Интернет-технологии и распределённая обработка данных

ЛЕКЦИЯ 7

Работа с блоками и текстом в CSS

Блоки, их граница и фон

Градиенты

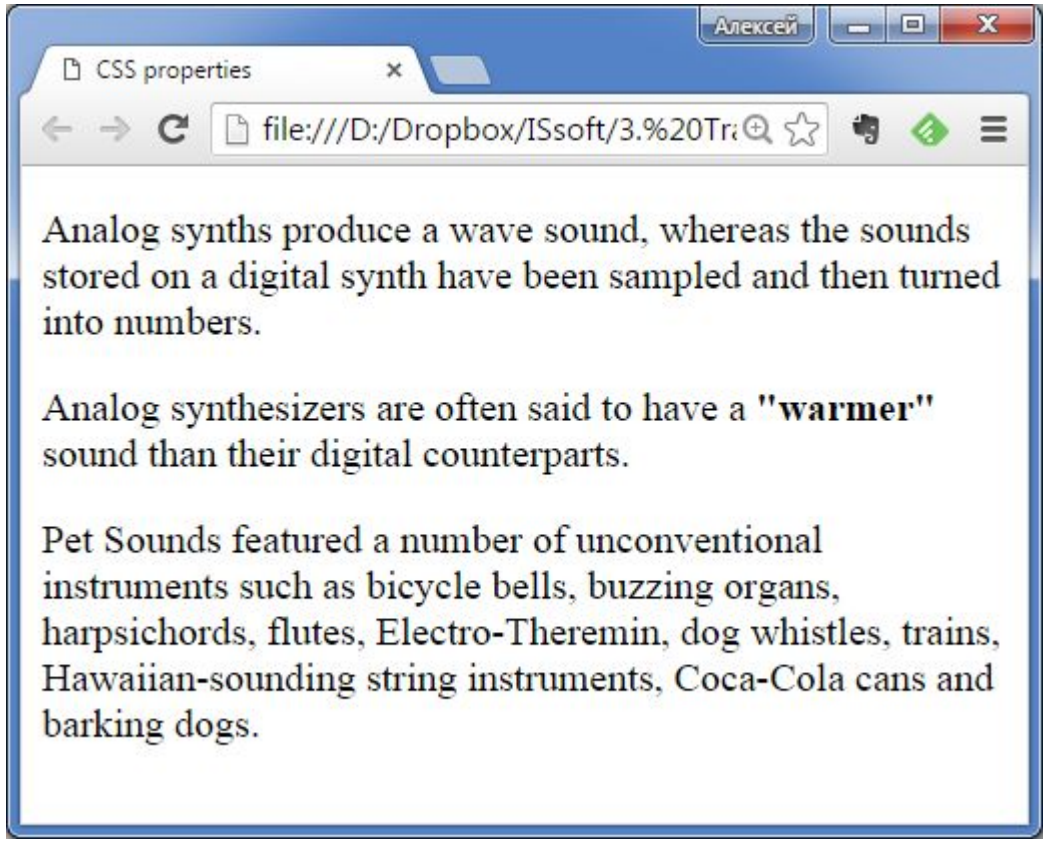
Размер блока, отступы и поля

Позиционирование блоков

Документ для примеров

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="mystyle.css">
</head>
<body>
  <p class="one">
    Analog synths produce a wave sound, whereas the sounds stored on a digital synth
    have been sampled and then turned into numbers.</p>
  <p class="two">
    Analog synthesizers are often said to have a <b>"warmer"</b> sound
    than their digital counterparts.</p>
  <p class="three">
    Pet Sounds featured a number of unconventional instruments such as bicycle bells,
