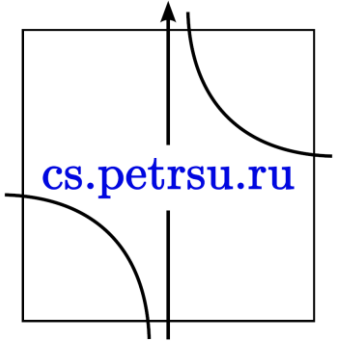
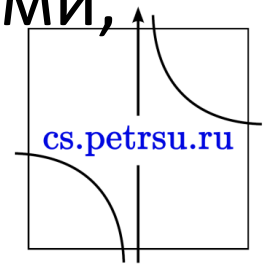


CSS

Лекция №2

Каскадные таблицы стилей

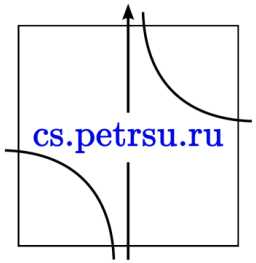
- Таблицы стилей (или каскадные таблицы стилей, CSS) – это описание правил, задающих параметры представления отдельных элементов на языке HTML.
- CSS появились одновременно с HTML 4.0 (Dynamic HTML). Сам термин «каскадные таблицы стилей» был предложен в 1994 году. Все объявления CSS называются селекторами, записываются в фигурных скобках.



Размещение каскадных таблиц

- Встраивание в тэги документа
- Внедрение
- Связывание
- Импортирование

Рассмотрим подробнее:



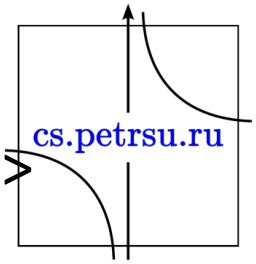
Размещение каскадных таблиц

- Свойства можно определять непосредственно с тегом – встроенный стиль.
  - Например: красный на синем

- Таблицу можно разместить между тегами <style> и </style> - внетренний/внедренный стиль.

- Таблицу стилей можно разместить в отдельном файле – внешняя таблица стилей.
  - Подключение затем выполняется так:

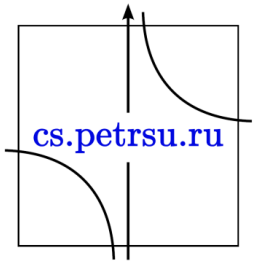
    <link rel="stylesheet" type="text/css" href="style.css">



Правило @import

- Правило @import позволяет загружать внешние таблицы стилей.
- Чтобы директива @import работала, она должна располагаться в таблице стилей (внешней или внутренней) перед всеми остальными правилами:

```
<style>
@import url(mobile.css);
@import url(https://fonts.googleapis.com/css?family=Open+Sans&
p {
   font-size: 0.9em;
   color: grey;
}
</style>
```

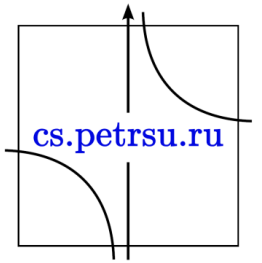


Правило @media

- HTML4 и CSS2 (2001 год).
- Позволяет указать тип носителя, для которого будет применяться указанный стиль.
- В качестве типов выступают различные устройства, например, принтер, КПК, монитор и др.

- Синтаксис:

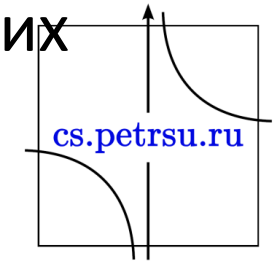
```
@media тип1 [, тип2] {
    Описание стиля
}
```



Тип носителя	Описание
all	Все типы. Это значение используется по умолчанию.
aural	Речевые синтезаторы, а также программы для воспроизведения текста вслух. Сюда, например, можно отнести речевые браузеры.
braille	Устройства, основанные на системе Брайля, которые предназначены для слепых людей.
handheld	Наладонные компьютеры и аналогичные им аппараты.
print	Печатающие устройства вроде принтера.
projection	Проектор.
screen	Экран монитора.
tv	Телевизор.

Характеристики носителя

- width - ширина области просмотра
- height - высота области просмотра
- aspect-ratio - соотношение ширины к высоте
- orientation - ориентация области просмотра
- resolution - рзрешение экрана
- color - количество бит на каждый из цветовых компонентов устройства вывода
- monochrome - количество битов на пиксель монохромного устройства
- -webkit-device-pixel-ratio - количество физических пикселей устройства на каждый CSS-пиксель



Примеры медиа запросов

- HTML

```
<link rel="stylesheet" media="screen and (color)"
href="example.css">
```

- Импорт

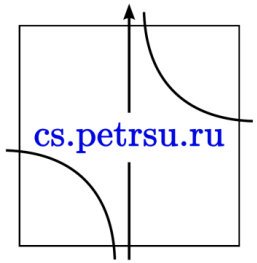
```
@import url(color.css) screen and (color);
```

- В коде стреницы

```
<style>
   @media (max-width: 600px) {
      #sidebar {display: none;}
   }
 </style>
```

- Внешняя таблица стилей

```
@media (max-width: 600px) {
   #sidebar {display: none;}
}
```



