий момент существует CSS3. Все имеющиеся в природе стандарты и версии CSS, да и HTML тоже можно найти на сайте W3C.

CSS (Cascading Style Sheets), или каскадные таблицы стилей, описывают правила форматирования отдельного элемента веб-страницы. Создав стиль один раз, его можно применять к любым элементам страницы сколько угодно раз.

Рис. 1. Структура объявления стиля элемента в CSS

Определение стиля состоит из двух основных частей: самого элемента веб-страницы – **селектора**, и команды форматирования –**блока объявления**. Селектор сообщает браузеру, какой именно элемент форматировать, в блоке объявления перечисляются форматирующие команды.

A. Добавление CSS к веб-странице

Первый способ (external style sheets - внешние таблицы стилей). Если вы хотите использовать описанные *правила стилей* для многих страниц своего сайта, то их нужно сохранять в отдельном *файле стилей*. Для этого создается обычный текстовый файл, с помощью текстового редактора, с расширением *.css* и в него вписываются посредством *CSS* описанные *правила стилей*. Затем для того, чтобы увязать *web-страницу* и *файл стилей*, применяем метатег <LINK> в теге <HEAD> такого вида:

<LINK REL="STYLESHEET" TYPE="text/css" HREF="URL">
или
<STYLE TYPE="text/css" MEDIA="all">@import "URL";</STYLE>,
где атрибуты REL и TYPE указывают браузеру, что в *web-документе* будут использоваться *каскадные таблицы стилей CSS*. Ну а атрибут HREF это URL(Uniform Resource Locator) *файла стилей*. Здесь URL - это либо абсолютный адрес файла стилей, если он размещен на другом сервере, либо относительный адрес, если файл размещен на том же веб-сервере. Параметр MEDIA определяет тип системы отображения информации (монитор, принтер или другие), @import - указывает, что стили должны импортироваться с файла, по указанному адресу.

Итак,

- *файл стилей* находится на веб-сервере, его можно применять для множества веб-страниц, у которых есть на него ссылка;
- при загрузке на компьютер пользователя, происходит его кэширование, что не влияет на время загрузки других страниц сайта;
- при необходимости изменения стиля какого-то элемента веб-страниц, изменения вносятся только в одном месте - в файле стилей;
- соблюдена схема разделения описания оформления данных страниц от самих данных (в самом веб-документе находится только информация и ни слова о ее оформлении).

Для HTML5 подключение выполняется следующим образом!

<LINK HREF="STYLE.CSS" TYPE="TEXT/CSS">

Второй способ (document style sheets - таблицы стилей документа) - включение описаний *правил стилей* непосредственно в код *web-страницы*, внутри тегов <HEAD>...</HEAD>. Для этого применяется тег

<STYLE TYPE="text/css">...</STYLE>.

Параметр TYPE обязательный и указывает браузеру об использовании *каскадных таблиц стилей CSS*. Ниже приведем пример, определяющий *правило стиля* для заголовков первого уровняю H1. Тип шрифта - Verdana или другой из указанного списка, на случай, если у пользователя такой не установлен в операционной системе; размер шрифта - 18 пикселей; выравнивается текст заголовка по центру; цвет заголовка - коричневый.

<head>
<style type="text/css">
H1 { Font-Family: Verdana, ARIAL, HELVETICA, SANS-SERIF; Font-size: 18px; Text-align: center; color: #996666;}
</style>
</head>

Для данного способа:
- заданные *правила стилей* применяются только к веб-странице, в которую они включены;
- объем страницы увеличивается, что увеличивает время ее загрузки;
- для изменения стиля, правки вносятся непосредственно в html-код страницы;
- в коде веб-документа размещается и описание оформления данных.

Третий способ (inline styles - стили, вставляемые в строку) - *стиль* задается конкретному элементу в его *теге*. Этот способ применяется, когда нужно задать соответствующий стиль какому-то одному элементу в веб-документе. Ниже посмотрим на тег
** Пример задания стиля в теге, определяющем абзац **.

Здесь для абзаца, значением параметра *style*, задан отступ слева от края окна на 20 пикселей, определен шрифт - Arial, выделение - полужирным, размер шрифта - 16 пикселей, цвет - темно-синий. К другим элементам веб-страницы этот стиль никакого отношения не имеет. А вот как это будет выглядеть в окне вашего браузера:

Пример задания стиля в теге, определяющем абзац.

В этом способе:
- как и в предыдущем, объем страницы увеличивается, что ведет к увеличению времени ее загрузки;
- для изменения стиля, правки нужно вносить непосредственно в конкретный тег страницы;
- в тегах элементов веб-документа размещается описание стилей.

б. Селекторы

С помощью **селекторов** создаются CSS-правила для форматирования элементов веб-страницы. В качестве селекторов используются не только сами элементы и их классы и идентификаторы, а также псевдоклассы и псевдоэлементы.

Псевдоклассы позволяют добавлять особые классы к элементам, выбирая объекты, которых нет в структуре веб-страницы, либо которые нельзя выбрать с помощью обычных селекторов, например, первая буква или первая строка одного абзаца.

Псевдоэлементы выбирают элементы, которые не являются элементами структуры html-страницы.

	Описание, пример
Универсальный	Универсальный селектор соответствует любому элементу, например, следующая запись обнулит отступы для всех элементов веб-сайта: *** {margin: 0;}**
Элемента	Селекторы элементов используются для определения стилей элементов для всех страниц веб-сайта, например, стиль заголовков h1 или общий стиль абзацев: **h1 {font-family: Lobster, cursive;}** **p {letter-spacing: 0.1em;}**
Класса	Селекторы класса используются для определения стилей для нескольких элементов одного типа, размещенных в разных частях или на разных страницах веб-сайта. Для создания заголовка класса headline необходимо добавить атрибут class с соответствующим значением в открывающий тег . Далее необходимо задать стиль для указанного класса. Данный стиль оформления можно применить и для других элементов. Код HTML: **<h1 class="headline">Инструкция пользования персональным компьютером ** Код CSS: **.headline {** **text-transform: uppercase;** **color: lightblue;}**
Идентификатора	Селекторы идентификатора используются для присваивания стиля одному конкретному элементу. Идентификатор id элемента можно использовать в документе только один раз, так как он выделяет уникальный элемент. **#sidebar {** **text-transform: uppercase;** **color: lightblue;}**

Потомка	К потомкам элемента относятся его дочерние элементы. Селекторы потомков позволяют стилизовать все вложенные элементы, например, можно отформатировать внешний вид элементов маркированного списка: **ul li {text-transform: uppercase;}** – выберет все элементы li, являющиеся потомками всех элементов ul. Если нужно отформатировать потомки определенного элемента, то можно задать ему стилевой класс: p.first a {color: green;} – означает, что нужно применить данный стиль ко всем ссылкам, потомкам абзаца, относящегося к классу с именем first; **p .first a {color: green;}** -если добавить пробел, то будут выбраны ссылки, расположенные внутри любого тега класса .first, который является потомком элемента <p>; **.first a {color: green;}** – данный стиль применяется к любой ссылке, расположенной внутри других тегов, обозначенных классом .first.
Дочерний	Дочерний тег является прямым потомком содержащего его тега, т.е. отношения “родители-дети” существуют между элементами и теми элементами, которые содержатся непосредственно внутри них. У одного элемента может быть несколько дочерних элементов, а родительский элемент может быть у каждого элемента только один. **p > strong** – выбирает все элементы strong, являющиеся дочерними по отношению к элементу p.
Сестринский	Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать теги 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 после него.
Атрибута	Селекторы атрибутов позволяют форматировать элементы на основе выборки любых содержащихся в них атрибутов или значений атрибутов, варианты: **[атрибут]** – выбирает элементы, для которых задан атрибут. **img[alt]** – выбирает все картинки, содержащие атрибут alt. **img[title="flower"]** – выбирает все картинки, название которых содержит слово flower. **a[href^="http://"]** – выбирает все ссылки, начинающиеся на http://. **img[src$=".png"]** – выбирает все картинки, название которых заканчивается на .png.

	a[href*="ru"] – выбирает все ссылки, название которых содержит слово ru. **input[type="text"]** – выбирает только текстовые поля формы. **article[class~="feature"]** – выбирает статьи по частичному значению атрибута, т.е. статьи, название класса которых содержит данное слово. **article[id|="feature"]** – выбирает элемент, атрибут которого эквивалентен feature или начинается на feature.
Псевдокласс	Псевдоклассы – это классы, фактически не прикрепленные к тегам html-кода. Они вызывают CSS-правила при совершении того или иного события или подчиняющиеся тому или иному правилу: **a:link** – ссылается на не посещенную ссылку. **a:visited** – ссылается на уже посещенную ссылку. **a:hover** – ссылается на любой элемент, по которому проводят курсором мыши. **a:focus** – ссылается на любой элемент, над которым находится курсор мыши. **a:active** – ссылается на элемент, который был активизирован пользователем. **input:valid** – выберет поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу. **input:invalid** – выберет поля формы, содержимое которых не соответствует указанному типу. **input:enabled** – выберет все доступные (активные) поля форм. **input:disabled** – выберет заблокированные поля форм, т.е., находящиеся в неактивном состоянии. **input:in-range** – выберет поля формы, значения которых находятся в заданном диапазоне. **input:out-of-range** – выберет поля формы, значения которых не входят в установленный диапазон. **input:not([type="submit"])** – выберет все поля формы, кроме полей отправки данных. **p:lang(en-US)** – выберет все абзацы на английском языке. **:not(селектор)** – выберет элементы, которые не содержат указанный селектор, например, класс, идентификатор или селектор элемента. **:target** – выбирает элемент с символом #, на который ссылаются в документе. **:checked** – выбирает выделенные (выбранные пользователем) элементы.