    buzzing organs, harpsichords, flutes, Electro-Theremin, dog whistles, trains,
    Hawaiian-sounding string instruments, Coca-Cola cans and barking dogs.</p>
</body>
</html>
```

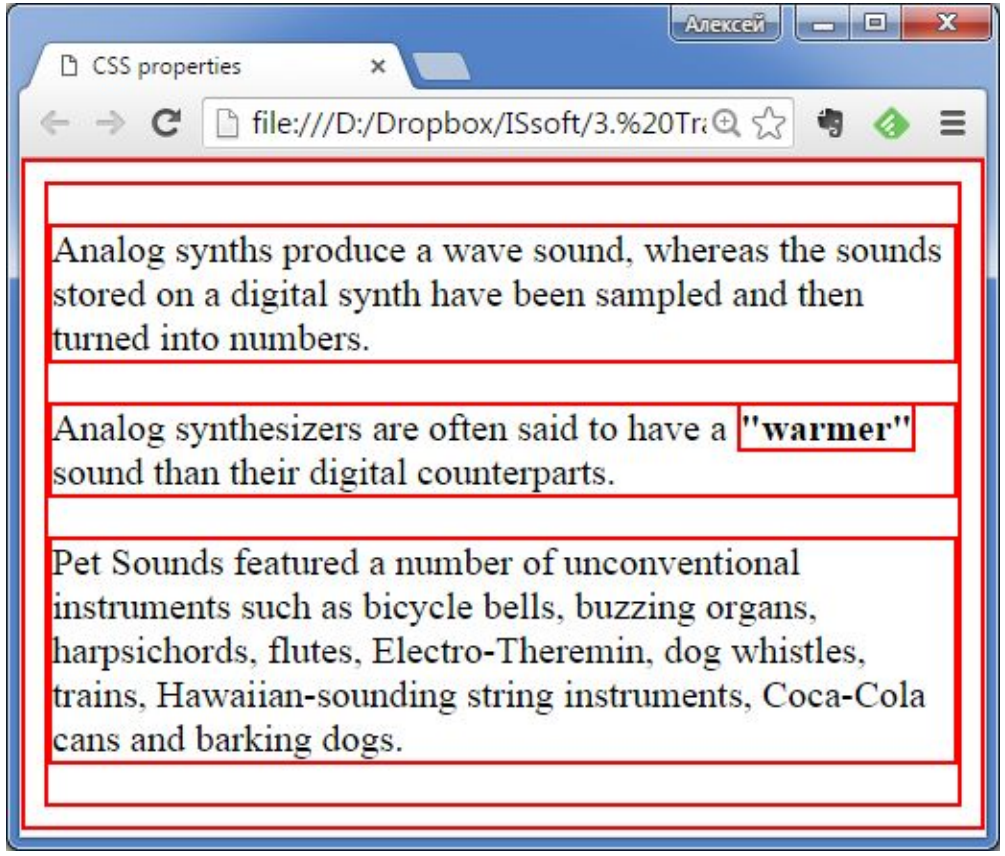
Документ для примеров



CSS-блоки

Если у нас есть веб-страница, нужно представить, что вокруг **каждого видимого** HTML-элемента (и ещё `html` и `body`) существует *прямоугольный блок*, который мы можем настроить при помощи CSS.

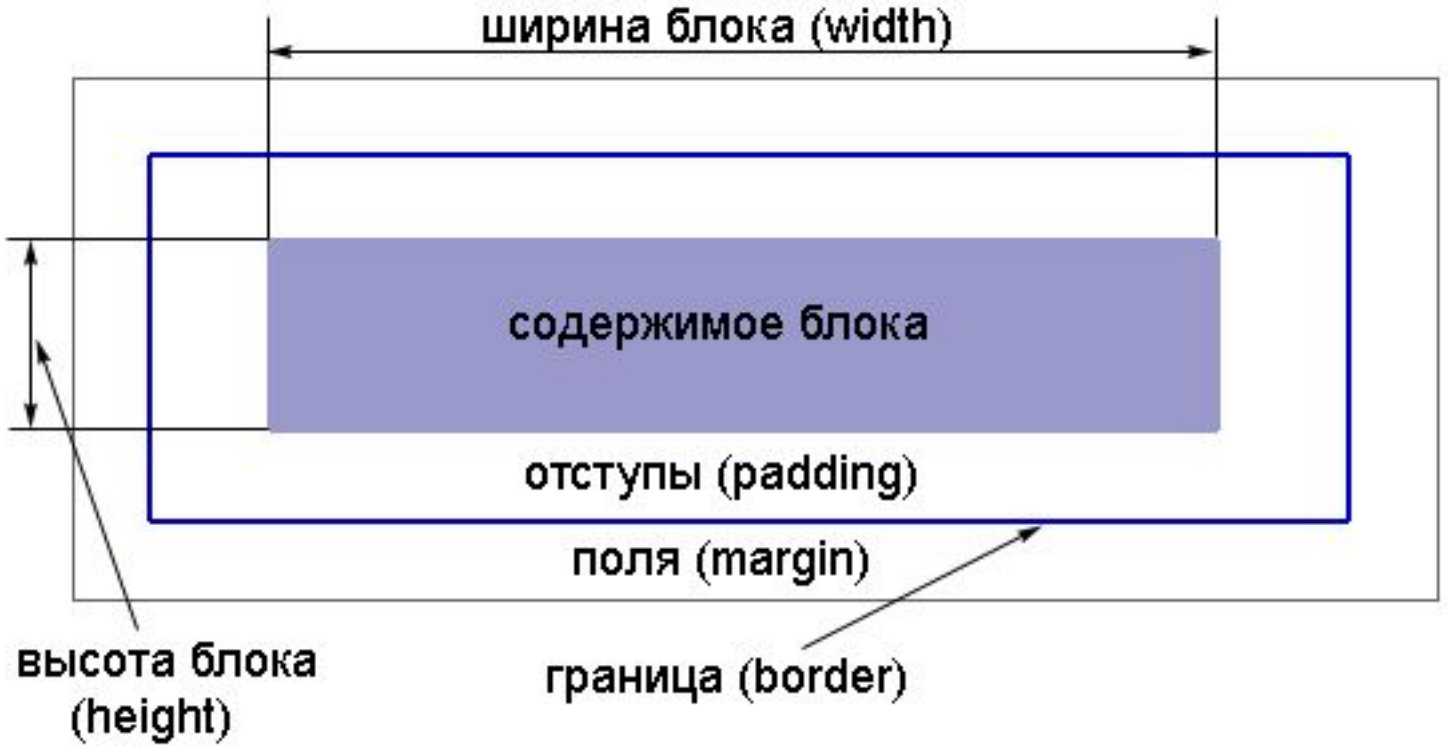
CSS-блоки


Алексей
CSS properties
file:///D:/Dropbox/ISsoft/3.%20Tr
Analog synths produce a wave sound, whereas the sounds stored on a digital synth have been sampled and then turned into numbers.
Analog synthesizers are often said to have a "warmer" sound than their digital counterparts.