Логические операторы media

- and

  @media screen and (max-width: 600px) { /* CSS-стили */;}

  @media (min-width: 600px) and (max-width: 800px) { }

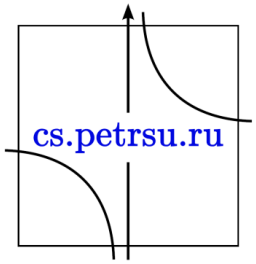
- ,

  @media screen, projection { /* CSS-стили */; }

- not

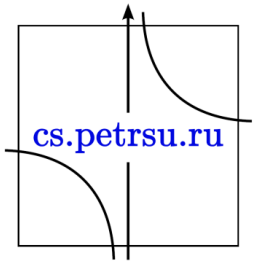
  @media not all and (monochrome) {...}

- only
  - используется, чтобы скрыть стили от старых браузеров



Метатег viewport

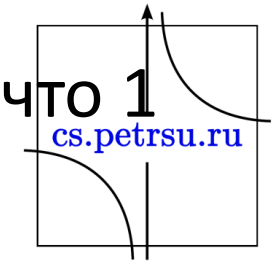
- Управления разметкой в мобильных браузерах.
  - Мобильные браузеры отображают страницы в виртуальном окне просмотра, которое обычно шире, чем экран устройства.
  - С помощью метатега viewport можно контролировать размер окна просмотра и масштаб.



Метатег viewport

<meta name="viewport" content="width=device-width, initial-scale=1.0">

- Свойство width определяет виртуальную ширину окна просмотра, значение device-width — физическую ширину устройства.
- При первой загрузке страницы свойство initial-scale управляет начальным уровнем масштабирования, initial-scale=1 означает, что 1 пиксель окна просмотра = 1 пиксель CSS.

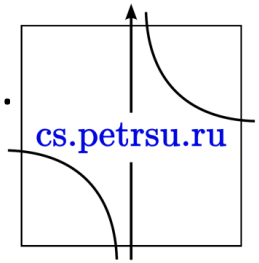


cs.petrsu.ru

Стратегии использования

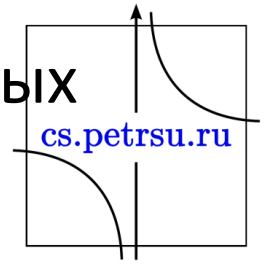
Для создания дизайна, позволяющего лучшим образом отображать сайт на различных устройствах, используют общие стратегии медиа-запросов:

1. Уменьшение количества колонок (столбцов) и постепенная отмена обтекания для мобильных устройств.
2. Использование свойства max-width вместо width при задании ширины блока-контейнера.
3. Уменьшение полей и отступов на мобильных устройствах (например, нижних отступов между заголовком и текстом, левого отступа для списков и т.п.).
4. Уменьшение размеров шрифтов для мобильных устройств.
5. Превращение линейных навигационных меню в раскрывающиеся.
6. Скрытие второстепенного содержимого на мобильных устройствах с помощью display: none.
7. Подключение фоновых изображений уменьшенных размеров.

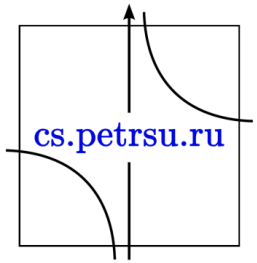


Правила построения CSS

- Каждое правило CSS из файла имеет две основные части — *селектор* и *блок объявлений.*
- ***Селектор***, расположенный в левой части правила до знака «{», определяет, на какие части документа (возможно, специально обозначенные) распространяется правило.
- ***Блок объявлений*** располагается в правой части правила. Он помещается в фигурные скобки, и состоит из одного или более *объявлений*, разделённых знаком «;». Каждое *объявление* представляет собой сочетание *свойства CSS* и *значения*, разделённых знаком «:».

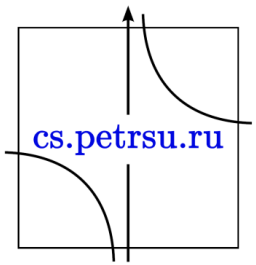


```
селектор, селектор {
  свойство: значение;
  свойство: значение;
  свойство: значение;
}
```



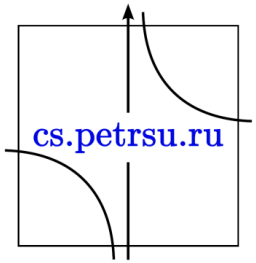
Виды селекторов

- **Универсальный селектор**
- **Селектор тегов**
- **Селектор классов**
- **Селектор идентификаторов**
- **Селектор атрибутов**
- **Селектор потомков**
- **Селектор дочерних элементов**
- **Селектор элементов одного уровня (смежные и соседние)**
- **Селектор псевдоклассов**
- **Селектор псевдоэлементов**



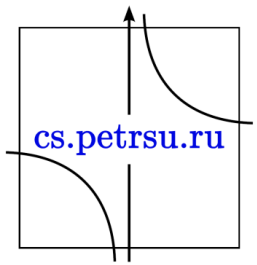
Универсальный селектор

- Соответствует любому HTML-элементу.
- Например, * {margin: 0;} обнулит внешние отступы для всех элементов сайта.
- Также селектор может использоваться в комбинации с псевдоклассом или псевдоэлементом: *:after {CSS-стили}, *:checked {CSS-стили}.



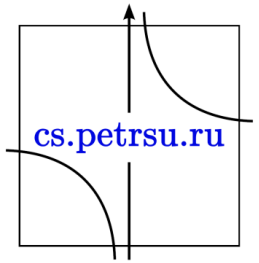
Селектор элемента

- Селекторы элементов позволяют форматировать все элементы данного типа на всех страницах сайта.
- Например, h1 {font-family: Lobster, cursive;} задаст общий стиль форматирования всех заголовков h1.



