

# Cascading Style Sheets

Спецификация CSS

<http://www.w3.org/TR/CSS21/propidx.html>

# CSS

- **CSS - Cascading Style Sheets** - каскадные таблицы стилей
- 1994 г. – термин предложен Хокон Виум Ли
- позволяет отделить **представление** от **содержания** HTML-документа
- используется для одновременного управления стилями и компоновкой множества страниц Web
- CSS **не является** ни языком программирования, ни языком разметки, это - **система правил**, которые определяют, какие элементы HTML должны быть дополнительно оформлены.
- Консорциум W3C поддерживает стандарты каскадных таблиц стилей.

# Стили

- впервые появились в HTML 4.0
- обычно хранятся в таблицах стилей
- Таблицы стилей могут быть определены
  - внутри HTML-документа,
  - в специальном файле с расширением **.css**
- можно определить несколько стилей, которые, подчиняясь существующим правилам, будут каскадно задавать один определенный стиль.

# Стили оформления:

- **стили шрифта** - для задания типа шрифта, размера и насыщенности;
- **стили текста** - для задания интервала между буквами и словами, высоты строк, горизонтального и вертикального интервала и абзацных отступов;
- **стили цвета** - для задания цвета фона и переднего плана;
- **стили рамок** - для вывода различных рамок, окружающих текстовые и графические элементы;
- **стили отступов** - для задания ширины различных отступов, окружающих текстовые и графические элементы;
- **стили фильтрации** - для применения специальных эффектов к текстовым и графическим элементам;
- **стили задания размера** - для задания высоты и ширины текстовых и графических контейнеров;
- **стили позиционирования** - для позиционирования элементов страницы в фиксированных пиксельных координатах на странице.

# Преимущества использования CSS:

- Несколько дизайнов страницы для разных устройств просмотра.
- Уменьшение времени загрузки страниц сайта за счет переноса правил представления данных в отдельный CSS-файл.
- Простота последующего изменения дизайна.
- Дополнительные возможности оформления.

# Недостатки

- Различное отображение верстки в различных браузерах (особенно устаревших).
- Необходимость на практике исправлять не только один CSS-файл, но и теги HTML и код PHP.

# Комментарии CSS

`/* текст комментария */`

`// однострочный комментарий`

Комментарии могут охватывать несколько строк, и в этом случае браузер будет игнорировать эти строки.

# Способы использования таблиц стилей

- **Линейная (in-line)** таблица стилей появляется внутри тега, к которому применяются ее объявления стилей; (внутри конкретного элемента HTML)
- **Встроенная (embedded)** таблица стилей является отдельным разделом стилей страницы Web, которая применяет свои стили ко всем определяемым на странице тегам; (внутри элемента <head> страницы HTML)
- **Внешняя (linked)** таблица стилей является внешним документом, содержащим задания стилей, которые применимы ко всем страницам, которые с ней соединены (во внешнем файле .css).

# Линейная таблица стилей

- когда особый стиль должен быть применен к единственному появлению элемента.
- Для описания используется атрибут `style` в соответствующем теге.
- Атрибут `style` может содержать любое свойство CSS.

```
<html>
...
<body>
...
  <p style="color: red; margin-left: 20px">
    Это параграф
  </p>
...
</body>
</html>
```

# Встроенная таблица стилей

- когда один документ использует единый стиль.
- определяются в разделе заголовка `<head>` с помощью тега `<style>`.

```
<html>
<head>
<style type="text/css">
  body {background-color: red}
  p {margin-left: 20px}
</style>
</head>
...
</html>
```

# Внешняя таблица стилей

- когда стиль применяется к нескольким страницам.
- позволяет изменить внешний вид всего Web-сайта, изменяя один файл.
- Каждая страница должна соединяться с таблицей стилей с помощью тега `<link>` (в разделе заголовка `<head>`)

```
<html>
<head>
<link rel="stylesheet" type="text/css"
      href="mystyle.css">
<link rel="stylesheet" type="text/css"
      href="http://www.htmlbook.ru/main.css">
</head>
...
</html>
```

# Внешняя таблица стилей

## (альтернативный вариант подключения)

- таблица стилей может быть подключена к веб-документу посредством директивы `@import`, располагающейся в этом документе между тегами `<style>` (в заголовке `<head>`)
- Все правила этой таблицы действуют на протяжении всего документа;

```
<head>
```

```
.....
```

```
<style type="text/css" media="all">  
@import url(style.css);
```

```
</style>
```

```
</head>
```

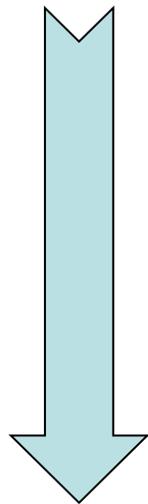
# Примечания:

- внешнюю таблицу стилей можно создать в любом текстовом редакторе;
- файл с внешней таблицей стилей не должен содержать никаких тегов HTML;
- файл с внешней таблицей стилей необходимо сохранить с расширением `.css`.