Структурные псевдоклассы	Структурные псевдоклассы форматируют дочерние элементы в соответствии с указанным параметром в скобке, варианты: **:nth-child(odd)** – выбирает нечетные дочерние элементы. **:nth-child(even)** – выбирает четные дочерние элементы. **:nth-child(3n) –** выбирает 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).
Структурные псевдоклассы типа	Позволяют указать на конкретный тип дочернего тега: **:nth-of-type()** – выбирает элементы по аналогии с :nth-child(), при этом берет во внимание только тип элемента. **:first-of-type** – позволяет выбрать первый дочерний элемент. **:last-of-type** – выбирает последний тег конкретного 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.
Псевдоэлементы	Псевдоэлементы используются для добавления содержимого, которое генерируется с помощью свойства content, и для изменения внешнего вида части элемента: **: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.

в. Комбинация селекторов

Чтобы добиться более четкого выбора элементов для форматирования, можно не ограничиваться заданием одного типа селектора, а использовать комбинации селекторов, например:
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.

г. Группировка селекторов

Можно применить один и тот же стиль к нескольким элементам. Для этого необходимо в левой части объявления поместить через запятую нужные селекторы, например:

h1, h2, h3, h4 {color: tomato; background: white;}

д. Принцип каскадирования и специфичность правила

Каскадирование представляет собой процесс применения различных правил к одному и тому же элементу. Более конкретные правила имеют приоритет над более общими. Если в отношение одного и того же элемента определено несколько стилей, то в результате к нему будет применен последний из них. Для каждого правила браузер вычисляет **специфичность селектора**, и, если у элемента имеются конфликтующие объявления свойств, во внимание принимается правило, имеющее наибольшую специфичность.

Значение специфичности состоит из четырех частей: 0, 0, 0, 0.
Специфичность селектора определяется следующим образом:
для id добавляется 0, 1, 0, 0;
для class добавляется 0, 0, 1, 0;
для каждого элемента и псевдо элемента добавляется 0, 0, 0, 1;
для встроенного стиля, добавленного непосредственно к элементу1, 0, 0, 0;
универсальный селектор не имеет специфичности.

h1 {color: lightblue;} /*специфичность 0, 0, 0, 1*/
h1 em {color: gold;} /*специфичность: 0, 0, 0, 1 + 0, 0, 0, 1 = 0, 0, 0, 2*/
div#main p.about {color: blue;}
/*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 + 0, 0, 0, 1 + 0, 0, 1, 0 = 0, 1, 1, 2*/
#sidebar {color: orange;} /*специфичность 0, 1, 0, 0*/
li#sidebar {color: aqua;} /*специфичность: 0, 0, 0, 1 + 0, 1, 0, 0 = 0, 1, 0, 1*/

В результате к элементу применятся те правила, специфичность которых больше, например, если на элемент действуют две специфичности со значениями 0, 0, 0, 2 и 0, 1, 0, 1, то выиграет второе правило.

Вес правила также можно задать с помощью ключевого слова !important, которое добавляется после значения свойства, например, font-weight: bold!important. Правило необходимо размещать в конец объявления перед закрывающей скобкой.

2 Форматирование текста в CSS

CSS предоставляют вебмастеру отличную возможность управления отображением текста. Изменение расстояния между символами, всевозможные отступы, стиль символов и управление шрифтами – это и многое другое позволяют изменять таблицы стилей.

А)Свойства, позволяющие форматировать текст в CSS:

Color – позволяет задает цвет элемента. В качестве параметра может выступать как шестнадцетиричное, так и буквенное значение цвета.

Direction – позволяет изменить направление текста. К сожалению, не во всех браузерах этот параметр будет отображаться правильно.

ltr – слева направо

rtl – справа налево.

```
div.d1{direction: ltr;}
```

Letter – spacing – задает интервал между символами.

normal – обычные интервалы

length – пользовательский интервал.

```
div.c {letter – spacing: 7px;}
```

Text – align – задает выравнивание текста внутри элемента и может принимать значения:

Left – выравнивает текст слева

```
div.e1{text – align: left;}
```

Right – выравнивает текст справа

Center – выравнивает текст по центру

Justify – выравнивает текст по ширине

Text – decoration – оформление текста. Может принимать значения:

None – обычный текст

Underline – подчеркнутый снизу текст

Overline – подчеркнутый сверху текст

Line – through – зачеркнутый текст

Blink – мигающий текст

Text – indent – задает отступ для первой строки текста.

Text – transform – управляет размером символов и может принимать следующие значения:

None – обычный текст

Capitalize – каждое слово начинается с заглавной буквы.

Uppercase – только большие буквы

Lowercase – маленькие буквы

White – space – задает способ отображения пробелов и может принимать следующие значения:

normal – допускается только один пробел.

pre – вся структура документа, с неограниченным количеством пробелов сохраняется.

nowrap – текст не будет переноситься, пока не встретит тег ***
***

Word – spacing – задает интервал между словами.

Б)Работа со шрифтами в CSS

Font – family – определяет список допустимых имен шрифтов для элемента. Использован будет первый, распознанный браузером шрифт.

```
h1 {font – family : "Comic Sans MS", "Times New Roman";}
```

Font – size – задает размер шрифта и может принимать значения:

xx – small – наименьший

x – small – очень маленький

small – маленький

medium – средний

large – большой

x – large – очень большой

xx – large – наибольший

smaller – меньше, чем у порождающего элемента

larger – больше, чем у порождающего элемента

length – задает фиксированное значение шрифта

% - размер шрифта в % от размера шрифта порождающего элемента

Font – size – adjust – задает значение аспекта шрифта. Аспект шрифта – отношение между размерами маленькой буквы *x* и размером тшрифта. Чем выше это значение, тем лучше шрифт будет читаться при уменьшении размера.

Font – stretch – позволяет задать интервал между символами внутри шрифта. Принимает значения:

normal – Задает масштаб сжатия или расширения как обычный

wider – Задает масштаб расширения как следующее расширенное значение

narrower – Задает масштаб сжатия как следующее сжатое значение

ultra – condenced – максимальный масштаб сжатия

extra – condenced – сильный масштаб сжатия

condenced – сжатие

semi – condenced – слабое сжатие

semi – expanded – слабое расширение
expanded – расширение
extra – expanded – сильный масштаб расширения
ultra – expanded – максимальный масштаб расширения
div.k1 {font – stretch : wider;}

Font – style – задает стиль шрифта.
normal – нормальный шрифт.
italic – курсив.
oblique – наклонный шрифт.

Font – variant – используется для создания шрифта – капители (все маленькие символы преобразуются в большие).
normal – обычный шрифт
small – caps – шрифт – капиель.

Font – weight – позволяет задать толщину символов:
normal – обычные символы
bold – жирные символы
bolder – более жирные символы
lighter – более тонкие символы

в) Параметры размеров элементов CSS

Параметры, приведенные в таблице ниже, позволяют управлять всеми возможными размерами элементов.

Параметр	Описание	Значения
height	Задает высоту элемента	***Auto, length, %***
line – height	Задает интервал между строками	***Normal, number, length, %***
max – height	Задает максимальную высоту элемента	***None, length, %***
max – width	Задает максимальную ширину элемента	***None, length, %***
min – height	Задает минимальную высоту элемента	***Length, %***
min – width	Задает минимальную ширину элемента	***Length, %***
width	Задает ширину элемента	***Auto, %, length***

3 Единицы измерения

Для задания размеров различных элементов в CSS используются абсолютные и относительные единицы измерения. Абсолютные единицы не зависят от устройства вывода, а относительные единицы определяют размер элемента относительно значения другого размера.

Относительные единицы

Относительные единицы обычно используют для работы с текстом. В табл. 1 перечислены основные относительные единицы.

Табл. 1. Относительные единицы измерения

Единица	Описание
em	Размер шрифта текущего элемента
ex	Высота символа x
%	размер, который составляет определенную часть (%) от родительского элемента.
px	Пиксел

Единица em это изменяемое значение, которое зависит от размера шрифта текущего элемента (размер устанавливается через стилевое свойство font-size). В каждом браузере заложен размер текста, применяемый в том случае, когда этот размер явно не задан. Поэтому изначально 1em равен размеру шрифта, заданного в браузере по умолчанию или размеру шрифта родительского элемента. Процентная запись идентична em, в том смысле, что значения 1em и 100% равны.

Единица ex определяется как высота символа «x» в нижнем регистре. На ex распространяются те же правила, что и для em, а именно, он привязан к размеру шрифта, заданного в браузере по умолчанию, или к размеру шрифта родительского элемента.

Единица ch равна ширине символа «0» для текущего элемента и подобно em зависит от размера шрифта.

Разница между em и rem следующая. em зависит от размера шрифта родителя элемента и меняется вместе с ним, а rem привязан к корневому элементу, т. е. размеру шрифта заданного для элемента html.

Также есть группа относительных единиц привязанных к размеру области просмотра браузера. В табл. 2 показан их список с описанием.