Селектор класса

- Селекторы класса позволяют задавать стили для одного и более элементов с одинаковым именем класса, размещенных в разных местах страницы или на разных страницах сайта.
- Стили, созданные с помощью класса, можно применять к другим элементам, не обязательно данного типа.
- Пример создания заголовка с классом headline:



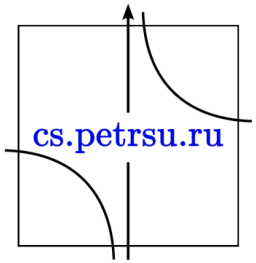
Селектор класса

- HTML

```
<h1 class="headline">Заголовок</h1>
```

- CSS

```
.headline {
    text-transform: uppercase;
    color: lightblue;
}
```

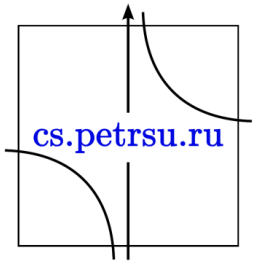


Селектор класса

- Если элемент имеет несколько атрибутов класса, их значения объединяются с пробелами.

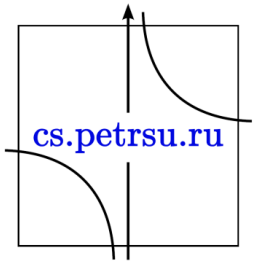
- HTML

```
<h1 class="headline post-title z1">Заголовок</h1>
```



Селектор идентификатора

- Позволяет форматировать один конкретный элемент.
- Значение id должно быть уникальным, на одной странице может встречаться только один раз и должно содержать хотя бы один символ.
- Значение не должно содержать пробелов.



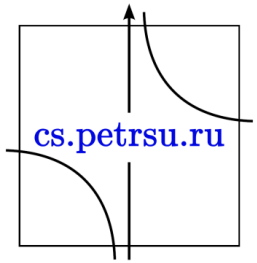
Селектор идентификатора

- HTML

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

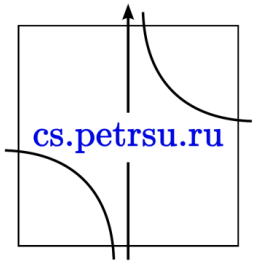
- CSS

```
#sidebar {
    width: 300px;
    float: left;
}
```



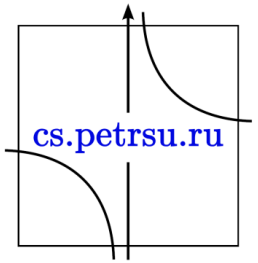
Селектор потомка

- Применяют стили к элементам, расположенным внутри элемента-контейнера.
- ul li {text-transform: uppercase;} — выберет все элементы li, являющиеся потомками всех элементов ul.
- Рассмотрим ещё несколько примеров:



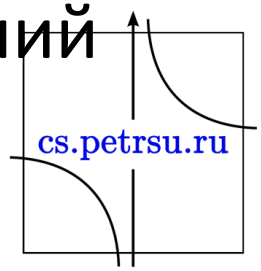
Селектор потомка

- p.first a {color: green;} —применится ко всем ссылкам, потомкам абзаца с классом first;
- p .first a {color: green;} — если добавить пробел, то будут стилизованы ссылки, расположенные внутри любого тега класса .first, который является потомком элемента <p>;
- .first a {color: green;} — данный стиль применится к любой ссылке, расположенной внутри другого элемента, обозначенного классом .first.



Селектор дочерних элементов

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



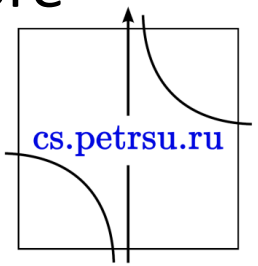
Селектор дочерних элементов

- CSS

  p > strong { } — выберет все элементы strong, являющиеся дочерними по отношению к элементу p (вложенные в p).

- HTML

  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

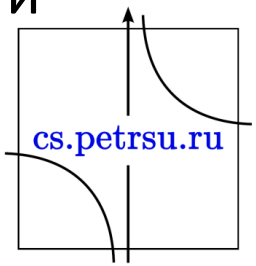


Сестринский селектор

- Сестринские отношения возникают между элементами, имеющими общего родителя. Селекторы сестринских элементов позволяют выбрать элементы из группы элементов одного уровня:

  h1 + p { } — “смежные” выберет все первые абзацы, идущие непосредственно за любым тегом <h1>, не затрагивая остальные абзацы;

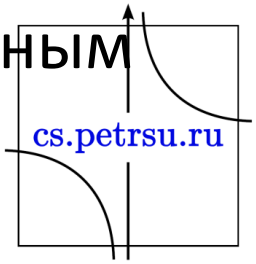
  h1 ~ p { }— выберет все абзацы, являющиеся сестринскими по отношению к любому заголовку h1 и идущие сразу после него.



Селектор атрибута

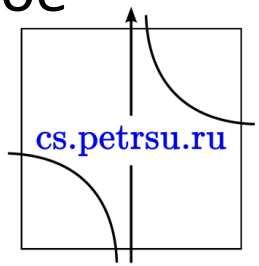
Выбирают элементы на основе имени атрибута или значения атрибута:

- [атрибут] — все элементы, содержащие указанный атрибут, [alt] — все элементы, для которых задан атрибут alt;
- селектор[атрибут] — элементы данного типа, содержащие указанный атрибут, img[alt] — только картинки, для которых задан атрибут alt;
- селектор[атрибут="значение"] — элементы данного типа, содержащие указанный атрибут с конкретным значением;



Селектор атрибута

- селектор[атрибут~="значение"] — элементы частично содержащие данное значение;
- селектор[атрибут|="значение"] — элементы, список значений атрибута которых начинается с указанного слова;
- селектор[атрибут^="значение"] — элементы, значение атрибута которых начинается с указанного значения;
- селектор[атрибут$="значение"] — элементы, значение атрибута которых заканчивается указанным значением;
- селектор[атрибут*="значение"] — элементы, значение атрибута которых содержит в любом месте указанное слово.