# Приоритет использования стилей

Если для элемента HTML определено более одного стиля, то все стили будут последовательно "каскадированы" в новую "виртуальную" таблицу стилей

- стили, используемые по умолчанию браузером;  
(низкий)
- стили, хранящиеся во внешней таблице;
- стили, хранящиеся во внутренней таблице стилей (внутри тега <head>);
- встроенный стиль (внутри элемента HTML).  
(высокий)



# Синтаксис правил стилей

селектор { свойство : значение ; }

- Селектор — элемент/тег HTML, который необходимо определить.
- Свойство — атрибут, значение которого задаем.
- Каждое свойство может принимать значение.

```
p {  
  margin: 10px;  
  font-family: "Times New Roman";  
  color: green;  
}
```

- CSS не чувствителен к регистру, переносу строк, пробелам и символам табуляции, поэтому форма записи зависит от желания разработчика.
- **Замечание.** Имена селекторов обязательно должны начинаться с латинского символа (a-z, A-Z) и могут содержать в себе цифры.

- Свойство и значение разделяются двоеточием и помещаются внутри фигурных скобок:

```
p {font-size:75%;}
```

- Если значение состоит из нескольких слов, то необходимо поместить значение в кавычки:

```
h1 {font-family: "lucida calligraphy"}
```

- Если требуется определить более одного свойства, то необходимо разделить свойства точкой с запятой:

```
table {  
    font-family: arial, "sans serif";  
    border-style: dotted  
}
```

- Чтобы определения стилей было удобно читать, можно каждое свойство писать на отдельной строке:

```
h2{  
    font-family: arial;  
    margin-right: 20pt;  
    color:#ffffff  
}
```

- При определении правил допускается группировка селекторов, при этом в качестве разделителя селекторов используется запятая. В примере в группу были объединены все элементы абзацев, таблиц и списков. Все эти элементы будут выведены шрифтом sans serif:

```
p,table,li {font-family: sans-serif;}
```

# Виды селекторов

- **селектор элемента;**  
`p {font-family: Garamond, serif;}`
- **селектор объединения в группу**  
`h2, p {color: red}`
- **селектор класса;**  
`.note {color:red; background:yellow; font-weight: bold;}`
- **селектор идентификатора;**  
`#paragraph1 {margin: 0;}`
- **селектор атрибута;**  
`a[href="http://www.romantiki.ru"]{font-weight:bold;}`
- **контекстный селектор;**  
`div#paragraph1 p.note {color: red;}`
- **дочерний селектор;**  
`p.note > b {color: green;}`
- **соседний селектор;**  
`h1 + p {font-size: 24pt;}`
- **селектор псевдокласса;**  
`a:active {color:yellow;}`
- **селектор псевдоэлемента.**  
`p:first-letter {font-size: 32pc;}`
- **универсальный селектор**, обозначающий любой элемент, встречающийся в ДОКУМЕНТЕ. Перед любым селектором, задающим класс или идентификатор, можно поставить знак универсального селектора  
`* {color:red;}`

# Селектор элемента (тега)

- В качестве селектора может выступать любой тег HTML, для которого определяются правила форматирования (цвет, фон, размер и т.д. )

```
тег { свойство1:значение; свойство2:значение; ... }
```

- соответствует всем элементам на странице с указанным названием (элементам `p`, в приведенном случае)

```
p {  
    font-size: 110%;  
    font-family: Verdana, Arial, Helvetica, sans-serif;  
}
```

# Селектор объединения в группу

- применить одинаковое оформление к `h2` и `p`, тогда можно было бы написать следующий CSS:

```
h2 {color: red}
```

```
p {color: red}
```

- можно сократить код CSS, группируя селекторы вместе с помощью запятой - правила в скобках применяются к обоим селекторам:

```
h2, p {color: red}
```

# Селектор класса

- соответствует всем элементам, которые имеют атрибут *class* с указанным значением.

```
тег.имя_класса {свойство1: значение; свойство2: значение; ...}
```

- Чтобы указать в коде HTML, что тег используется с определенным стилем, к тегу добавляется атрибут

```
class="имя_класса"
```

- Можно использовать классы и без указания тега.

```
.имя_класса { свойство1: значение; свойство2: значение; ... }
```

- При такой записи, класс можно применять к любому тегу.

# Пример

```
<style type="text/css">
  p {font-size: 110%;
     font-family: Verdana, Arial, Helvetica, sans-serif;
     }
  p.Color {color: blue;}
  .left {margin-left: 40pt;}
</style>
</head>
<body>
  <p>Пример использования селектора тегов.</p>
  <p class="Color">Пример использования класса.</p>
  <table class="left" border="1">
    <tr><td>Эта таблица будет иметь
      <span class="Color"> внешний отступ</span>,
      равный 40 пунктам. </td></tr>
  </table>
  <p class="left">Этот параграф будет иметь внешний отступ,
    равный 40 пунктам.</p>
```

# Селектор ID (Идентификатор)

- соответствует элементу, который имеет атрибут *id* с указанным значением
- определяет уникальное имя элемента, которое используется для изменения его стиля и обращения к нему через скрипты, что позволяет управлять стилем элемента динамически.

```
#Имя_идентификатора { свойство1: значение;  
    свойство2: значение; ... }
```

- В отличие от классов, идентификаторы должны быть **уникальны** (встречаться в коде документа только один раз).
- Обращение к идентификатору происходит аналогично классам, но в качестве ключевого слова у тега используется параметр *id*, значением которого выступает имя идентификатора. Символ решетки при этом уже не указывается.
- Идентификаторы можно применять к конкретному тегу.

```
Тег#Имя_идентификатора { свойство1: значение;  
    свойство2: значение; ... }
```