Табл. 2. Относительные единицы измерения

Единица	Описание
vw	1% от ширины области просмотра
vh	1% от высоты области просмотра
vmin	1% от меньшего значения из ширины и высоты области просмотра
vmax	Определяется, что больше, значение ширины или высоты области просмотра и от него вычисляется 1%

Абсолютные единицы

Абсолютные единицы представляют собой физические размеры — дюймы, сантиметры, миллиметры, пункты, пики, а также пиксели. Для устройств с низким dpi (количество точек, приходящихся на один дюйм, определяет плотность точек) привязка идёт к пикселю.

В этом случае один дюйм равен 96 пикселям. Очевидно, что реальный дюйм не будет совпадать с дюймом на таком устройстве. На устройствах с высоким dpi реальный дюйм совпадает с дюймом на экране, поэтому размер пикселя вычисляется как 1/96 от дюйма.

В табл. 3 перечислены основные абсолютные единицы.

Табл. 3. Абсолютные единицы измерения

Единица	Описание
px	Пиксель
in	Дюйм (1 дюйм равен 2,54 см)
cm	Сантиметр
mm	Миллиметр
pt	Пункт (1 пункт равен 1/72 дюйма)
pc	Пика (1 пика равна 12 пунктам)

г) Фон в CSS

CSS предоставляет множество инструментов для работы с фоном, которые позволяют создать действительно уникальный и гармоничный для вашего сайта фон.

Background–attachment – определяет привязку фона на странице. Значение ***scroll*** – привязывает фон к полосе прокрутки (при прокрутке фон будет оставаться на месте) и значение ***fixed*** – привязывает фон к виртуальной оси координат (при прокрутке фон будет прокручиваться вместе со всей страницей).

Background–color – заполняет элемент цветом. Значение ***color –*** задает цвет элемента, а значение ***transparent –*** задает прозрачный фон.

Background–image – позволяет использовать изображение в качестве фона.

none *– фоновое изображение не используется*
url *– позволяет задать путь к изображению*

Background–position – задает начальное положение фонового изображения и может принимать следующие значения:

top – left *– верхний левый угол экрана*
top – center *– верхний центральный*
top – right *– верхний правый*
center - left *– центральный левый*
center -center *– центр экрана*

center -right *– центральный правый*
bottom -left *– нижний левый*
bottom -center *– нижний центральный*
bottom -right *– нижний правый угол экрана*
%(x y) *– задание положения в%(верхний левый 0% 0%)*
pixels(x y) *– задание положения в пикселях (верхний левый 0 0)*

При задании координат посредством ключевых слов, то если задано только одно слово (например ***top***), то второе принимается равным ***center***. При задании координат посредством процентов, если задано только одно значение, то второе принимается равным 50%. При указании положения в процентах или координатно, можно смешивать значения, например (30% 500).

Background – repeat – задает метод тиражирования изображения.
Может принимать следующие значения:

repeat *– тиражирование в обоих направлениях*
no – repeat *– изображение выводится только один раз*
repeat – x *– тиражирование по горизонтали*
repeat – y *– тиражирование по вертикали*.

1. Списки в CSS

List–style–image – позволяет установить изображение в качестве маркера для *списков в CSS*.

none *– используется маркер по умолчанию*
url *– в качестве маркера используется изображение*

List–style–position – определяет место размещения маркеров списка.

inside *– маркер располагается внутри текста*
outside *– маркер располагается слева от текста*

List–style–type – задает разнообразные маркеры для списков, которые вы можете видеть ниже:

none *– маркер не используется*
disc *– в качестве маркера используется закрашенный круг*
circle *– в качестве маркера используется закрашенный круг*
square *– закрашенный квадрат*
decimal *– простые числа*
decimal – leading – zero *– 01, 02, 03 b .....*
lower – roman *– i, ii, iii, iv, v .....*
upper – roman *– (I, II, III, IV, V .....*
lower – alpha *– a, b, c, d, e .....*
upper – alpha *– A, B, C, D, E .....*

lower – greek – альфа, бета, гамма .....
lower – latin – a, b, c, d, e .....
upper – latin – A, B, C, D, E .....
hebrew – еврейские числа
armenian – армянские числа
georgian – an, ban, gan .....
cjk – ideographic — идеографические числа
hiragana – Маркер — a, i, u, e, o, ka, ki, ...
katakana – A, I, U, E, O, KA, KI .....
hiragana – iroha – i, ro, ha, ni, ho, he, to .....
katakana – iroha – I, RO, HA, NI, HO, HE, TO .....

2. «@media» - Типы носителей в CSS

Нередко бывают случаи, что сайт необходимо отобразить различным образом в зависимости от используемого устройства, например, сайт должен выглядеть по-разному на домашнем компьютере и на мобильном телефоне.

@media позволяет использовать различные стили, для вывода на разных устройствах.

Атрибут media определяет под какие устройства оптимизирован файл. Главным образом он используется с файлами таблиц стилей, для определения различных стилей под разные типы носителей.

В примере ниже будут использоваться различные стили при выводе документа на экран и при печати документа:

```
@media screen
{div.test  {font – family: times,serif; font –
size:8px}}
@media print
{div.test{
font – family: verdana,sans – serif; font –
size:30px}}
```

Основные типы носителей вы можете видеть в списке ниже:

Устройства

Значение	Описание
all	Подходит для всех устройств (значение по умолчанию)
aural	Синтезаторы речи
braille	Устройство обратной связи шрифта Брайля
handheld	Карманные устройства (маленький экран, ограниченная

	пропускная способность)
projection	Проекторы
print	Режим для печати страниц
screen	Компьютерные экраны
tty	Телетайпы и аналогичных средств массовой информации, с помощью символов фиксированного шага сетки
tv	Телевизионный тип устройства (низкое разрешение, ограниченная способность прокрутки)

Значения параметров устройств

Значение	Описание
width	Определяет ширину области целевой страницы. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (min-width:500px)".
height	Определяет высоту области целевой страницы. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (max-height:700px)".
device-width	Определяет ширину области отображения экрана/бумаги. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (device-width:500px)".
device-height	Определяет высоту области отображения экрана/бумаги. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (device-height:500px)".
orientation	Задает способ отображения. Возможные значения: portrait или landscape. Пример: media="all and (orientation: landscape)".
aspect-ratio	Определяет соотношение ширины и высоты. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (aspect-ratio:16/9)".
device-aspect-ratio	Задает ширину/высоту устройства по отношению к области отображения экрана/бумаги. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (aspect-ratio:16/9)".
color	Указывает биты для цвета отображения. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (color:3)".
color-index	Определяет количество цветов, для экрана целевого устройства. В значение можно использовать префиксы "min-"

	и "max-". Пример: media="screen and (min-color-index:256)".
monochrome	Указывает количество битов на пиксель для буфера монохромных кадров. В значение можно использовать префиксы "min-" и "max-". Пример: media="screen and (monochrome:2)".
resolution	Определяет плотность пикселей(dpi или dpcm) для целевого экрана/бумаги. В значение можно использовать префиксы "min-" и "max-". Пример: media="print and (resolution:300dpi)".
scan	Определяет метод сканирования ТВ дисплея. Возможные значения: "progressive" и "interlace". Пример: media="tv and (scan:interlace)".

Пример:

```
<head>
<title>CSS Media</title>
<style type="text/css" media="screen">
.warning {color:#ff0000}
h1.warning {text-decoration:underline}
p.warning {font-weight:bold}
.printDisplay {display:none}
</style>
<style type="text/css" media="print">
.warning {color:#660000;}
h1.warning {text-decoration:underline; font-
size:1in;}
p.warning {font-weight:bold; font-size:.5in;}
.screenDisplay {display:none}
</style>
</head>
<body>
<h1 class="warning">WARNING</h1>
<p class="warning">Don't go there!</p>
<p class="printDisplay">This is the print
version.</p>
<p class="screenDisplay">This is the screen
version.</p>
</body></html>
```

Пример медиа-запроса :

```
@media screen and (min-width: 600px)
```

```
 and (max-width: 900px)
{  .class {
    background: #333;
  }}
```

Работа с типами носителей (@media) в CSS позволяет сделать ваш сайт действительно кроссбраузерным и мультиплатформенным, позволяет упростить отображение сайта на мобильных устройствах и многое другое...

3 Работа с псевдоклассами в CSS

Работа с классами и псевдо-классами в CSS позволяет применить дополнительные методы форматирования к текстам и некоторым другим элементам страниц сайта.

Использование псевдо–классов позволяет добиться определенных эффектов для элементов.

В общем виде синтаксис псевдо–классов выглядит следующим образом:

```
selector:pseudo-class {property: value}
```

А вот и наглядный пример:

```
a.silver:visited {color: #C0C0C0 }
```

Существует несколько основных CSS методов работы со ссылками:

`a:link {color: #808080 }` - непосещенная ссылка

`a:visited {color: #008000 }` - посещенная ссылка

`a:hover {color: #008080 }` - выбранная ссылка

`a:active {color: #00FF00 }` - нажатая ссылка

Порядок написания данного CSS кода должен сохраняться. То есть `a:hover` должен идти после `a:visited` и `a:link`, а `a:active` должен идти после `a:hover`, иначе ничего не получится.