Pet Sounds featured a number of unconventional instruments such as bicycle bells, buzzing organs, harpsichords, flutes, Electro-Theremin, dog whistles, trains, Hawaiian-sounding string instruments, Coca-Cola cans and barking dogs.

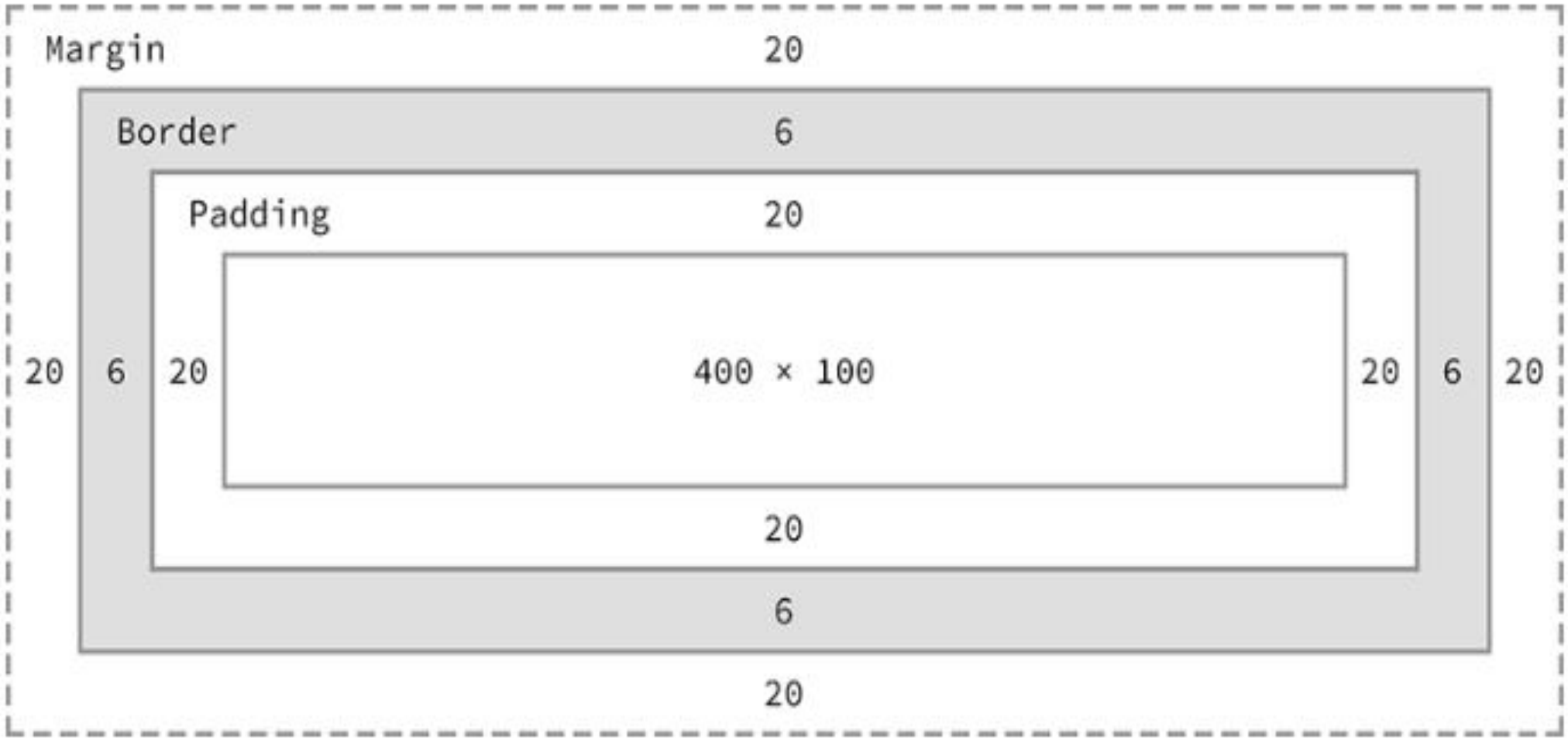
Структура блока

- Блок имеет прямоугольную форму.
- Вокруг *содержимого* блока есть *отступы*. Фон отступов такой же, как и у содержимого.
- Содержимое и отступы обрамлены *границей.* Она может быть видимой или невидимой. Граница имеет толщину.
- Вокруг границы размещены прозрачные *поля*.
- Для содержимого блока определены *ширина* и *высота*. Поля и отступы при задании размера блока не учитываются.

Структура блока