# Селектор ID (Идентификатор)

```
...  
<head>  
...  
  <style type="text/css">  
    #help {  
      position: absolute;  
      left: 160px;  
      top: 50px;  
      width: 225px;  
      height: 180px;  
      background: #f0f0f0;  
    }  
  </style>  
</head>  
  
<body>  
<div id="help">Пример использования Идентификаторов </div>  
  
</body>  
...  
EXAMPLES\css\_02.html
```

# Универсальные селекторы

- можно использовать для выбора каждого элемента на странице для применения к ним стилей оформления.

```
* { описание_правил_стиля }
```

для каждого элемента на странице должна быть добавлена сплошная красная граница толщиной 1 пиксель:

```
* {border: 1px solid #FF0000;}
```

[EXAMPLES\css\\_04.html](#)

# Селекторы атрибутов

- **позволяют установить стиль тега по присутствию определенного параметра или его значения**
- поддерживаются браузером Internet Explorer начиная с версии 7.0, для корректной работы необходимо добавить правильный `<!DOCTYPE>`.

[атрибут] { описание\_правил\_стиля }

селектор[атрибут] { описание\_правил\_стиля }

В первом случае стиль применяется ко всем элементам, которые содержат указанный атрибут.  
во втором – только к определенным селекторам.

создать красную границу вокруг любого изображения (`img`), которое имеет атрибут `alt`.

```
img[alt] {border: 1px solid #FF0000;} EXAMPLES\css\_08.html
```

- **Можно выбирать элементы также и по значению атрибута, а не только по названию атрибута.**

[атрибут="значение"] { описание\_правил\_стиля }

селектор[атрибут="значение"] { описание\_правил\_стиля }

создать черную границу вокруг изображения с атрибутом `src` со значением `alert.gif`:

```
img[src="alert.gif"] {border: 1px solid #000000;}
```

# Дочерние селекторы

## (Селекторы потомков)

- для выбора только определенных элементов, которые являются потомками других определенных элементов (в дереве элементов находится прямо внутри родительского элемента).
- не поддерживаются в браузере IE 6 (и более младших версиях).

```
Селектор_1 > Селектор_2 { Описание_правил_стиля }
```

- Стиль применяется к Селектору\_2, но только в том случае, если он является дочерним для Селектора\_1.
- По своей логике дочерние селекторы похожи на контекстные селекторы. Разница между ними следующая: Стиль к дочернему селектору применяется только в том случае, когда он является прямым потомком, иными словами, непосредственно располагается внутри родительского элемента. Для контекстного селектора допустим любой уровень вложенности.

правило задает цвет текста `green`, только тех `strong` элементов, которые являются потомками элементов `p`, но не для других элементов `strong`:

```
p > strong {color: green;} EXAMPLES\css\_111.html
```

# Контекстные селекторы

- селекторы, которые работают только в определенном контексте.
- Контекстный селектор состоит из простых селекторов разделенных пробелом.

`тег1 тег2 { ... }`

- В этом случае стиль будет применяться к `тегу2`, когда он размещается внутри `тег1`.
- Не обязательно контекстные селекторы содержат только один вложенный тег. В зависимости от ситуации допустимо применять два и более последовательно вложенных друг в друга тегов.
- Более широкие возможности контекстные селекторы дают при использовании идентификаторов и классов. Это позволяет устанавливать стиль только для того элемента, который располагается внутри определенного класса.

# Контекстные селекторы

- определенные элементы, которые являются нижележащими относительно других конкретных элементов (не просто прямыми потомками, но также расположенные ниже в дереве)
- очень похожи на селекторы потомков, за исключением того, что селекторы потомков выбирают только непосредственно нижележащих, а контекстные селекторы выбирают подходящие элементы в любом месте иерархии элементов, а не только непосредственно нижележащих.

```
<div>
  <em>Привет</em>
  <p>и сразу же
    <em>Пока</em>.
  </p>
</div>
```

- элемент `div` является предком всех других элементов. Он имеет двух потомков, `em` и `p`. Элемент `p` имеет один элемент-потомок `em`.

# Контекстные селекторы

```
... <head> ...  
<style type="text/css">  
  p b {  
    font-weight: bold; color: red;  
  }  
</style>  
</head>  
<body>  
  <div>  
    <b>Жирное начертание текста</b>  
  </div>  
  <p>  
    <b>Одновременно жирное начертание текста и  
      выделенное цветом  
    </b>  
  </p>  
</body>
```

...

В данном примере показано обычное применение тега `<b>` и этого же тега, когда он вложен внутрь параграфа `<p>`. При этом меняется цвет текста

# Соседние селекторы

(Селекторы смежных одноуровневых элементов )

- позволяют выбирать определенный элемент, который следует в коде документа непосредственно после другого определенного элемента.
- Не поддерживаются в браузере IE 6 (и более младших версиях).

```
<p>Это <b>пример</b> <var>соседних</var>  
селекторов.</p>
```

Теги `<var>` и `<b>` представляют собой соседние элементы.

```
селектор_1 + селектор_2 { описание_правил_стиля }
```

- стиль применяется к селектору\_2, но только в том случае, если он является соседним для селектора\_1 и следует сразу после него.
- выбрать элементы `p`, которые следуют непосредственно после элементов `h2`, но никакие другие элементы `p`:

```
h2 + p { ... }
```

# Псевдоклассы

- используются для обеспечения стилевого оформления не для элементов, а для различных состояний элементов.

`элемент:псевдокласс { описание_правил_стиля }`

- Наиболее обычным применением, которое можно встретить, является оформление состояний ссылок.