Список псевдо–классов CSS

`:active` – позволяет добавить стиль выбранному элементу

`:focus` – позволяет добавить стиль элементу выделенному элементу

`:hover` – позволяет добавить стиль элементу, над которым находится курсор мыши

`:link` – позволяет добавить стиль непосещенной ссылке

`:visited` – позволяет добавить стиль посещенной ссылке

`:first - child` – позволяет добавить стиль первому потомку некоторого элемента

:lang – позволяет определить язык, используемый для данного элемента

Использование псевдо–элементов позволяет добиться специальных эффектов для некоторых селекторов. Ниже приведены примеры.

Синтаксис псевдо–элементов:

В общем виде синтаксис псевдо–элементов выглядит следующим образом:

```
selector:pseudo-element {property: value}
```

```
Пример:
div:first – letter {color: #808000; font – size:
150%}
div:first – line {color: #808080 }
```

Псевдо–элемент: first–line- данный псевдо- элемент позволяет добавить специальный стиль для первой строки текста в элементе:

Псевдо–элемент: first–letter - позволяет добавить специальный стиль для первой буквы в элементе:

Before - позволяет выполнить вставку каждый раз перед появлением указанного элемента.

After - позволяет выполнить вставку каждый раз после появления указанного элемента

Следующий стиль будет выводить изображение указателя каждый раз перед появлением заголовка h5. (это и есть заголовок h5).

```
h5:before
{content: url(img/list_style_type_1.gif);}
```

Следующий стиль будет выводить изображение указателя каждый раз после появлением заголовка h6. (это и есть заголовок h6).

```
h6:after
{content: url(img/list_style_type_3.gif);}
```

4. CSS: оформление таблиц

По умолчанию таблица на веб-странице отображается без рамки, для добавления рамки к таблице, как и ко всем другим элементам, используется CSS свойство border. Но стоит обратить внимание на то, что если добавить рамку только к элементу <table>, то она отобразиться вокруг всей таблицы. Для того, чтобы ячейки таблицы тоже имели рамку, надо будет установить свойство border и для элементов <th> и <td>.

table, th, td { border: 1px solid black; }

Теперь и таблица, и ячейки имеют рамки, при этом и каждая ячейка, и таблица имеют свои собственные рамки. В результате между рамками появилось пустое пространство, управлять размером этого пространства

позволяет свойство border-spacing, которое задается для всей таблицы целиком. Другими словами, нельзя управлять промежутками между различными ячейками индивидуально.

Даже если убрать промежутки между ячейками с помощью значения 0 свойства border-spacing, то рамки ячеек будут соприкасаться друг с другом, удваиваясь.

Для объединения рамок ячеек используется свойство **border-collapse**. Оно может принимать два значения:

- **separate:** является значением по умолчанию. Ячейки отображаются на небольшом расстоянии друг от друга, каждая ячейка имеет свою собственную рамку.
- **collapse:** соединяет соседние рамки в одну, все промежутки между ячейками, а также между ячейками и рамкой таблицы игнорируются.

Пример:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Название документа</title>
    <style>
      table, td, th {
        border: 4px outset blue;
       border-spacing: 5px;
      }
      .first { border-collapse: collapse; }
    </style>
  </head>

  <body>
    <table>
      <tr><th>Имя</th><th>Фамилия</th></tr>
      <tr><td>Гомер</td><td>Симпсон</td></tr>
      <tr><td>Мардж</td><td>Симпсон</td></tr>
    </table>    <br>
    <table class="first">
      <tr><th>Имя</th><th>Фамилия</th></tr>
      <tr><td>Гомер</td><td>Симпсон</td></tr>
      <tr><td>Мардж</td><td>Симпсон</td></tr>
    </table>
  </body>
</html>
```

После добавления рамок к ячейкам таблицы стало заметно, что содержимое ячеек слишком близко расположено к краям. Для добавления свободного пространства между краями ячеек и их содержимым можно воспользоваться свойством padding:

th, td { padding: 7px; }

Размер таблицы зависит от ее содержимого, но часто возникают ситуации, когда таблица оказывается слишком узкой и появляется необходимость ее растянуть. Ширину и высоту таблицы можно изменять с помощью свойств width и height, задавая нужные размеры или самой таблице или ячейкам:

table { width: 70%; }
th { height: 50px; }

По умолчанию текст в заголовочных ячейках таблицы выравнивается по центру, а в обычных ячейках текст выровнен по левому краю, используя свойство text-align можно управлять выравниванием текста по горизонтали.

CSS свойство vertical-align позволяет управлять выравниванием текстового содержимого по вертикали. По умолчанию текст выровнен вертикально по центру ячеек. Вертикальное выравнивание можно переопределить с помощью одного из значений свойства **vertical-align**:

- **top:** текст выравнивается по верхней границе ячейки
- **middle:** выравнивает текст по центру (значение по умолчанию)
- **bottom:** текст выравнивается по нижней границе ячейки

Пример

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Название документа</title>
    <style>
      table, td, th {
        border: 1px solid black;
        border-collapse: collapse;
      }
      table { width: 70% }
      td { text-align: right; }
      th { height: 50px; }
      .test1 { vertical-align: top; }
      .test2 { vertical-align: bottom; }
    </style>
  </head>
  <body>
```

```
    <table>
      <tr><th class="test1">Имя</th><th
class="test2">Фамилия</th></tr>
      <tr><td>Гомер</td><td>Симпсон</td></tr>
      <tr><td>Мардж</td><td>Симпсон</td></tr>
    </table>
  </body>
</html>
```

При просмотре больших таблиц, содержащих много строк с большим количеством информации, бывает трудно отследить, какие данные относятся к конкретной строке. Чтобы помочь пользователям сориентироваться, можно использовать два разных фоновых цвета поочередно. Для создания описанного эффекта «зебры» можно использовать селектор класса, добавляя его к каждой второй строке таблицы:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Название документа</title>
    <style>
      table {width: 70%;
        border-collapse: collapse;       }
      td, th { border: 1px solid #98bf21;
        padding: 3px 7px 2px 7px;       }
      th {text-align: left;
        padding: 5px;
        background-color: #A7C942;
        color: #fff;}
      .alt td { background-color: #EAF2D3; }
    </style>
  </head>
  <body>
    <table>
<tr><th>Имя</th><th>Фамилия</th><th>Положение</th
>
</tr>
<tr><td>Гомер</td><td>Симпсон</td><td>отец</td>
</tr>
```

```
<tr
class="alt"><td>Мардж</td><td>Симпсон</td><td>мат
ь
</td></tr>
<tr><td>Барт</td><td>Симпсон</td><td>сын</td></tr
>
<tr
class="alt"><td>Лиза</td><td>Симпсон</td><td>дочь
</td></tr>
    </table>
  </body> </html>
```

Добавлять атрибут class к каждой второй строке довольно утомительное занятие. В CSS3 был добавлен псевдо-класс :nth-child, позволяющий решить эту проблему альтернативным путем. Теперь эффекта чередования можно достичь исключительно средствами CSS, не прибегая к изменению HTML-разметки документа. С помощью псевдо-класса **:nth-child** можно выбрать все четные или нечетные строки таблицы, используя одно из ключевых слов: **even (четные) или odd (нечетные):**

tr:nth-child(odd) { background-color: #EAF2D3; }

Еще одним способом повышения удобочитаемости табличных данных является изменение фонового цвета строки при наведении на нее курсора мыши. Это поможет выделить нужное содержимое таблицы и повысит визуальное восприятие данных.

Реализовать такой эффект очень просто, для этого нужно добавить псевдо-класс **:hover** к селектору строки таблицы и задать нужный цвет фона: **tr:hover { background-color: #E0E0FF; }**

5. Блочные элементы

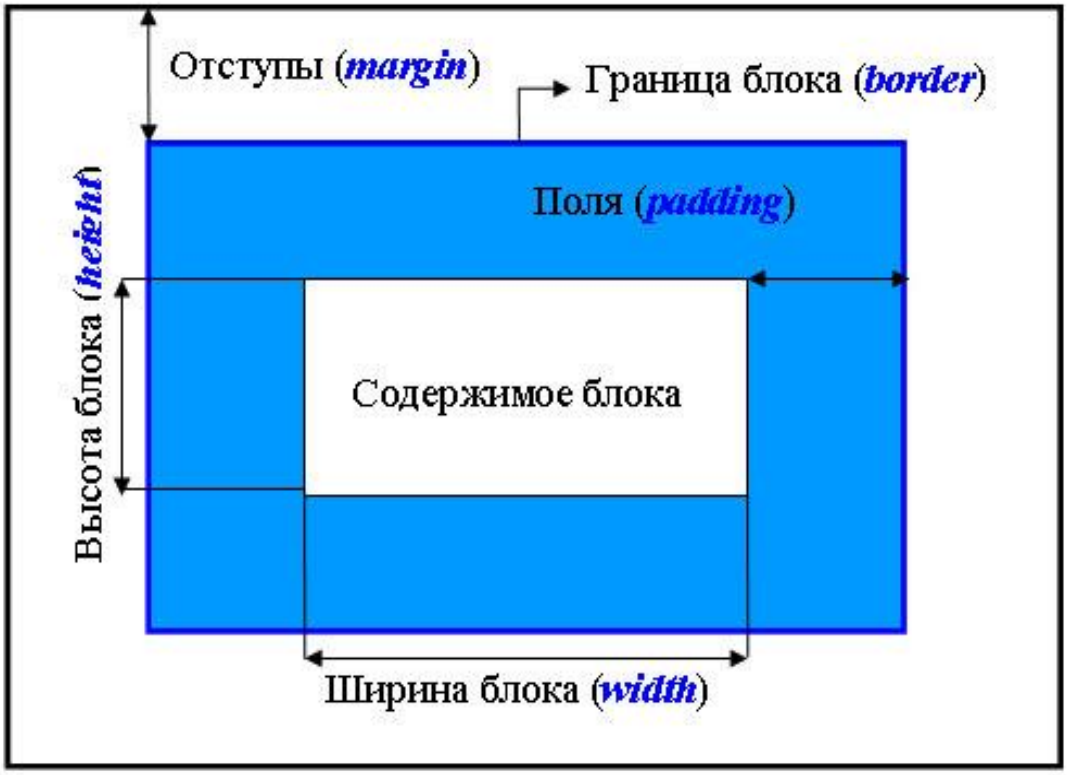
К блочным элементам относятся теги:

*** ,<dl>,<h1>...<h6>,<hr>,<ol>,<p>,<table>,<ul>***

Блок представляет собой как бы отдельную структурную единицу. Блочные элементы в общем потоке располагаются последовательно один под другим.

По умолчанию два блочных элемента не могут располагаться на одной строке. Один или несколько блоков также могут располагаться внутри другого (*родительского*) блочного элемента.

В CSS блоки создаются на основе элементов HTML и имеют следующую структуру.



Блочный элемент имеет границу и содержимое: это может быть текстовая информация, фотографии, логотип фирмы и т.п., называемое контентом (англ. content — содержание).

Границы - это линии вокруг полей элемента на одной, двух, трёх или всех четырёх его сторонах. Границе блока при помощи свойства **border** можно придать необходимое оформление: задать толщину, цвет и стиль линий. По умолчанию граница невидима.

Для создания рамки применяется универсальное свойство **border** одновременно задающее все эти параметры, а для создания линий на отдельных сторонах элемента можно воспользоваться свойствами **border-left, border-top, border-right и border-bottom,** соответственно устанавливающих границу слева, сверху, справа и снизу.

В примере 1 показано добавление линии слева от элемента.

Пример 1. Красная пунктирная линия