```
div {
    border: 6px solid #949599;
     height: 100px;
     margin: 20px;
     padding: 20px;
     width: 400px;
     }
```



Ширина: 492px = 20px + 6px + 20px + 400px + 20px + 6px + 20px

Высота: 192px = 20px + 6px + 20px + 100px + 20px + 6px + 20px

```css
blockquote { padding: 20px; }

blockquote { padding-bottom: 20px;}

blockquote { padding: 20px 0 10px 50px; }

blockquote { padding: 20px 0; }
/*          сверху
             20px

слева                       справа
  0                           0

            снизу
            20px
*/
```

- margin
- padding
- border-width

margin (отступ) добавляет пространство *снаружи*, между элементом и другими элементами.

```
p { margin: 40px; } //отступы вокруг абзаца
```

Структура блока

При работе с полями следует учитывать такое явление, как *«схлопывание полей»* (margin collapsing).

У двух соседних по вертикали блоков значения полей не складываются – **используется большее**!

Из этого правила есть исключения: плавающие блоки, абсолютно-позиционированные блоки, inline-блоки.

Оба отступа положительны

выбирается наибольшее значение из двух отступов, и оно устанавливается как расстояние между блоками. При одинаковых значениях отступов за расстояние между блоками принимается одно из них.

Один из отступов отрицательный

складывание отступов по правилам математики:

x + (-y) = x – y

Здесь x и y величина прилегающих отступов элементов.

Если полученное значение в результате суммирования окажется отрицательным, то оно будет действовать на нижний блок, соответственно, он сдвинется вверх на указанное значение.

Оба отступа отрицательны

Из двух значений выбирается наибольшее по модулю, оно же и выступает в качестве отрицательного отступа между элементами. Так, если отступы равны -10px и -20px, то итоговое значение будет -20px.

Объединение вертикальных margin

заголовок и подзаголовок.

CSS

```
.title { margin-bottom: 30px; }
.subtitle { margin-top: 15px; }
```

HTML

```
<h1 class="title">MarkSheet</h1>
<h2 class="subtitle">Простое руководство по
HTML и CSS</h2>
```

Схлопывание не срабатывает для элементов, у которых :

- установлено свойство **padding**.
- на стороне схлопывания задана граница;
- **position** установлено как **absolute**;
- свойство **float** задано как **left** или **right**;
- для строчных элементов;
- для **<html>**.
- значение **overflow** задано как **auto**, **hidden** или **scroll** - схлопывание не действует для их дочерних элементов;
- свойство **clear** - не схлопывается верхний отступ с нижним отступом родительского элемента.

Структура блока

Замечание: как правило, CSS-свойства, связанные с настройкой блоков, являются **ненаследуемыми**.

Напоминание: многие ненаследуемые свойства могут принимать специальное значение `inherit`, которое *заставляет* брать значение свойства у родителя.

Настройка границы блока

У границы можно настроить цвет (`color`), толщину линии (`width`) и тип линии (`style`).

Имена CSS-свойств конструируем так: дописываем через дефис указанные атрибуты либо к свойству `border` (все границы), либо к свойствам `border-top`, `border-right`, `border-bottom`, `border-left`. Например:

`border-width` толщина всех границ

`border-top-color` цвет верхней границы

Настройка границы блока

Цвет задаётся как обычно в CSS.

Для толщины используем слова `thin` (2px), `medium` (4px), `thick` (6px), или любую единицу размера, **исключая %**.

Для `style` используем: `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`.

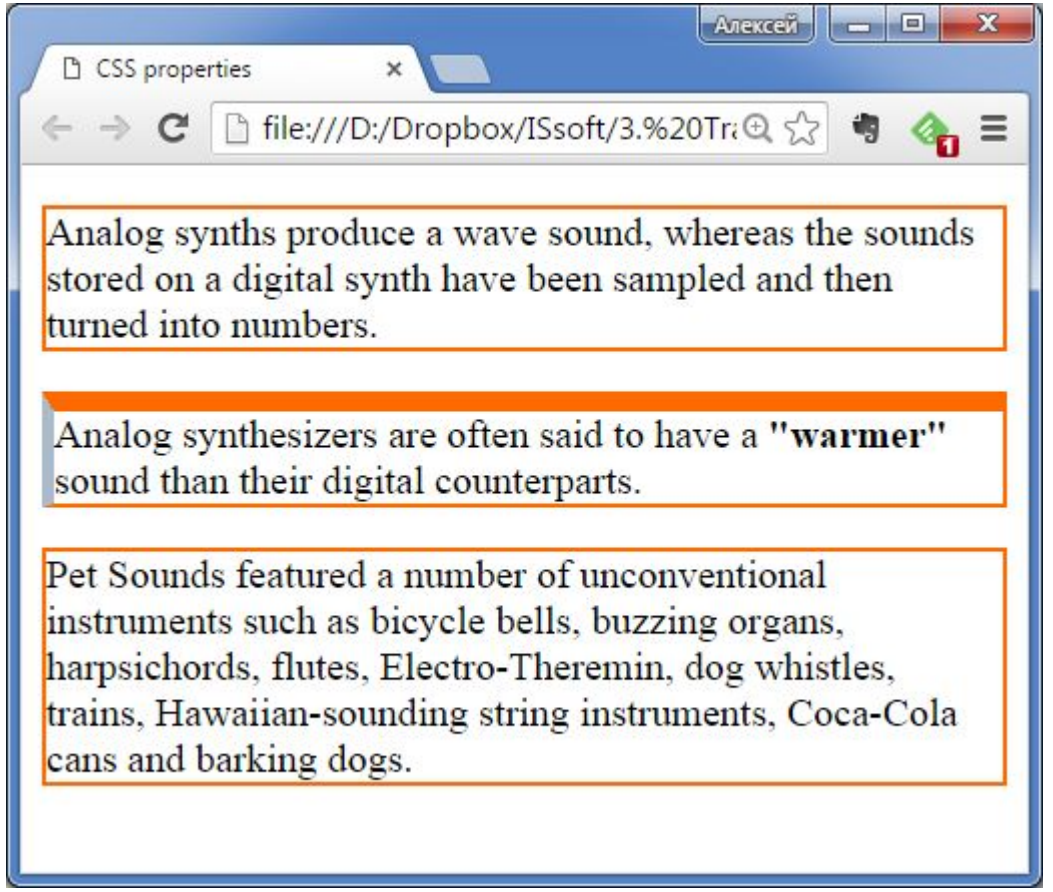
`none` Не отображает границу и (`border-width`) задается нулевой.

`hidden` тот же эффект, что `none` кроме `border-style` у ячеек таблицы

`inherit` Наследует значение родителя.

Настройка границы блока – пример