Различные псевдо-классы и описание состояния ссылки, которое они выбирают:

**:link** - обычное состояние непосещенных ссылок по умолчанию

**:visited** – посещенные ссылки

**:hover** - ссылка, на которой в данный момент находится указатель мыши

**:active** - ссылка, на которой в данный момент происходит щелчок

**:focus** - ссылка (или поле формы, или что-то еще), в которой в данный момент находится курсор (клавиатуры)

# Псевдоклассы

Следующие правила CSS определяют что:

- по умолчанию ссылки будут синими.
- когда курсор мыши оказывается над ссылкой, используемое по умолчанию подчеркивание ссылки исчезает.
- когда ссылка будет посещена, она станет серой.
- когда ссылка активна, она становится жирной, как дополнительный признак, что что-то сейчас произойдет.

```
a:link {color: blue;}
a:visited {color: gray;}
a:hover {text-decoration: none;}
a:active {font-weight: bold;}
```

Обратите внимание, если вы не определите эти правила в том же порядке, как они показаны выше, они могут работать не так, как вы ожидаете. Это обусловлено правилом специфичности, заставляющем более поздние правила в таблице стилей переопределять более ранние правила.

выделить поле ввода, которое содержит активный мигающий курсор:

```
input:focus {
  border: 2px solid red;
  background-color: lightgray;
}
```

# Псевдоэлементы

1. позволяют оформить определенные части элементов, а не весь элемент (например, первую букву в этом элементе),
  2. позволяют вставлять содержимое перед или после определенных элементов (генерация контента с помощью CSS)
- не поддерживаются в IE 6 (и более младших версиях).
  - не могут применяться к внутренним стилям, только к таблице связанных или глобальных стилей.

**селектор:псевдоэлемент { описание\_правил\_стиля }**

- **first-letter** – определяет стиль первого символа в тексте элемента, к которому добавляется. Позволяет создавать в тексте буквицу и выступающий инициал.
- **first-line** – определяет стиль первой строки блочного текста.
- **after** – вставка назначенного контента после элемента, работает совместно со стилевым свойством `content`, который определяет содержимое для вставки. Не поддерживаются браузером Internet Explorer ни в одной его версии.
- **before** – по своему действию `before` аналогичен псевдоэлементу `after`, но вставляет контент до элемента.

# Псевдоэлементы

- создать буквицу в начале каждого параграфа документа:

```
p:first-letter {  
  font-weight: bold;  
  font-size: 250%;  
  background-color: red;}
```

- сделать первую строку каждого параграфа жирной:

```
p:first-line {  
  font-weight: bold;}
```

- вставка декоративных изображений после каждой ссылки на страницу:

```
a:after{ content: " " url(flower.gif);}
```

Можно использовать функцию `attr()` для вставки значений атрибутов элементов после элемента.

- Вставить в скобках целевой адрес каждой ссылки в документе :

```
a:after{ content: "(" attr(href) " "};}
```

Такие правила подходят для таблиц стилей печати.

# Сокращенная запись CSS

- Можно объединить несколько связанных свойств CSS в одно свойство.
- правило для полей (сокращения для отступов и границ работают таким же образом):

```
div.foo {  
  margin-top: 1px;  
  margin-right: 1.5px;  
  margin-bottom: 2px;  
  margin-left: 2.5px;  
}
```

Это правило можно записать короче:

```
div.foo {  
  margin: 1px 1.5px 2px 2.5px;  
}
```

# Задание менее четырех значений для сокращенного свойства

- Одно значение применяется ко всем четырем сторонам

```
margin: 2px;
```

- Первое значение применяется к верху и низу, второе к левому и правому краю

```
margin: 2px 5px;
```

- Первое и третье значения применяются к верху и низу соответственно, второе значение применяется к левому и правому краю

```
margin: 2px 5px 1px;
```

- Значения применяются к верху, правому краю, низу, и левому краю

```
margin: 1px 1.5px 2px 2.5px;
```

# Сокращения для шрифта

- определить размер шрифта, толщину, стиль, семейство и высоту строки.

```
font-size: 1.5em;
```

```
line-height: 200%;
```

```
font-weight: bold;
```

```
font-style: italic;
```

```
font-family: Georgia, "Times New Roman",  
    serif
```

- с помощью одной строки:

```
font: 1.5em/200% bold italic Georgia, "Times  
    New Roman", serif;
```

# Сокращение для фона

- определить цвет фона, фоновое изображение, повторение изображения и позицию изображения.

```
background-color: #000000;
```

```
background-image: url(image.gif);
```

```
background-repeat: no-repeat;
```

```
background-position: top left;
```

- **с помощью одной строки:**

```
background:#000000 url(image.gif) no-repeat  
top left;
```

# Сокращения для списков

- задать значения типа маркера списка, позиции и изображения.

```
list-style-type: circle;
```

```
list-style-position: inside;
```

```
list-style-image: url(bullet.gif);
```

- **Это эквивалентно следующему:**

```
list-style: circle inside url(bullet.gif);
```

# Фундаментальные концепции CSS

- **Наследование** связано с тем, как элемент в разметке HTML наследует свойства своих элементов-предков (в которых он содержится) и передает их своим потомкам,
- **Каскадирование** имеет дело с объявлениями CSS, которые применяются к документу, и как конфликтующие правила переопределяют друг друга.