```
<html ><head>
  <meta http-equiv="Content-
Type" content="text/html; charset=utf-8" />
  <title>Линия</title>
  <style type="text/css">
   P.line {
    border-left: 1px dotted red;
    padding: 10px;}
  </style>
 </head>
 <body>
```

```
    <p class="line">Лев ревет только в том случае,
когда сообщает, что  территория принадлежит ему
или провозглашает себя царем природы.</p>
 </body>
</html>
```

Результат данного примера показан на рис. 1.

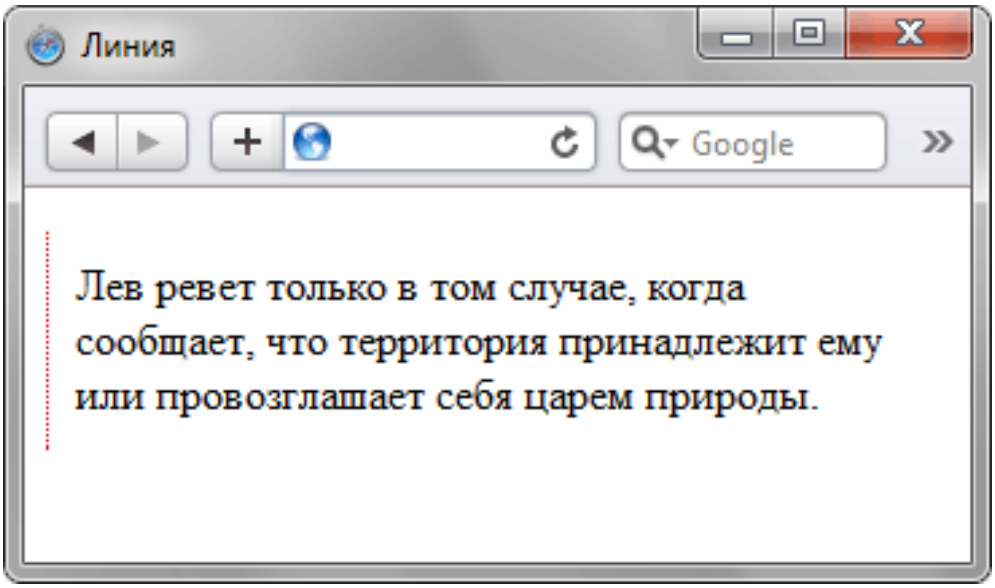


Рис. 1. Линия возле текста

Полем (свойство **padding**) будем называть расстояние от внутреннего края границы или края блока до воображаемого прямоугольника, ограничивающего содержимое блока.

Если явно не установить поля у блока, то граница будет вплотную примыкать к содержимому блока, либо отстоять от него на каком-то минимальном расстоянии. С другой стороны, поля могут иметь совершенно конкретные размеры, которые указывает разработчик.

А) Использование отступов

Вокруг блока - за его границей существуют пустые, ничем не занятые области, называемые отступами (свойство **margin**). Отступы - это расстояния от границы блока, до ближайших элементов, или, если их нет, то до краев окна браузера. Отступы также, как и поля, по умолчанию отсутствуют, либо имеют минимальную ширину, автоматически определяемую браузером.

Для отступов характерны следующие особенности.

1. Отступы прозрачны, на них не распространяется цвет фона или фоновая картинка, заданная для блока. Однако если фон установлен у родительского элемента, он будет заметен и на отступах.

2. Отступы в отличие от полей могут принимать отрицательное значение, это приводит к сдвигу всего блока в указанную сторону. Так, если задано **margin-left: -10px**, это сдвинет блок на десять пикселов влево.

3. Для отступов характерно явление под названием «схлопывание», когда отступы у близлежащих элементов не суммируются, а объединяются меж собой.

4. Отступы, заданные в процентах, вычисляются от ширины контента блока. Это касается как вертикальных, так и горизонтальных отступов.

В примере 2 показано схлопывание отступов и их прозрачность.

Пример 2. Использование отступов

```
<html>
 <head>
  <meta http-equiv="Content-
Type" content="text/html; charset=utf-8" />
  <title>Отступы</title>
  <style type="text/css">
   .layer1, .layer2 {
    background: #F2EFE6;
    border: 1px solid #B25538;
    padding: 10px;
    margin: 20px;
   }
  </style>
 </head>
 <body>
  <div class="layer1">Лев ревет только в том
случае, когда сообщает, что
   территория принадлежит ему или провозглашает
себя царем природы.</div>
  <div class="layer2">Охотничий участок льва
может иметь длину и ширину
   до тридцати километров.</div>
 </body>
</html>
```

Результат данного примера показан на рис. 2. Обратите внимание, что расстояние между блоками равно 20 пикселов, а не 40, которые получаются суммированием верхнего и нижнего отступа у блоков. Это происходит за счёт эффекта «схлопывания», при котором близлежащие отступы объединяются.

Если один из отступов отрицательный- происходит складывание отступов по правилам математики:

x + (-y) = x – y здесь x и y величина прилегающих отступов элементов.

Если оба отступа отрицательны- из двух значений выбирается наибольшее по модулю, оно же и выступает в качестве отрицательного отступа между элементами.

Схлопывание не срабатывает:

1. для элементов, у которых установлено свойство padding.
2. для элементов, у которых на стороне схлопывания задана граница;
3. на элементах с абсолютным позиционированием, т.е. таких, у которых position установлено как absolute;
4. на плавающих элементах (для них свойство float задано как left или right);
5. для строчных элементов;

Также схлопывание не срабатывает при соблюдении некоторых условий:

1. для элементов, у которых значение overflow задано как auto, hidden или scroll схлопывание не действует для их дочерних элементов;
2. у элементов, к которым применяется свойство clear, не схлопывается верхний отступ с нижним отступом родительского элемента.

На рисунке хорошо видно, что цвет, задаваемый через свойство background, не выходит за пределы границы элемента и не оказывает влияние на отступы.

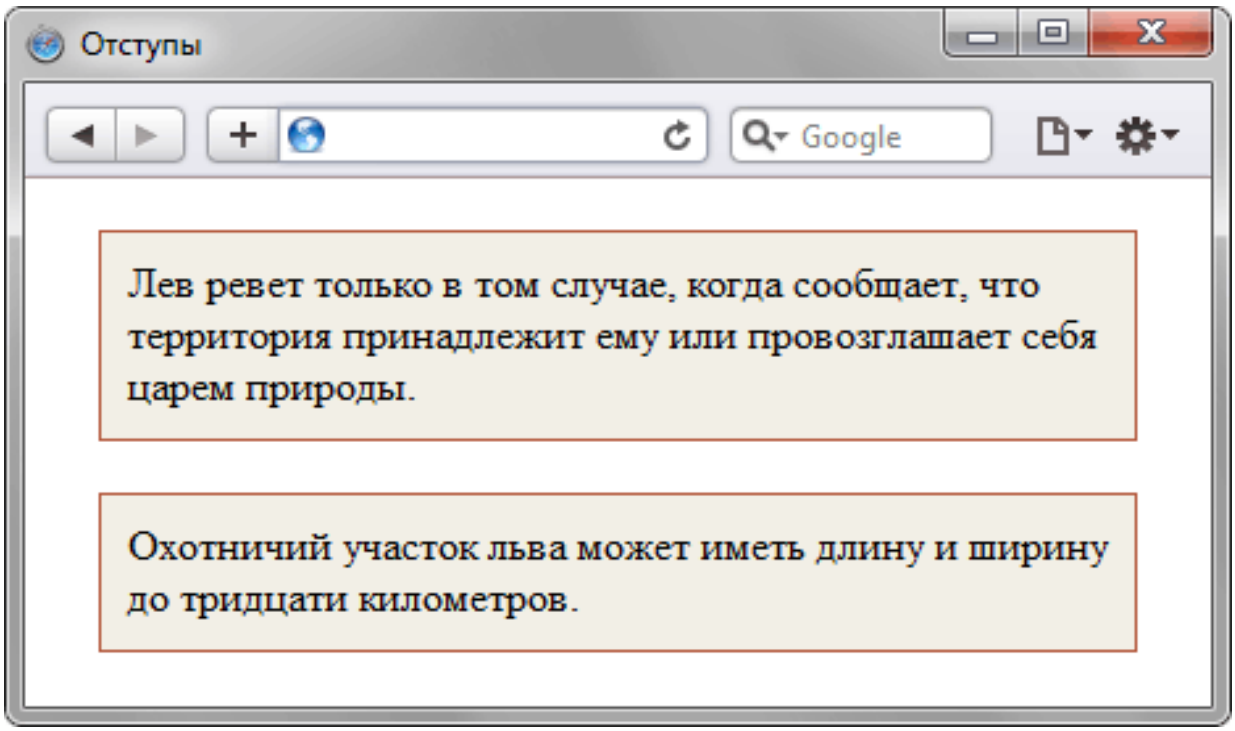


Рис. 2. Отступы в элементе

Б). Расчет ширины и высоты блока

Также для блока можно задать фиксированную ширину (свойство **width**) и высоту (свойство **height**), но лишь для его содержимого. Отступы, граница и поля туда не входят. По умолчанию, как высота, так и ширина блока подстраиваются под размеры пространства, занимаемого контентом.

Ширина блока - это комплексная величина и складывается из нескольких значений свойств:

width — ширина контента, т.е. содержимого блока;
padding-left и padding-right — поле слева и справа от контента;
border-left и border-right — толщина границы слева и справа;
margin-left и margin-right — отступ слева и справа.

Если значение **width** не задано, то оно по умолчанию устанавливается как **auto**. В этом случае ширина блока будет занимать всю доступную ширину при сохранении значений полей, границ и отступов. Под доступной шириной в данном случае подразумевается ширина контента у родительского блока, а если родителя нет, то ширина контента браузера.

Допустим, для слоя написан следующий стиль.