```
p {
  border-color: rgb(255,106,0);
  border-width: 2px;
  border-style: solid;
}

p.two {
  border-top-width: 0.5em;
  border-left-color: #ABC;
  border-left-style: solid;
  border-left-width: 4pt;
}
```



1 пиксел	3 пиксела	5 пикселов	7 пикселов
dotted	dotted	dotted	dotted
dashed	dashed	dashed	dashed
solid	solid	solid	solid
double	double	double	double
groove	groove	groove	groove
ridge	ridge	ridge	ridge
inset	inset	inset	inset
outset	outset	outset	outset

Настройка границы блока

У границы или отдельной грани можно задать сразу три свойства: *толщина тип цвет*:

```
border: 2px solid red;
```

Настраивая свойства `border-*`, можно задать одно значение (все грани), два значения (верх-низ и левый-правый), три значения (верх, левый-правый, низ), четыре значения (верх, правый, низ, левый):

```
border-width: 10px 20px;
border-width: 1px 3px 5px 7px;
```

самый быстрый способ получить три границы

```
blockquote {
border: 1px solid yellow;
border-left: none;
}
```

Нельзя смешивать два свойства:

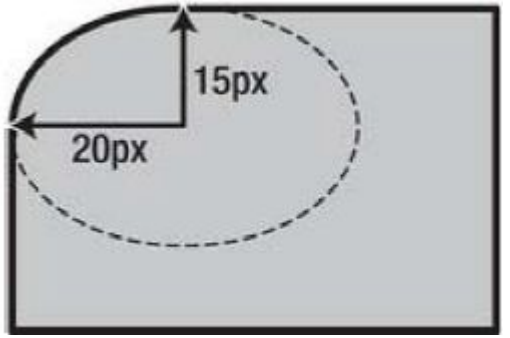
```
blockquote { border: 1px 0 solid green; }
/* Не работает */
```

Можно опустить толщину в border и установить её отдельно:

```
blockquote { border: solid yellow; border-width: 1px 0; }
```

Закругление углов

CSS3 позволяет настроить закругления для углов границы блока.



Параметром закругления является либо одна единица размера (радиус закругления), либо две единицы (два радиуса эллипса закругления – X и Y).

Закругление углов

Свойства для настройки закругления углов:

- `border-top-left-radius`
- `border-top-right-radius`
- `border-bottom-left-radius`
- `border-bottom-right-radius`

Пример:

```
p { border-top-left-radius: 20px 15px; }
```

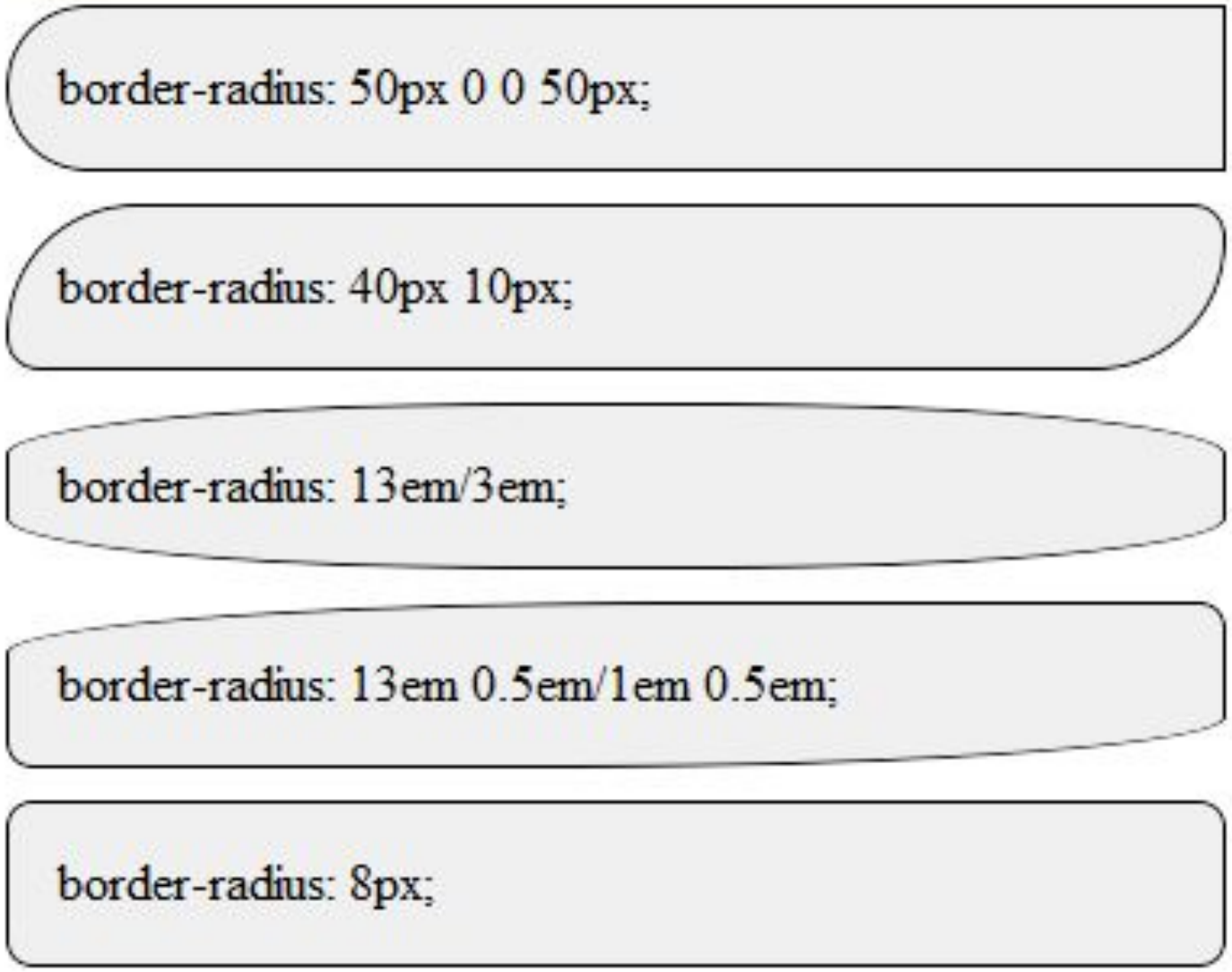
Закругление углов

`border-radius` позволяет задать несколько закруглений сразу (через пробел). Если нужны вторые радиусы для эллипсов, то их перечисляем через слэш (радиус по горизонтали/ радиус по вертикали)

Число значений	Результат
1	Радиус указывается для всех четырех углов.
2	Первое значение задаёт радиус верхнего левого и нижнего правого угла, второе значение – верхнего правого и нижнего левого угла.
3	Первое значение устанавливает радиус для верхнего левого угла, второе – для верхнего правого и нижнего левого, а третье – для нижнего правого угла.
4	Устанавливает радиус для верхнего левого, верхнего правого, нижнего правого и нижнего левого угла.

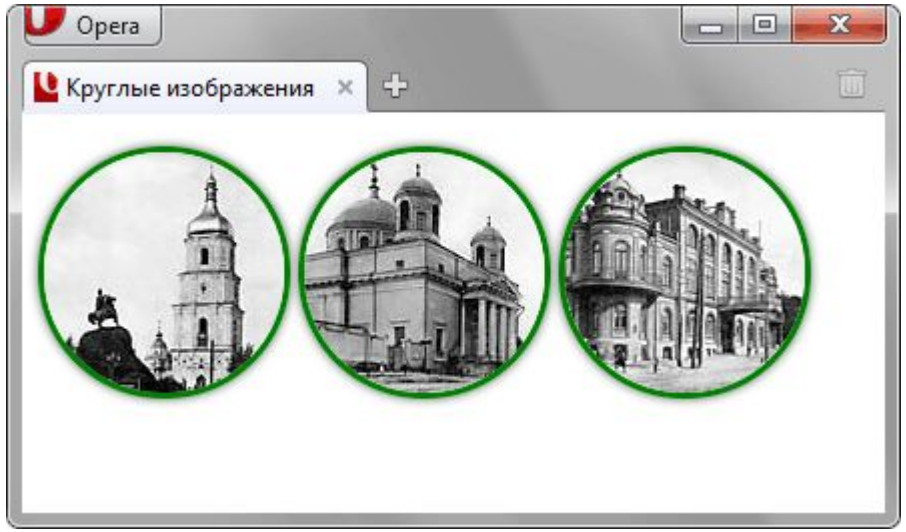
```
p {background: #f0f0f0;
   border: 1px solid black;
   padding: 15px;
   margin-bottom: 10px;}
```