# Наследование в CSS

- Каждый элемент в документе HTML будет наследовать все наследуемые свойства своего предка, за исключением корневого элемента (html), который не имеет предка.
- Наследуются не все свойства CSS (таблица свойств в Спецификации CSS (<http://www.w3.org/TR/CSS21/propidx.html>)).
- ! значения, заданные в виде процентных величин, не наследуются.

```
<h1>Этот заголовок <em>очень важен</em>!</h1>
```

Если элементу `em` не присвоен цвет, то он унаследует цвет своего предка, т.е. элемента `h1`. Для задания стиля отображения элементов по умолчанию, достаточно задать стиль элемента `body`.

- Для свойств, которые не наследуются по умолчанию, можно определить принудительное наследование, используя ключевое слово **inherit**. (IE не поддерживает)

Например, следующее правило заставит все параграфы наследовать все свойства фона от своих предков:

```
p {background: inherit;}
```

# Каскадирование

- механизм, который управляет конечным результатом, в случае когда несколько конфликтующих объявлений CSS применяются к одному элементу.
- Три основные концепции, которые управляют порядком, в котором применяются объявления CSS:
  1. Важность (наиболее значимая)
  2. Специфичность
  3. Порядок исходного кода
- Если два объявления имеют одинаковую важность, специфичность правил определяет, какое из них будет применяться.
- Если правила имеют одинаковую специфичность, то порядок исходного кода управляет результатом.

# Важность

**Важность** объявления CSS зависит от того, где оно определено. Конфликтующие объявления будут применяться в следующем порядке, более поздние будут переопределять предыдущие:

1. Встроенные таблицы стилей браузера пользователя
  2. Обычные объявления в таблицах стиля пользователя (Не все браузеры поддерживают, могут быть очень полезны для пользователей с некоторыми типами функциональных недостатков)
  3. Обычные объявления в таблицах стиля автора (разработчика)
  4. Важные объявления в таблицах стиля автора
  5. Важные объявления в таблицах стиля пользователя
- Для того чтобы превратить обычное объявление в **важное** за ним необходимо разместить директиву **!important**.
  - Важные объявления в таблице стилей пользователя будут перекрывать все остальное.

```
* { font-family: "Comic Sans MS" !important; }
```

# Специфичность

- **Специфичность** определяют как меру того, насколько конкретным является селектор некоторого правила.
- Селектор с низкой специфичностью может соответствовать многим элементам (такой как `*`, который соответствует каждому элементу в документе), в то время как селектор с высокой специфичностью может соответствовать только одному элементу на странице (такой как `#nav`, который соответствует только элементу с `id` равным `nav`).
- Если два или больше объявлений конфликтуют за данный элемент, и все объявления имеют одинаковую важность, то приоритет в правиле будет иметь объявление с наиболее специфичным селектором.

# Порядок исходного кода

- Если два объявления влияют на один и тот же элемент, имеют одинаковую важность и одинаковую специфичность, то окончательное решение определяет порядок исходного кода. **Объявление, которое появляется позже в таблицах стилей будет переопределять те, которые встречаются раньше.**
- Если имеется единственная внешняя таблица стилей, то **объявления в конце файла будут переопределять объявления, которые встречаются раньше в файле, если возникает конфликт.**
- Конфликтующие объявления могут также возникать в различных таблицах стилей. В этом случае **порядок, в котором присоединяются таблицы стилей, включаются или импортируются, управляют тем, какое объявление будет применяться.**

# Пример таблицы стилей:

```
h2 {
  font-size: 1.75em;
  color: #469;
}

#container {
  padding: 0;
}

#col_r_content, #col_l_content {
  margin: 10px;
}

p#paragraph1 {
  margin: 0;
}
```

```
#masthead img {
  float: left;
  margin: 0;
  padding: 0;
}

#navigation a:hover {
  color: #000;
  text-decoration: none;
  border: 1px solid #ed9;
  background-color: #ed9;
}

.style_italic {
  font-style: italic;
}
```

# CSS-свойства для работы со шрифтами

Свойство	Значение	Описание
<b>font-family</b>	<i>имя_шрифта</i>	Задаёт список шрифтов
<b>font-style</b>	normal italic oblique	Нормальный шрифт Курсив Наклонный шрифт
<b>font-variant</b>	normal small-caps	Капитель (особые прописные буквы)
<b>font-weight</b>	normal lighter bold bolder 100-900	Нормальная жирность Светлое начертание Полужирный Жирный 100-светлый шрифт, 900-самый жирный
<b>font-size</b>	normal pt px %	Нормальный размер Пункты Пиксели Проценты

# CSS-свойства для работы с текстом

Свойство	Значение	Описание
<b>line-height</b>	normal множитель точно %	Межстрочный интервал
<b>text-decoration</b>	none underline overline line-through blink	Убрать все оформление Подчеркивание Линия над текстом Перечеркивание Мигание текста
<b>text-transform</b>	none capitalize uppercase lowercase	Убрать все эффекты Начинать с прописных Все прописные Все строчные
<b>text-align</b>	left right center justify	Выравнивание текста
<b>text-indent</b>	точно %	Отступ первой строки