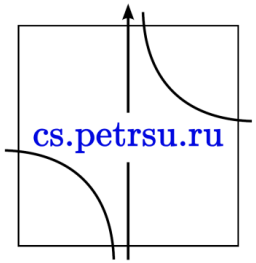


Псевдоэлементы и псевдоклассы

- Псевдоклассы применяются к различным типам элементов.
- Псевдоэлементы применяются к различным частям элементов.
- Селекторы псевдоэлементов и псевдоклассов записываются в виде:

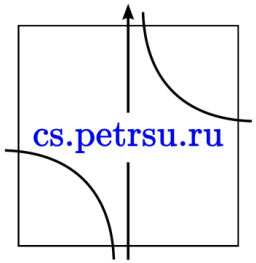
Элемент:псевдокласс

Элемент:псевдоэлемент



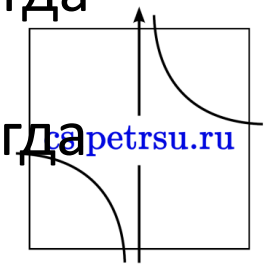
Псевдоклассы

- Это классы, фактически не прикрепленные к HTML-тегам.
- Позволяют применить CSS-правила к элементам при совершении события или подчиняющимся определенному правилу.



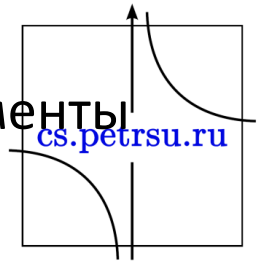
Примеры псевдоклассов

- :first-child { } – псевдокласс для элемента, являющегося первым дочерним элементом какого-либо другого элемента.
- :link { } - псевдокласс для непосещенной гиперссылки (применим к <a>)
- :visited { } - псевдокласс для посещенной гиперссылки (применим к <a>)
- :hover { } - псевдокласс применяется к элементу, когда пользователь навел на него курсор мыши, но не активировал щелчком
- :active { } - псевдокласс применяется к элементу, когда пользователь активировал его щелчком
- :focus { } - псевдокласс применяется к элементу, когда он получает фокус



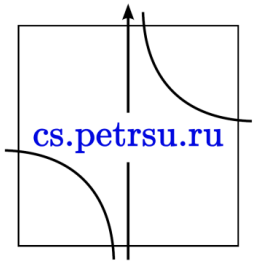
Примеры псевдоклассов

- :valid — поля формы, содержимое которых прошло проверку в браузере на соответствие указанному типу данных;
- :invalid — поля формы, содержимое которых не соответствует указанному типу данных;
- :enabled — все активные поля форм;
- :disabled — заблокированные поля форм, т.е., находящиеся в неактивном состоянии;
- :lang() — элементы с текстом на указанном языке;
- :not(селектор) — элементы, которые не содержат указанный селектор — класс, идентификатор, название или тип поля формы — :not([type="submit"]);
- :target — элемент с символом #, на который ссылаются в документе;
- :checked — выделенные (выбранные пользователем) элементы формы.



Примеры пресвдоэлементов

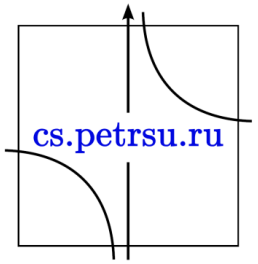
- :first-line { } - псевдоэлемент применим к любому блочному элементу и задает стиль отображения его первой строки
- :first-letter { } - псевдоэлемент применим к любому блочному элементу и задает стиль отображения его первой буквы
- :before { } - псевдоэлемент используется для вставки автоматически генерируемого содержимого перед указанным элементом
- :after { } - псевдоэлемент используется для вставки автоматически генерируемого содержимого после указанного элемента



Объединение селекторов
Комбинация селеторов

- Селекторы любых рассмотренных типов можно объединять между собой с любой сложностью.
- Стили применяются только к тем элементам, которые соответствуют сразу всем указанным селекторам

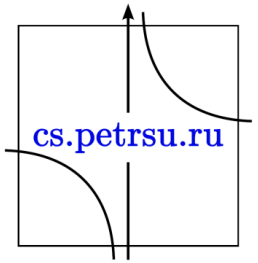
  div#main p.first a[href][title] { } — выберет все ссылки, для которых заданы атрибуты href и title, которые находятся в параграфе с классом first, который находится в блоке с идентификатором main;



Группировка селекторов

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

```
h1, h2, p, span {
    color: tomato;
    background: white;
}
```

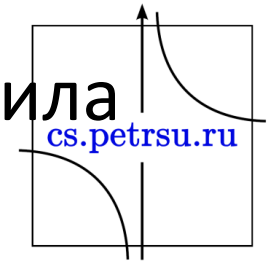


Наследование. Каскадирование.

Применение CSS к документам HTML основано на принципах *наследования* и *каскадирования*.