```
width: 300px; /* Ширина контента */
margin: 7px; /* Значение отступов */
border: 4px solid black; /* Параметры границы */
padding: 10px; /* Поля вокруг текста */
```

Ширина блока согласно этой записи будет равна 342 пиксела, эта величина получается складыванием значения ширины контента плюс отступ слева, граница слева и поле слева, плюс поле справа, граница справа и отступ справа. Суммируем все числа.

Ширина = 300 + 7 + 7 + 4 + 4 + 10 + 10 = 342

Надо отметить, что блочная модель с формированием ширины несет в себе кучу неудобств. Постоянно приходится заниматься вычислениями, когда требуется задать определенную ширину блока. Также начинаются проблемы при сочетании разных единиц измерения, в частности, процентов и пикселов. В итоге может получиться так, что общая ширина блока превысит ширину веб-страницы, что приведёт к появлению горизонтальной полосы прокрутки. Выходов из подобной ситуации два — поменять алгоритм блочной модели и воспользоваться вложенными блоками.

В браузере Internet Explorer в режиме совместимости (когда не указан доктайп) алгоритм блочной модели меняется автоматически и ширина всего блока устанавливается равной **width**. Остальные браузеры так просто не меняют алгоритм.

В) свойство box-sizing

В CSS3 есть замечательное свойство **box-sizing**. При значении **border-box** ширина начинает включать поля и границы, но не отступы. Таким образом, подключая box-sizing со значением border-box к своему стилю, мы можем задавать ширину в процентах и спокойно указывать border и padding, не боясь, что изменится ширина блока

В табл. 1 приведена поддержка браузерами этого свойства.

Табл. 1. Поддержка браузерами свойства box-sizing

Браузер	Internet Explorer	Chrome	Opera	Safari	Firefox
Версия	8.0+	2.0+	7.0+	3.0+	1.0+
Свойство	box-sizing	-webkit-box-sizing	box-sizing	-webkit-box-sizing	-moz-box-sizing

Пример 3. Ширина блока

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
  <title>Ширина блока</title>
  <style type="text/css">
   div {     width: 100%; /* Ширина */
    background: #fc0; /* Цвет фона */
    padding: 20px; /* Поля */
    -moz-box-sizing: border-box; /* Для Firefox */
    -webkit-box-sizing: border-box; /* Для Safari и
Chrome */
    box-sizing: border-box; /* Для IE и Opera */   }
  </style>
 </head>
 <body>
  <div>Ширина блока 100%</div>
 </body>
</html>
```

Данный пример будет работать во всех браузерах, указанных в табл. 3.1, однако невалиден в CSS3 из-за применения нестандартных свойств начинающихся на -moz и -webkit. Ширина блока составляет 100% с учетом значений padding. Без свойства box-sizing в браузере появится горизонтальная полоса прокрутки.

Г) Вложенные слои

Идея простая — для внешнего блочного элемента задаётся только необходимая ширина, а для вложенного блока всё остальное — поля, границы и отступы. Поскольку по умолчанию ширина блока равна доступной ширине родителя, получится, что блоки в каком-то смысле накладываются друг на друга, при этом фактическая ширина такого комбинированного элемента будет чётко задана. В примере 4 показано использование вложенных слоев.

Пример 4. Вложенные слои

```
<html>
 <head>
  <meta http-equiv="Content-
Type" content="text/html; charset=utf-8" />
  <title>Ширина блока</title>
  <style type="text/css">
   .wrap {
    width: 50%; /* Ширина */
   }
   .wrap div {
    background: #fc0;
    margin: 10px;
    padding: 20px;
    border: 1px solid #000;   }
  </style>
 </head>
 <body>
  <div class="wrap">
   <div>Ширина слоя 100%</div>
  </div>
 </body>
</html>
```

Результат данного примера показан на рис. 3.

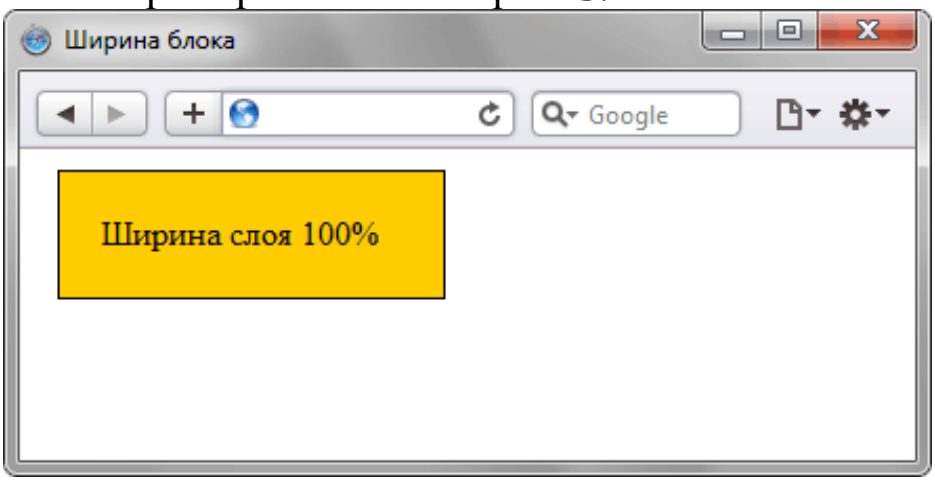


Рис. 3. Ширина блока в процентах

Преимуществом вложенных слоев является использование отступов (box-sizing их не учитывает), универсальность метода, также то, что фон по желанию можно добавлять к одному или другому слою.

Д) Высота блока

На высоту блока действуют те же правила, что и на ширину. А именно, высота складывается из значений высоты контента (**height**), полей **(padding**), границ **(border)** и отступов **(margin).** Если свойство **height** не указано, то оно считается как **auto**, в этом случае высота контента вычисляется автоматически на основе содержимого.

Вместе с тем, несмотря на схожесть принципов построения ширины и высоты, у них есть существенные различия. Это касается того случая, когда значение **width и height** не указано, тогда по умолчанию оно принимается как **auto**. Для ширины блока — это максимально доступная ширина контента, а для высоты блока — это высота контента.

Также для ширины блока известна ширина родителя, даже если она не указана явно. Это позволяет устанавливать значение width в процентах. Использование же процентов для height ни к чему не приведёт, потому что высота родителя не вычисляется и её надо указывать.

С высотой связана ещё одна особенность — при превышении содержимого блока его размеров при заданной высоте, содержимое начинается отображаться поверх блока (рис 4).