# CSS-свойства для работы с текстом

<b>color</b>	<i>Color</i>	Задает цвет текста
<b>direction</b>	<i>ltr</i> <i>rtl</i>	Направление текста слева направо Направление текста справа налево
<b>letter-spacing</b>	<i>normal</i> <i>length</i>	Обычный пробел между словами Фиксированный пробел между словами
<b>text-shadow</b>	<i>none</i> <i>color</i> <i>length</i>	
<b>white-space</b>	<i>normal</i> <i>pre</i> <i>nowrap</i>	Задает способ обращения с пробелами внутри элемента Браузер игнорирует пробел Браузер сохраняет пробел. Текст не будет переноситься на другую строку, пока не встретится тег <code>&lt;br&gt;</code>
<b>word-spacing</b>	<i>normal</i> <i>length</i>	Увеличивает или уменьшает пробел между словами

# CSS-свойства для работы с цветами

Свойство	Значение	Описание
<b>color</b>	<i>Цвет</i>	Устанавливает цвет текста
<b>background-color</b>	<i>Цвет</i> transparent	Цвет фона Прозрачный
<b>background-image</b>	<i>URL</i> none	Фоновый рисунок Отсутствует
<b>background-repeat</b>	repeat repeat-x repeat-y no-repeat	Повторяемость фонового рисунка
<b>background-attachment</b>	scroll fixed	Прокручиваемость фона вместе с документом
<b>background-position</b>	% <i>Пиксели</i> top center bottom left right	Начальное положение фонового рисунка

# Цвет

- **По его названию.** Браузеры поддерживают некоторые цвета по их названию.
- **По шестнадцатеричному значению.** Цвет можно устанавливать по его шестнадцатеричному значению, как и в обычном HTML.
  - Допустимо использовать сокращенную запись, вроде #fc0. Она означает, что каждый символ дублируется, в итоге получим #ffcc00.
- **С помощью RGB.** Можно определить цвет, используя значения красной, зеленой и синей составляющей в десятичном исчислении. Значение каждого из трех цветов может принимать значения от 0 до 255. Также можно задавать цвет в процентном отношении

Black	#000000	rgb(0,0,0)
Maroon	#800000	rgb(128,0,0)
Green	#008000	rgb(0,128,0)
Olive	#808000	rgb(128,128,0)
Navy	#000080	rgb(0,0,128)
Purple	#800080	rgb(128,0,128)
Teal	#008080	rgb(0,128,128)
Silver	#c0c0c0	rgb(192,192,192)
Gray	#808080	rgb(128,128,128)
Red	#FF0000	rgb(255,0,0)
Lime	#00FF00	rgb(0,255,0)
Yellow	#FFFF00	rgb(255,255,0)
Blue	#0000FF	rgb(0,0,255)
Fuchsia	#FF00FF	rgb(255,0,255)
Aqua	#00FFFF	rgb(0,255,255)
White	#FFFFFF	rgb(255,255,255)

# Псевдоклассы для работы с ссылками

Свойство	Описание
<code>a:link</code>	Определяет стиль для обычной непосещенной ссылки.
<code>a:visited</code>	Определяет стиль для посещенной ссылки.
<code>a:active</code>	Определяет стиль для активной ссылки. Активной ссылка становится при нажатии на нее.
<code>a:hover</code>	Определяет стиль для ссылки при наведении на нее мышью.

# CSS-свойства для работы со списками

Свойство	Значение	Описание
<b>list-style</b>	disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha none	Вид маркера. Первые три используются для создания маркированного списка, а остальные – для нумерованного.
<b>list-style-image</b>	none URL	Устанавливает символом маркера любую картинку.
<b>list-style-position</b>	outside inside	Выбор положения маркера относительно блока строк текста.

# Единицы измерения

- **Относительные единицы** определяют размер элемента относительно значения другого размера. Обычно используют для работы с текстом, либо когда надо вычислить процентное соотношение между элементами.

**em** - высота шрифта текущего элемента

**ex** - высота символа x

**px** – пиксель (элементарная точка, отображаемая монитором или другим подобным устройством)

**%** - процент

- **Абсолютные единицы** не зависят от устройства вывода, применяются реже, чем относительные и, как правило, при работе с текстом.