```
p.one {
border-radius: 50px 0 0 50px;}
p.two {
border-radius: 40px 10px;}

p.three {
border-radius: 13em/3em;}
p. four{
border-radius:
        13em 0.5em/1em 0.5em;}
p. five{
border-radius:
      13em 0.5em/1em 0.5em;}
```



Круглые изображения

```
.round {
    border-radius: 100px; /* Радиус скругления */
    border: 3px solid green; /* Параметры рамки */
    box-shadow: 0 0 7px #666; /* Параметры тени */
  }
```



```
.round {
    border-radius: 100px; /* Радиус скругления */
    box-shadow: 0 0 0 3px green, 0 0 13px #333;
/* Параметры теней */
    }
```

Изображения для границы

CSS3 позволяет строить границу, используя указанное изображения (**проверяйте поддержку! IE11**).

Реализуется при помощи свойств:

- `border-image-source`
- `border-image-slice`
- `border-image-width`
- `border-image-outset`
- `border-image-repeat`
- `border-image` (сокращённая форма задания)

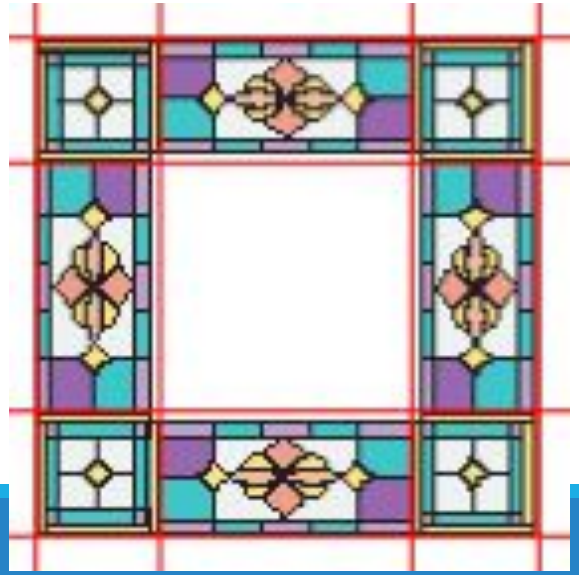
border-image

none | [<URL> [<число> | <проценты>]{1,4} [/ <толщина>{1,4}]?] && [stretch | repeat | round]{0,2}

border-image

none- Не отображает рисованную рамку, используется установленный стиль границы

URL - Путь к графическому файлу. Обязательный параметр.