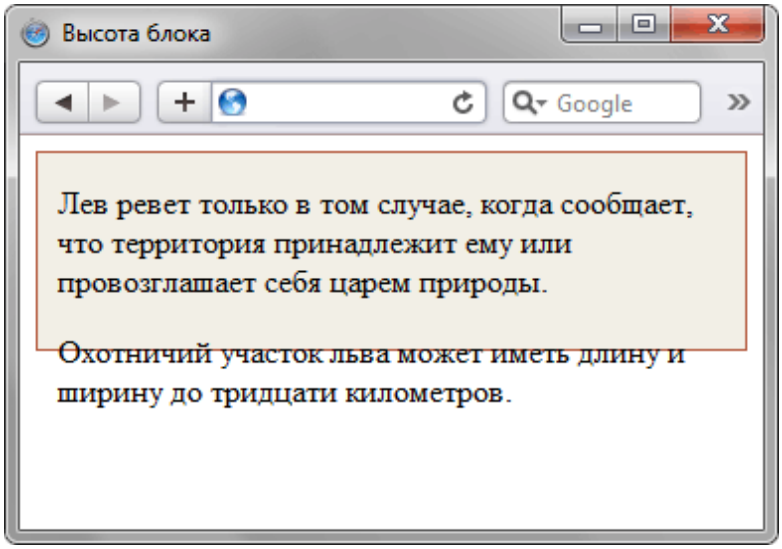
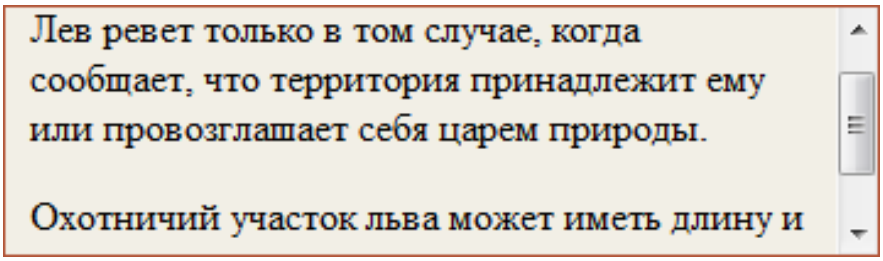


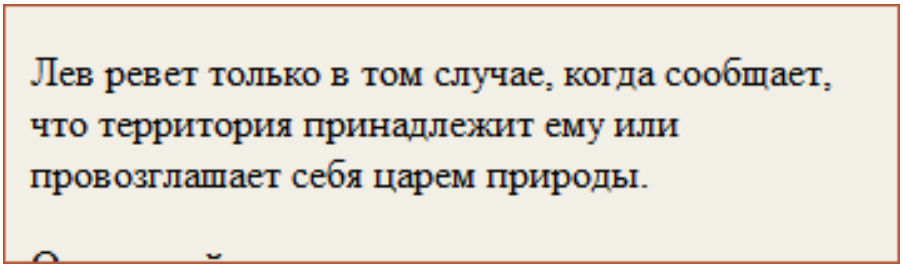
Рис. 4 Превышение размеров блока

Чтобы избежать подобных неприятностей, высоту контента лучше не задавать, тогда высота блока будет вычисляться автоматически. Впрочем, бывают случаи, когда высота должна быть чётко указана, тогда рекомендуется к стилю добавить свойство **overflow со значением auto или hidden.**

Результат у них разный, auto добавляет полосы прокрутки автоматически, когда они требуются (рис. 5а), hidden скрывает всё, что не помещается в заданные размеры (рис. 5б).



а. Значение auto



б. Значение hidden

Рис. 5. Использование свойства overflow

4. Позиционирование блоков

«Position в CSS»- CSS позиционирование является мощнейшим инструментом форматирования информации, расположенной на странице и поможет:

- расположить элемент в любом месте на странице
- расположить элемент в любом месте относительно другого элемента
- расположить несколько элементов друг за другом и определить им приоритет отображения
- определить правила отображения текста при переполнении какого–либо элемента

Position – позволяет определить способ позиционирования элемента:

static – позиционирование не применяется. Элемент статичен.

relative – позиционирование будет производиться относительно нормального положения элемента на странице.

absolute – позиционирование элемента будет производиться в абсолютных величинах. Начало оси координат располагается в левом верхнем углу. Ось x направленна вправо, ось y – вниз.

fixed – задает жесткую позицию элемента. То есть при прокрутке страницы элемент будет прокручиваться тоже, сохраняя свою позицию, относительно окна браузера.

А)Нормальное позиционирование (STATIC)

Если для элемента свойство position не задано или его значение static, элемент выводится в потоке документа как обычно. Иными словами, элементы отображаются на странице в том порядке, как они идут в исходном коде HTML.

Свойства left, top, right, bottom если определены, игнорируются.

Б)Абсолютное позиционирование (ABSOLUTE)

При абсолютном позиционировании элемент не существует в потоке документа и его положение задаётся относительно краёв браузера. Задать этот тип можно через значение absolute свойства position. Координаты указываются относительно краёв окна браузера, называемого «видимой областью» (рис. 6).

Для режима характерны следующие особенности.

- Ширина слоя, если она не задана явно, равна ширине контента плюс значения полей, границ и отступов.
- Блок не меняет своё исходное положение, если у него нет свойств right, left, top и bottom.
- Свойства left и top имеют более высокий приоритет по сравнению с right и bottom. Если left и right противоречат друг другу, то значение right игнорируется. То же самое касается и bottom.

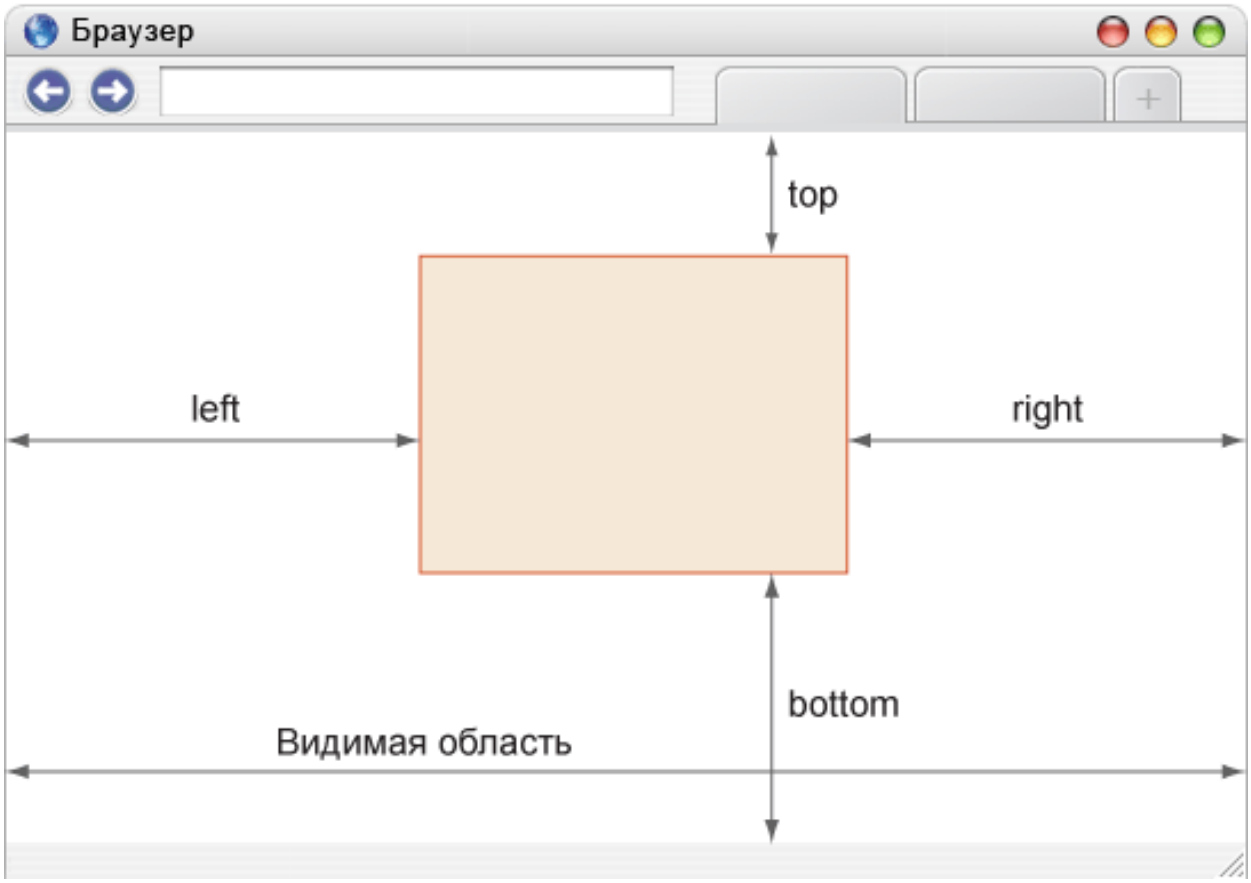


Рис. 6. Значения свойств left, right, top и bottom при абсолютном позиционировании

- Если left задать отрицательное значение, то блок уйдёт за левый край браузера, полосы прокрутки при этом не возникнет. Это один из способов спрятать элемент от просмотра. То же относится и к свойству top, только блок уйдёт за верхний край.
- Если left задать значение больше ширины видимой области или указать right с отрицательным значением, появится горизонтальная полоса прокрутки. Подобное правило работает и с top, только речь пойдёт о вертикальной полосе прокрутки.
- Одновременно указанные свойства left и right формируют ширину слоя, но только если width не указано. Стоит добавить свойство width и значение right будет проигнорировано. Аналогично произойдёт и с высотой блока, только уже участвуют свойства top, bottom и height.
- Элемент с абсолютным позиционированием перемещается вместе с документом при его прокрутке.