**in** - дюйм (1 дюйм равен 2,54 см)

**cm** – сантиметр

**mm** – миллиметр

**pt** - пункт (1 пункт равен 1/72 дюйма)

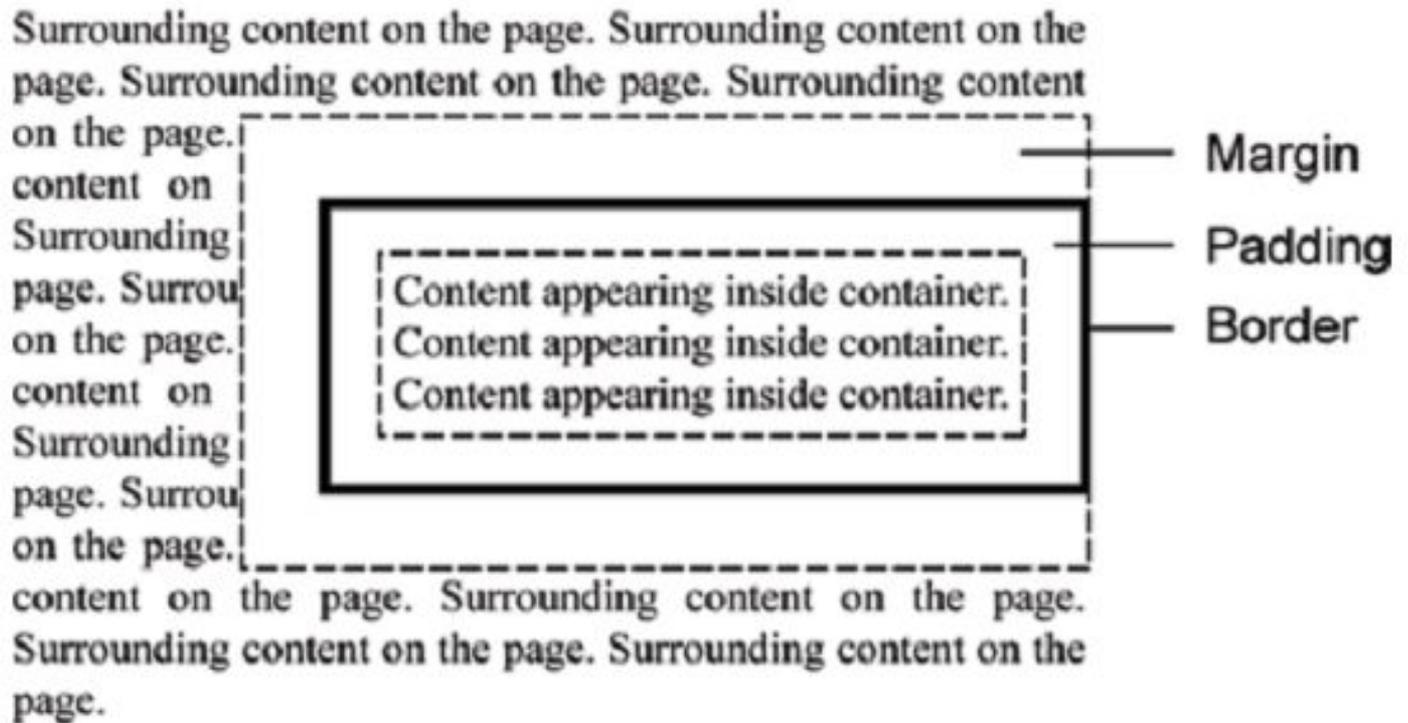
**pc** - пика (1 пика равна 12 пунктам)

# Параметры фона

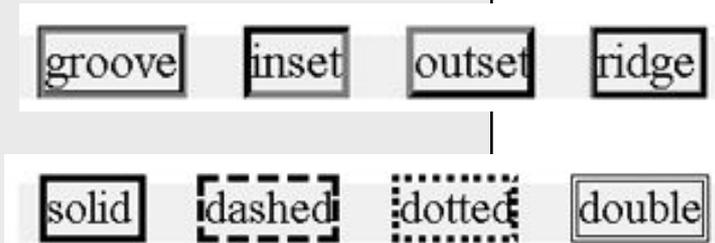
<b>background</b>	Служит для задания всех параметров фона в одном объявлении	background-color background-image background-repeat background-attachment background-position
<b>background-attachment</b>	фиксированное расположение изображения или перемещающееся вместе с остальной страницей	scroll fixed
<b>background-color</b>	Задаёт цвет фона элемента	color-rgb, color-hex, color-name, transparent
<b>background-image</b>	Задаёт в качестве фона изображение	url none
<b>background-position</b>	Задаёт начальное положение фонового изображения	top left, top center, top right, center left, center center, center right, bottom left, bottom center, bottom right, x-% y-%, x-pos y-pos
<b>background-repeat</b>	Определяет, будет ли и каким образом повторяться фоновое изображение	repeat repeat-x repeat-y no-repeat

# Поля, границы и отступы

- поля (padding)
- границы (border)
- отступы (margin)



# Стили и свойства границ

<p><code>border-style</code>  <code>border-top-style</code>  <code>border-right-style</code>  <code>border-bottom-style</code>  <code>border-left-style</code></p>	<p>dashed  dotted  double  groove  inset  none  outset  ridge  solid</p>	 <p>The image shows two rows of border style examples. The top row contains four boxes: 'groove' (a 3D channel), 'inset' (a 3D shadow), 'outset' (a 3D raised effect), and 'ridge' (a 3D ridge). The bottom row contains four boxes: 'solid' (a simple solid line), 'dashed' (a line of short dashes), 'dotted' (a line of small dots), and 'double' (two parallel solid lines).</p>
<p><code>border-width</code>  <code>border-top-width</code>  <code>border-right-width</code>  <code>border-bottom-width</code>  <code>border-left-width</code></p>	<p>thin  medium  thick  px</p>	
<p><code>border-color</code>  <code>border-top-color</code>  <code>border-right-color</code>  <code>border-bottom-color</code>  <code>border-left-color</code></p>	<p>#000000 - #ffffff  color name  rgb(r, g, b)</p>	
<p><code>border</code></p>	<p>border: style size color</p>	

# Стили и свойства полей

- Для каждой из частей этих компонентов можно задавать одинаковый размер для всех четырех сторон контейнера, или отдельные стороны могут иметь различные значения.