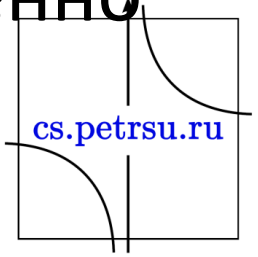
- Принцип ***наследования*** - свойства CSS, объявленные для элементов-предков, почти всегда наследуются элементами-потомками.
- Принцип ***каскадирования*** - какому-то элементу HTML одновременно поставлено в соответствие более одного правила CSS (происходит конфликт значений правил).

Чтобы разрешить конфликты, вводятся правила приоритета.



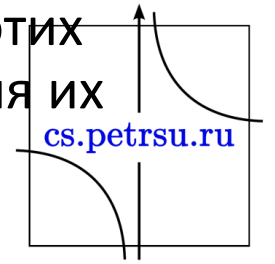
Приоритеты стилей CSS.

- Наиболее низким приоритетом обладает стиль браузера;
- Следующим по значимости является стиль, заданный пользователем браузера в его настройках;
- И наиболее высоким приоритетом обладает стиль, заданный непосредственно автором страницы.

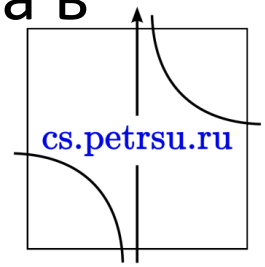


Далее приоритеты расставляются следующим образом:

- Самым низким приоритетом обладают стили, **наследуемые** в документе элементом от своих предков;
- Более высоким приоритетом обладают стили, заданные во **внешних** таблицах стилей, подключённых к документу;
- Ещё более высоким приоритетом обладают стили, заданные во **внедренной** таблице стилей (тег style).
  - Когда к элементу имеют отношение несколько селекторов - конфликты разрешаются с помощью расчёта **специфичности** каждого селектора и применения этих селекторов к данному элементу в порядке убывания их специфичностей.



- Ещё более высокий приоритет имеют **встроенные** стили заданные непосредственно у самого тега в атрибуте style.
- (!) Самым высоким приоритетом обладают стили, объявленные с помощью сопроводительного правила !important.
  - Если таких свойств несколько, то потребуется определить их специфичности по принципам, описанным выше, и применять эти свойства в порядке убывания этих специфичностей.



Специфичность селекторов

Делится на 4 группы — a, b, c, d:

- если стиль встроенный (определён как style="...", то a=1, иначе a=0) ;
- значение b равно количеству идентификаторов (иначе — id=" ", они начинаются с #) в селекторе;
- значение c равно количеству классов (class=" ", они начинаются с .), псевдоклассов (они начинаются с :, например a:hover) и селекторов атрибутов (input[type="text");
- значение d равно количеству селекторов типов элементов. После этого полученное значение приводится к числу (обычно в десятичной системе счисления).

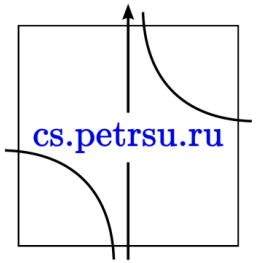
Селектор, обладающий большим значением специфичности, обладает и большим приоритетом.

Таблица расчёта специфичности:

Селектор	a, b, c, d	Число
span	0, 0, 0, 1	1
div .class	0, 0, 1, 1	11
#id .class	0, 1, 1, 0	110
div span	0, 0, 0, 2	2
.class	0, 0, 1, 0	10
#id span	0, 1, 0, 1	101

Порядок подключённых таблиц

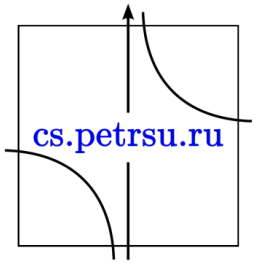
- Вы можете создать несколько внешних таблиц стилей и подключить их к одной веб-странице.
- Если в разных таблицах будут встречаться разные значения свойств одного элемента, то в результате к элементу применится правило, находящееся в таблице стилей, идущей в списке ниже.



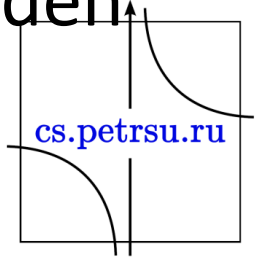
Параметры CSS, управляющие положением на странице

position: absolute | fixed | relative | static | inherit

– Устанавливает способ позиционирования элемента относительно окна браузера или других объектов на веб-странице. **absolute** указывает, что используются абсолютные координаты. **relative** указывает, что используются относительные координаты. **inherit** наследует значение родителя.

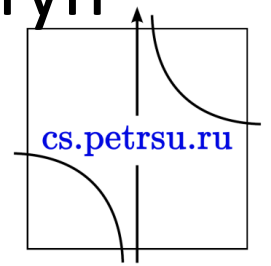


- **float** определяет, по какой стороне будет выравниваться элемент.
- **left** задает положение относительно левого края контейнера.
- **top** задает положение относительно верхнего края контейнера. Задаются в процентах или пикселях.
- **z-index** указывает на то, какой элемент должен располагаться выше приперекрытии. Измеряется в единицах.
- **visibility** определяет видимость элемента. Принимает значения visible (видимый), hidden (скрытый), inherit (наследуется от родительского элемента).



CSS блочная модель

- Описывает свойства padding и margin, которые создают поля внутри и отступы снаружи CSS блока.
- Размеры блока также могут быть увеличены за счет рамки.
- Каждый блок имеет прямоугольную область содержимого в центре, поля вокруг содержимого, рамку вокруг полей и отступ за пределами рамки.