Свойство position со значением absolute можно использовать для создания эффекта фреймов. Кроме абсолютного позиционирования для элементов необходимо назначить свойство overflow со значением auto. Тогда при превышении контентом высоты видимой области появится полоса прокрутки. Высота и ширина «фреймов» формируется автоматически путём одновременного использования свойств left, right для ширины и top, bottom для высоты.

В) Относительное позиционирование (RELATIVE)

При относительном позиционировании, блок смещается, но его прежнее место ничего не заполняет. Обозначается как

POSITION: RELATIVE;

Bottom – задает положение нижнего края элемента:

auto – браузер выставляет нижнюю позицию самостоятельно

% – позиция задается в процентном отношении от нижнего края окна

length – нижняя позиция задается в пикселах, также допускаются отрицательные значения

Left – задает положение левого края элемента:

auto – браузер выставляет левую позицию самостоятельно

% – позиция задается в процентном отношении от левого края окна

length – левая позиция задается в пикселах, также допускаются отрицательные значения

Right – задает положение правого края элемента:

auto – браузер выставляет правую позицию самостоятельно

% – позиция задается в процентном отношении от правого края окна

length – правая позиция задается в пикселах, также допускаются отрицательные значения

Top – задает положение верхнего края элемента:

auto – браузер выставляет верхнюю позицию самостоятельно
% – позиция задается в процентном отношении от верхнего края окна
length – верхняя позиция задается в пикселах, также допускаются отрицательные значения
Дополнительные параметры позиционирования:
Vertical – align – позволяет задать выравнивание элемента по вертикали:
baseline – элемент выравнивается по родительскому элементу.
sub – элемент выравнивается, как нижний индекс.
super – элемент выравнивается, как верхний индекс.
top – элемент выравнивается по уровню самого высокого элемента в строке
text – top – вершина элемента выравнивается по вершине шрифта родительского элемента.
middle – элемент выравнивается по середине.
bottom – элемент выравнивается по уровню самого низкого элемента.
text – bottom – нижняя часть элемента выравнивается по размеру самой маленькой буквы текста.
% – выравнивает элемент в % от значения параметра "line – height". Допускаются отрицательные значения.
Overflow – задает инструкции, выполняемые в случае переполнения элементом допустимой области родительского элемента:
visible – содержимое будет выводиться даже в случае переполнения
hidden – содержимое будет обрезанно по границе допустимого заполнения
scroll – содержимое будет обрезанно по границе допустимого заполнения, но браузер создаст полосу прокрутки для просмотра содержимого целиком
auto – ситуация будет определена браузером
z – index – задает уровень приоритета элемента, что позволяет располагать одни элементы позади других.
Значения z – index могут быть отрицательными.

5 Свойство FLOAT

Данное свойство определяет, будет ли блок плавающим и в какую сторону он будет перемещаться. Свойство может принимать следующие значения.

left - структурный блок перемещается влево. Остальное содержимое документа будет выводиться вдоль правой стороны блока, начиная с самого верха.

right - структурный блок перемещается вправо. Остальное содержимое документа выводится вдоль левой стороны блока, начиная с самого верха.

none - блок не перемещается (значение по умолчанию).

Плавающие элементы достаточно активно применяются при вёрстке веб-страниц и служат для реализации этих и не только задач:

- обтекание картинок текстом;
- создание врезок;
- горизонтальные меню;
- колонки.

На рис. 7 показан результат применения разных значений на изображение рядом с текстом.

Сам HTML-код остаётся практически неизменным и привёден в примере 4.

Пример 4. Использование float

```
..
 <body>
   <div>
   <img src="images/figure.jpg" style="float: left"
 />
    Lorem ipsum dolor sit amet, consectetuer
 adipiscing elit,
    sed diem nonummy nibh euismod tincidunt ut
 lacreet dolore
    magna aliguam erat volutpat. Ut wisis enim ad
 minim veniam,    quis nostrud exerci tution
 ullamcorper suscipit lobortis nisl    ut aliquip ex
 ea commodo consequat.
   </div>
  </body>
 </html>
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

a. Обтекания нет или float: none

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

б. Для картинки установлено float: left

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat. Ut wisis enim ad minim veniam, quis nostrud exerci tution ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

в. Для картинки установлено float: right

Рис 7. Обтекание картинки текстом

По умолчанию блочные элементы выстраиваются по вертикали один под другим, но при помощи свойства float их можно заставить располагаться рядом по горизонтали. При этом требуется установить ширину блоков и задать для них float. Если ширина не указана, она будет равна содержимому блока с учётом полей и границ. В примере 8 взят каталог товаров, созданный в предыдущем разделе с помощью строчно-блочных элементов, и переделан под использование float. Чтобы блоки были заметны, фон веб-страницы установлен серым.

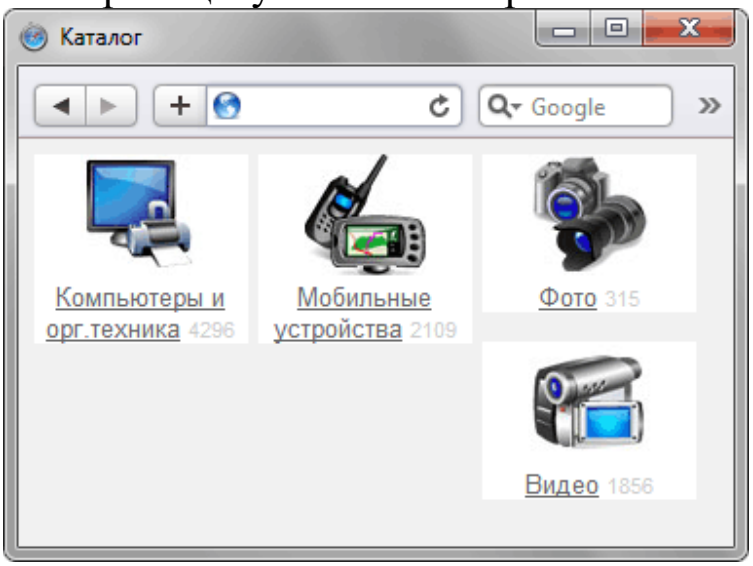


Рис. 8. Расположение слоев при использовании свойства float

Из-за разного текста в подписи высота блоков также получается разной, из-за чего некоторые блоки «цепляются» за другие и не переходят на другую строку. Здесь может помочь установка высоты всех блоков через свойство height, например 100px или возврат к использованию **display: inline-block.**

А) Отмена обтекания

Обтекание — это мощный инструмент вёрстки, применяемый для выравнивания и упорядочивания элементов. Однако, чтобы держать этот инструмент под контролем, необходим противовес, без которого огромный потенциал float сужается до пары узких задач. Речь идёт об отмене обтекания с помощью разных методов. Перечислим четыре наиболее популярных.

1. Ширина элемента

Если плавающий элемент будет занимать всю доступную ширину, то остальные элементы, следующие за ним, будут начинаться с новой строки. Для этого надо включить свойство width со значением 100%.

Этот метод применяется редко, поскольку ширину нельзя применить к изображениям, к тому же он не решает проблему с высотой блока и его фоном.

2. Использование overflow

Свойство overflow управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров. Значение auto в частности, добавляет полосы прокрутки при необходимости, а hidden отображает только область внутри элемента, остальное скрывает. Кроме этого, использование overflow со значением auto, scroll или hidden отменяет действие float. Overflow одно из самых популярных свойств, работающее в связке с float.

3. Свойство clear

Для отмены действия float применяется свойство clear со следующими значениями:

left — отменяет обтекание с левого края элемента (float: left). При этом все другие элементы на этой стороне будут опущены вниз, и располагаться под текущим элементом.

right — отменяет обтекание с правой стороны элемента (float: right).

both — отменяет обтекание элемента одновременно с правого и левого края. Это значение рекомендуется устанавливать, когда требуется отменить обтекание элемента, но неизвестно точно с какой стороны.

Чтобы отменить действие обтекания, свойство clear надо добавлять к элементу, идущему после плавающего. Обычно вводят универсальный класс, к примеру, clear и вставляют пустой тег с этим классом

Этот метод также является одним из самых популярных в вёрстке в силу простоты и универсальности.

4. Псевдоэлемент :after

Частое включение пустого тега со свойством clear захламляет код, особенно при активном использовании свойства float. Логично перенести всё в стили, избавившись от лишних тегов. Для этого воспользуемся псевдоэлементом :after, который в комбинации со свойством content добавляет текст после элемента. К такому тексту можно применить стилевые свойства, в частности clear. Остаётся только спрятать выводимый текст от браузера.

```
.clearleft:after {
 content: "."; /* Выводим текст после элемента
(точку) */
 clear: left; /* Отменяем обтекание*/
 visibility: hidden; /* Прячем текст */
 display: block; /* Блочный элемент */
 height: 0; /* Высота */
}
```

Какой именно выводить символ значения не имеет, он в любом случае скрывается от пользователя через visibility, но даже будучи скрытым, текст при этом занимает какую-то высоту и влияет на близлежащие элементы. Поэтому выводимый текст ещё необходимо превратить в блочный элемент и задать ему нулевую высоту.

Поскольку текст, генерируемый через псевдоэлемент :after, располагается после элемента, он с лёгкостью заменяет конструкцию .