<b>padding</b>	Параметр для задания всех полей в одном объявлении	padding-top padding-right padding-bottom padding-left
<b>padding-bottom</b>	Задаёт нижнее поле элемента	length %
<b>padding-left</b>	Задаёт левое поле элемента	length %
<b>padding-right</b>	Задаёт правое поле элемента	length %
<b>padding-top</b>	Задаёт верхнее поле элемента	length %

# Стили и свойства отступов

<b>margin</b>	Параметр для задания отступов в одном объявлении	margin-top margin-right margin-bottom margin-left
<b>margin-bottom</b>	Задаёт нижний отступ элемента	auto length %
<b>margin-left</b>	Задаёт левый отступ элемента	auto length %
<b>margin-right</b>	Задаёт правый отступ элемента	auto length %
<b>margin-top</b>	Задаёт верхний отступ элемента	auto length %

# Управление видимостью и положением элементов

<b>clear</b>	Задаёт стороны элемента, на которых не допускаются другие перемещаемые элементы	left right both none
<b>cursor</b>	Задаёт тип выводимого курсора	auto move text wait help ...
<b>display</b>	Определяет, как в документе будет показан элемент	none inline block list-item ...
<b>float</b>	Определяет, где в другом элементе появится изображение или текст (обтекание элементов)	left right none
<b>position</b>	Задаёт статическое, относительное, абсолютное или фиксированное положение элемента	static relative absolute fixed
<b>visibility</b>	Определяет, будет ли элемент видим или невидим	visible hidden collapse

# Задание размеров элементов

<b>height</b>	px % auto
<b>width</b>	px % auto
<b>overflow</b>	visible hidden scroll auto

Page content can appear within containers as well as flow throughout the main document. With tags such as `<div>`, `<p>`, and `<span>` to contain content, these elements can, if so chosen, be sized to various heights and widths.

`width: 125px;`  
`height: 100px;`  
**`overflow: visible`**

Page content can appear within containers as well as flow

`width: 125px;`  
`height: 100px;`  
**`overflow: hidden`**

Page content can appear within containers as

`width: 125px;`  
`height: 100px;`  
**`overflow: scroll`**

Page content can appear within containers as well as flow

`width: 125px;`  
`height: 100px;`  
**`overflow: auto`**

# Параметры позиционирования в CSS

<b>bottom</b>	Задаёт, насколько далеко нижний край элемента находится выше/ниже нижнего края родительского элемента	auto % length
<b>clip</b>	Задаёт форму элемента. Элемент вырезается по форме и выводится.	shape auto
<b>left</b>	Задаёт, насколько далеко левый край элемента находится правее/левее левого края родительского элемента	auto % length
<b>right</b>	Задаёт, насколько далеко правый край элемента находится левее/правее правого края родительского элемента	auto % length
<b>top</b>	Задаёт, насколько далеко верхний край элемента находится выше/ниже верхнего края родительского элемента	auto % length
<b>vertical-align</b>	Задаёт выравнивание элемента по вертикали	baseline sub super top text-top middle bottom text-bottom length %
<b>z-index</b>	Задаёт порядковый номер элемента в стеке	auto number

# Относительное позиционирование

```
<div style="font-size:24pt">  
  
<span style="position:relative;  
  top:-15px">Words  
</span>  
  
<span style="position:relative;  
  top:+10px">in  
</span>  
  
<span style="position:relative;  
  top:-5px">a  
</span>  
  
<span style="position:relative;  
  top:+5px">sentence.  
</span>  
</div>
```

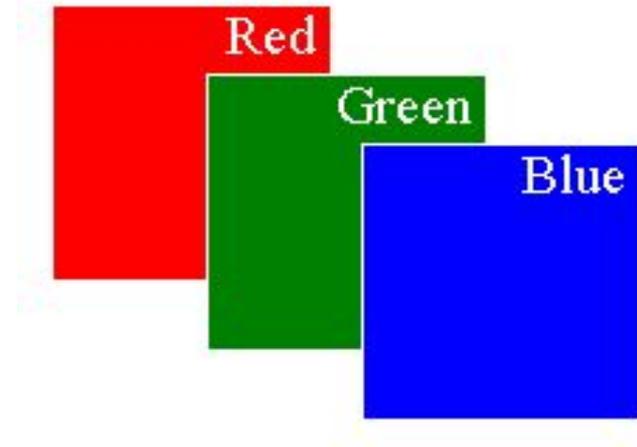
Words in a sentence.

[EXAMPLES\css\\_05.html](#)

# Разбиение элементов по слоям

```
<style type="text/css">
  .RED { position:absolute;
    width:100px; height:100px;
    left:0px; top:0px;
    background-color:red;
    border:solid 1px white;
    color:white;
    text-align:right;}
  .GREEN{position:absolute;
    width:100px; height:100px;
    left:50px; top:25px;
    background-color:green;
    border:solid 1px white;
    color:white;
    text-align:right;}
  .BLUE {position:absolute;
    width:100px; height:100px;
    left:100px; top:50px;
    background-color:blue;
    border:solid 1px white;
    color:white;
    text-align:right;
  }
</style>
...
<body>
  <div class="RED">Red</div>
  <div class="GREEN">Green</div>
  <div class="BLUE">Blue</div>
...

```



[EXAMPLES\css\\_06.html](#)

# Разбиение ЭЛЕМЕНТОВ ПО СЛОЯМ

```
<style type="text/css">
  .RED { position:absolute;
    width:100px; height:100px;
    left:0px; top:0px;
    background-color:red;
    border:solid 1px white;
    color:white;
    text-align:right;
    z-index:30;}
  .GREEN{position:absolute;
    width:100px; height:100px;
    left:50px; top:25px;
    background-color:green;
    border:solid 1px white;
    color:white;
    text-align:right;
    z-index:10;}
  .BLUE {position:absolute;
    width:100px; height:100px;
    left:100px; top:50px;
    background-color:blue;
    border:solid 1px white;
    color:white;
    text-align:right;
    z-index:3;
  }
</style>
...
```