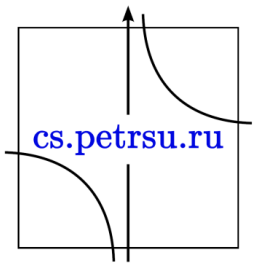


CSS блочная модель

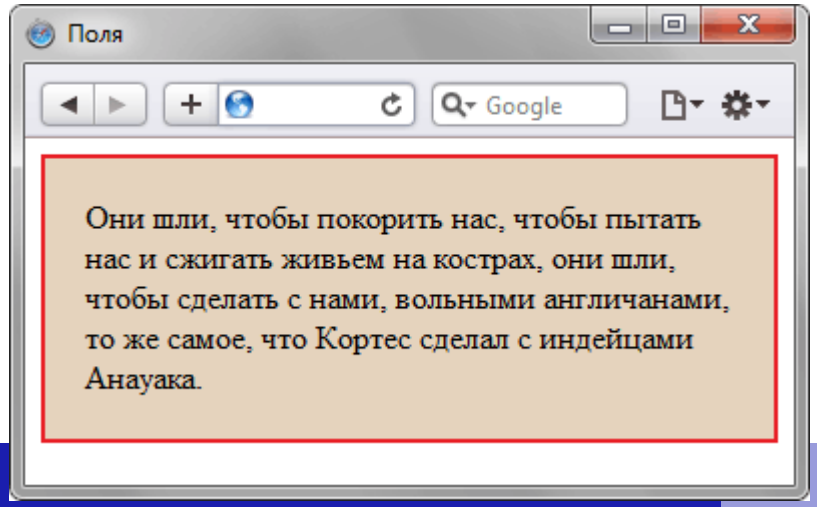
- Размеры областей определяют свойства padding и его подсвойства — padding-left, padding-тори т.д., border и его подсвойства, margin и его подсвойства.

Поля

- Расстояние от внутреннего края границы или края блока до воображаемого прямоугольника, ограничивающего содержимое блока.
- Т.к. значения полей могут различаться на каждой стороне, применяют выражения «верхнее поле» или «поле сверху», и т.д.
- Обозначение «поля» следует понимать как одинаковое значение полей для всех сторон.
- Основное предназначение полей — создать пустое пространство вокруг содержимого блочного элемента, например текста, чтобы он не прилегал плотно к краю элемента.
- Использование полей повышает читабельность текста и улучшает внешний вид страницы.

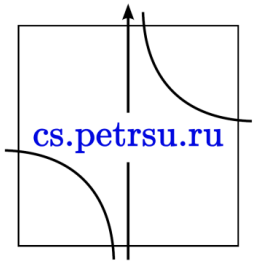


```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Поля</title>
    <style type="text/css">
    .space {
      padding: 20px; /* Поля */
      background: #E5D3BD; /* Цвет фона */
      border: 2px solid #E81E25; /* Параметры рамки */
    }
    </style>
  </head>
  <body>
    <div class="space"> Они шли, чтобы покорить нас, чтобы пытать нас
    и сжигать живьем на кострах, они шли, чтобы сделать с нами,
    вольными англичанами, то же самое, что Кортес сделал с индейцами
    Анауака.</div>
  </body>
</html>
```

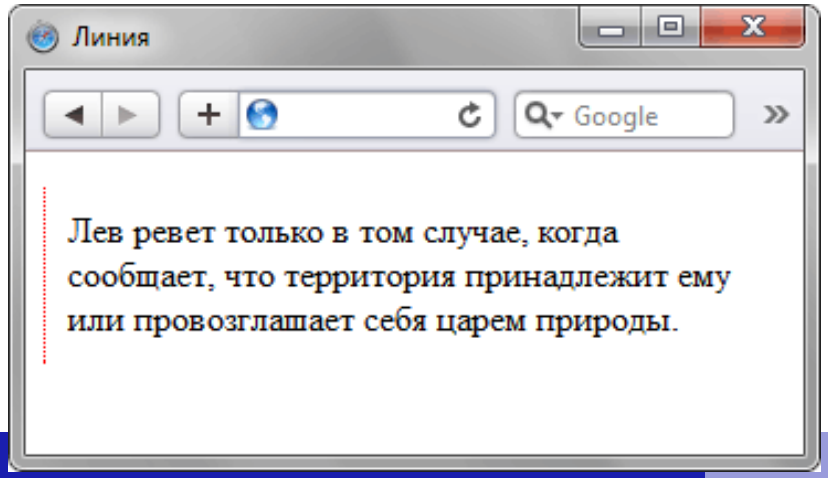


Границы

- Это линии вокруг полей элемента на одной, двух, трёх или всех четырёх его сторонах.
- У каждой линии есть толщина, стиль и цвет.
- 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.

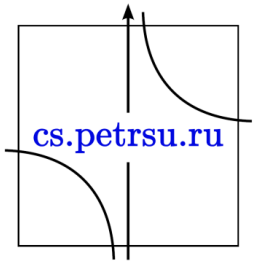


```
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
      <title>Линия</title>
      <style type="text/css">
      p.line {
         border-left: 1px dotted red;
         padding: 10px;
      }
      </style>
   </head>
   <body>
      <p class="line">Лев ревет только в том случае, когда сообщает, что
      территория принадлежит ему или провозглашает себя царем
      природы.</p>
   </body>
</html>
```



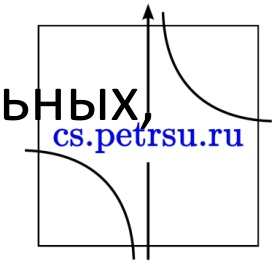
Отступы

- Пустое пространство от внешнего края границы, полей или содержимого блока.
- Применяют выражения «верхний отступ» или «отступ сверху», и т.д.
- Обозначение «отступы» следует понимать как одинаковое значение отступов для всех сторон.

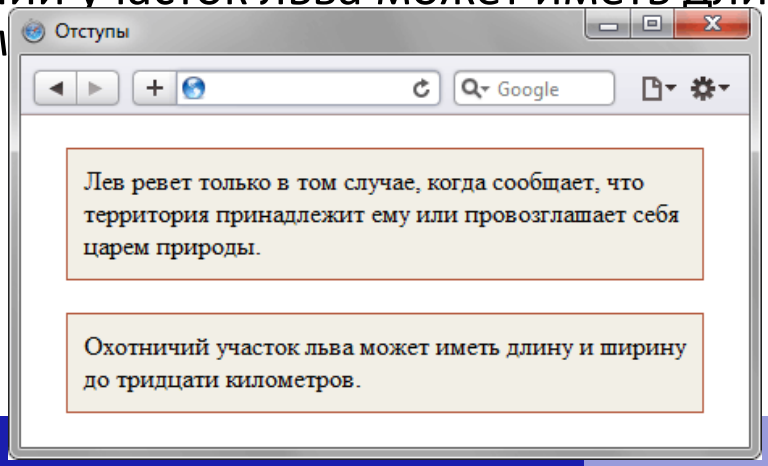


Особенности отступов

- 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.
- 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.
- Для отступов характерно явление под названием «схлопывание», когда отступы у близлежащих элементов не суммируются, а объединяются меж собой.
- Отступы, заданные в процентах, вычисляются от ширины контента блока. Это касается как вертикальных, так и горизонтальных отступов.



```
<!DOCTYPE html>
<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 киломе
  </body>
</html>
```



Ширина блока

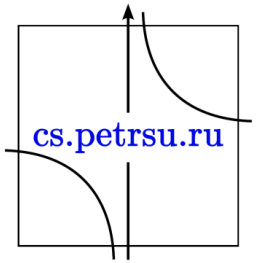
- Комплексная величина и складывается из:

  width — ширина контента, т.е. содержимого блока;

  padding-left и **padding-right** — поле слева и справа от контента;

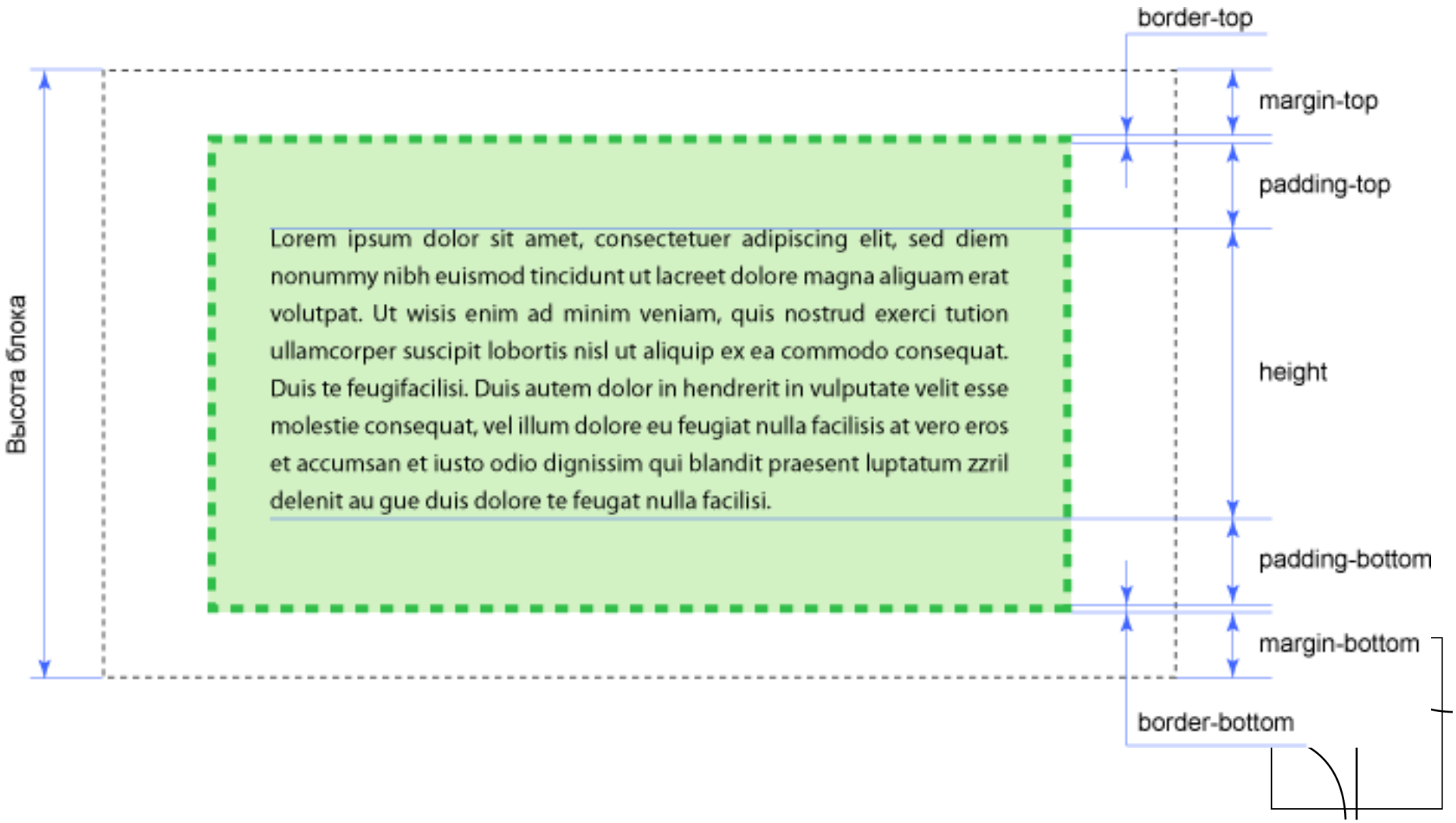
  border-left и **border-right** — толщина границы слева и справа;

  margin-left и **margin-right** — отступ слева и справа.



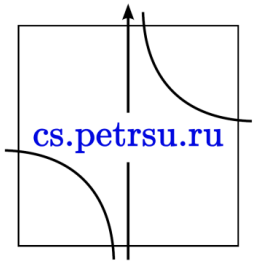
Ширина блока

Высота блока



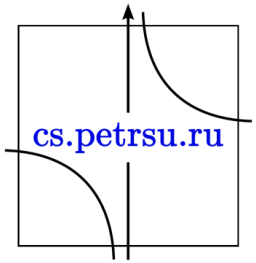
- Так как поля и отступы элемента не являются обязательными, по умолчанию их значение равно нулю.
- Тем не менее, некоторые браузеры добавляют этим свойствам положительные значения по умолчанию на основе своих таблиц стилей.
- Очистить стили браузеров для всех элементов можно при помощи универсального селектора:

```
* {
    margin: 0;
    padding: 0;
 }
```



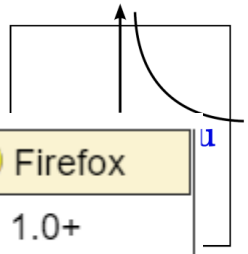
Минусы

- 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.
  - Предположим, что ширина контента задана как 90%, если сюда приплюсовать поля и границы, заданные в пикселах, то нельзя вычислить суммарную ширину блока, поскольку проценты напрямую в пикселы не переводятся (в CSS3 есть calc).
  - 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 слоями.



Свойство box-sizing

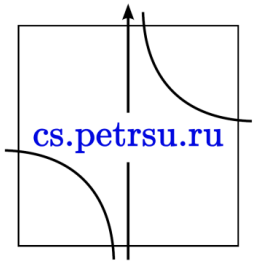
- При значении border-box ширина начинает включать поля и границы, но не отступы.
- Подключая box-sizing со значением border-box к своему стилю, мы можем задавать ширину в процентах и спокойно указывать border и padding, не боясь, что изменится ширина блока.
- Не все браузеры его понимают.



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

Вложенные слои

- Идея простая — для внешнего блочного элемента задаётся только необходимая ширина, а для вложенного блока всё остальное — поля, границы и отступы.
- По умолчанию ширина блока равна доступной ширине родителя, получится, что блоки в накладываются друг на друга, при этом фактическая ширина такого комбинированного элемента будет чётко задана.
- 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.
- Ещё один недостаток добавление дополнительного блока, который усложняет структуру кода, особенно при частом применении метода. Но это можно считать мелочью по сравнению с преимуществами.



Свойство: calc(выражение)

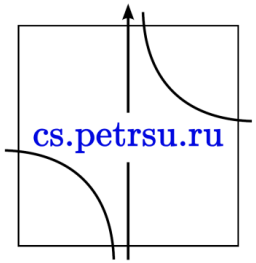
- Выражение представляет собой комбинацию значений размеров и следующих знаков математических действий.

  + — сложение (width: calc(20px + 20px));

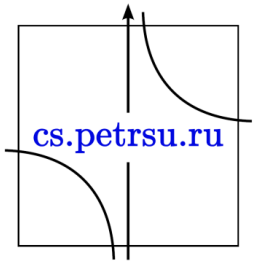
  - — вычитание (padding: calc(10% - 10px););

  * — умножение (height: calc(20%*2);)

  / — деление. На ноль делить запрещено (width: calc(100%/3);).



```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8">
      <title>calc</title>
      <style>
      div {
         border: 1px solid #000;
         height: 200px;
         background: url(images/figure.jpg) no-repeat;
         background-position: calc(100% - 20px) 0;
      }
      </style>
   </head>
   <body>
      <div></div>
   </body>
</html>
```

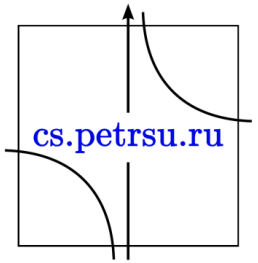


Свойство **overflow**

Управляет отображением содержания блочного элемента, если оно целиком не помещается и выходит за область заданных размеров

Значения:

- visible - отображается все содержание элемента, даже за пределами установленной высоты и ширины, значение по умолчанию.
- hidden - отображается только область внутри элемента, остальное будет скрыто.
- scroll - всегда добавляются полосы прокрутки.
- auto - полосы прокрутки добавляются только при необходимости.
- inherit - наследует значение родителя.



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>overflow</title>
    <style>
    .layer {
      overflow: scroll; /* Добавляем полосы прокрутки */
      width: 300px; /* Ширина блока */
      height: 150px; /* Высота блока */
      padding: 5px; /* Поля вокруг текста */
      border: solid 1px black; /* Параметры рамки */
    }
    </style>
  </head>
  <body>
    <div class="layer">
      <h2>Duis te feugifacilisi</h2>
      <p>Lorem ipsum dolor sit amet,
      nonummy nibh euismod tincidur
      erat volutpat. Ut wisis enim ad m
    </div>
  </body>
</html>
```