число – значения, разделенные пробелами, которые указывают размеры частей изображения в пикселах, задавая тем самым области деления. Сами единицы не пишутся, только число (10, а не 10px).

Число значений	Результат
1	Устанавливает границы одинаковой толщины на каждой стороне рисунка.
2	Первое значение устанавливает высоту верхней и нижней границы, второе — левой и правой.
3	Первое значение определяет высоту верхней границы, второе — левой и правой, а третье — высоту нижней границы.
4	Поочередно устанавливается размеры верхней, правой, нижней и левой границы

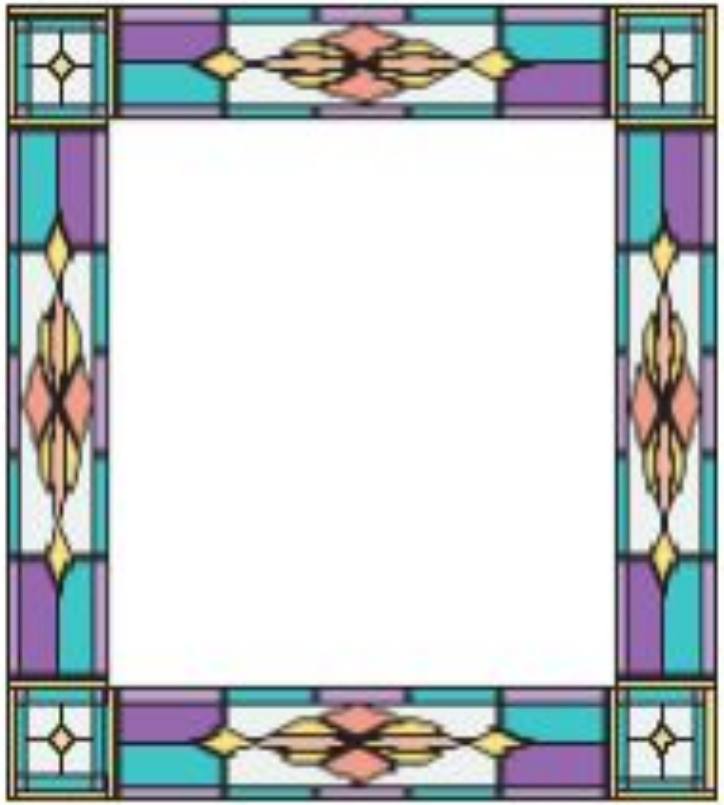
проценты – как <число>, но значения задаются в процентах. Тот или другой параметр обязателен.

толщина – через слэш пишется одно, два, три или четыре значения толщины границы на каждой стороне элемента. Является аналогом border-width и использует тот же синтаксис.

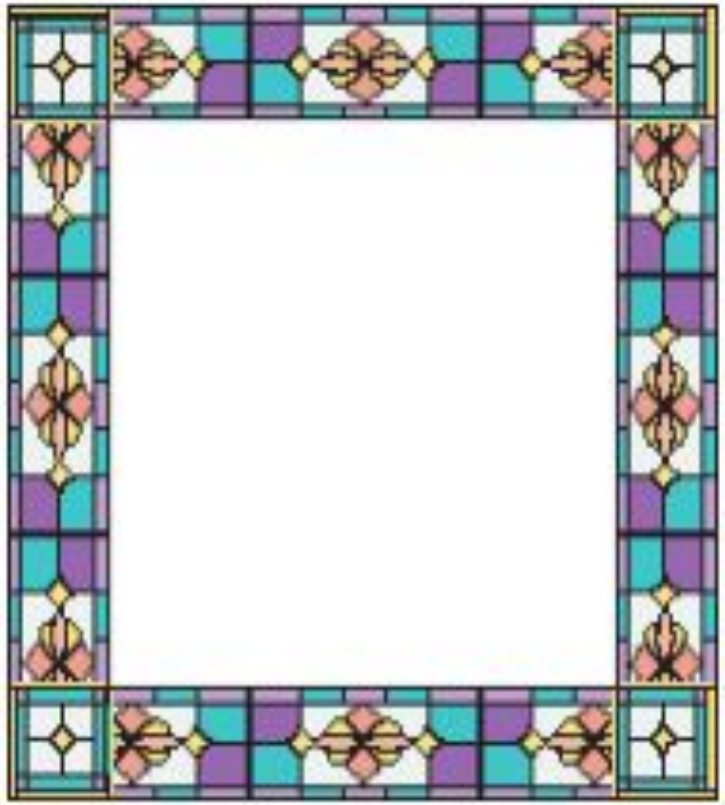
stretch – Растягивает рисунок границы до размеров элемента. (по умолчанию)

repeat – Повторяет рисунок границы.

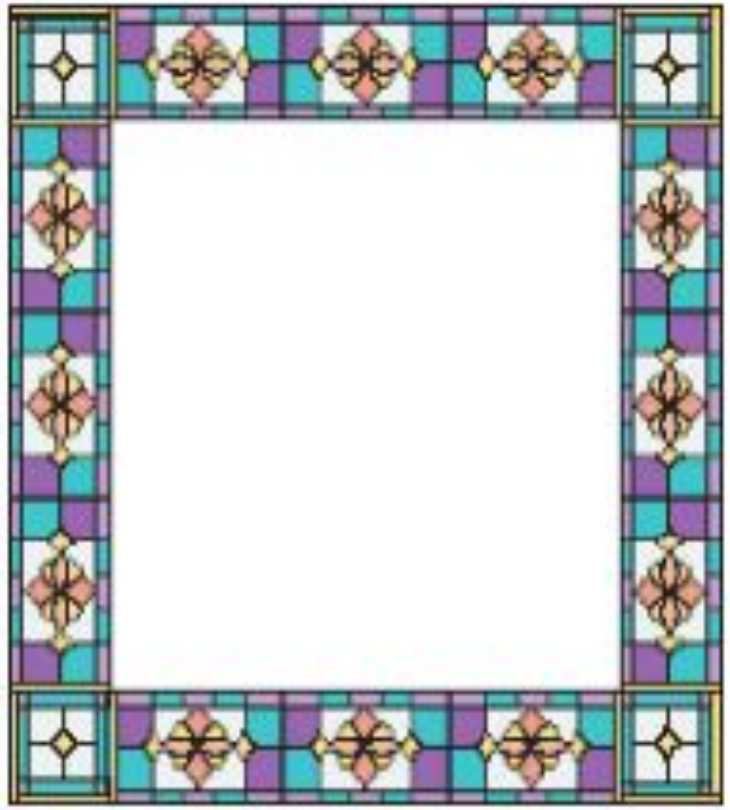
round – Повторяет рисунок и масштабирует его так, чтобы на стороне элемента оказалось целое число изображений.



stretch



repeat



round

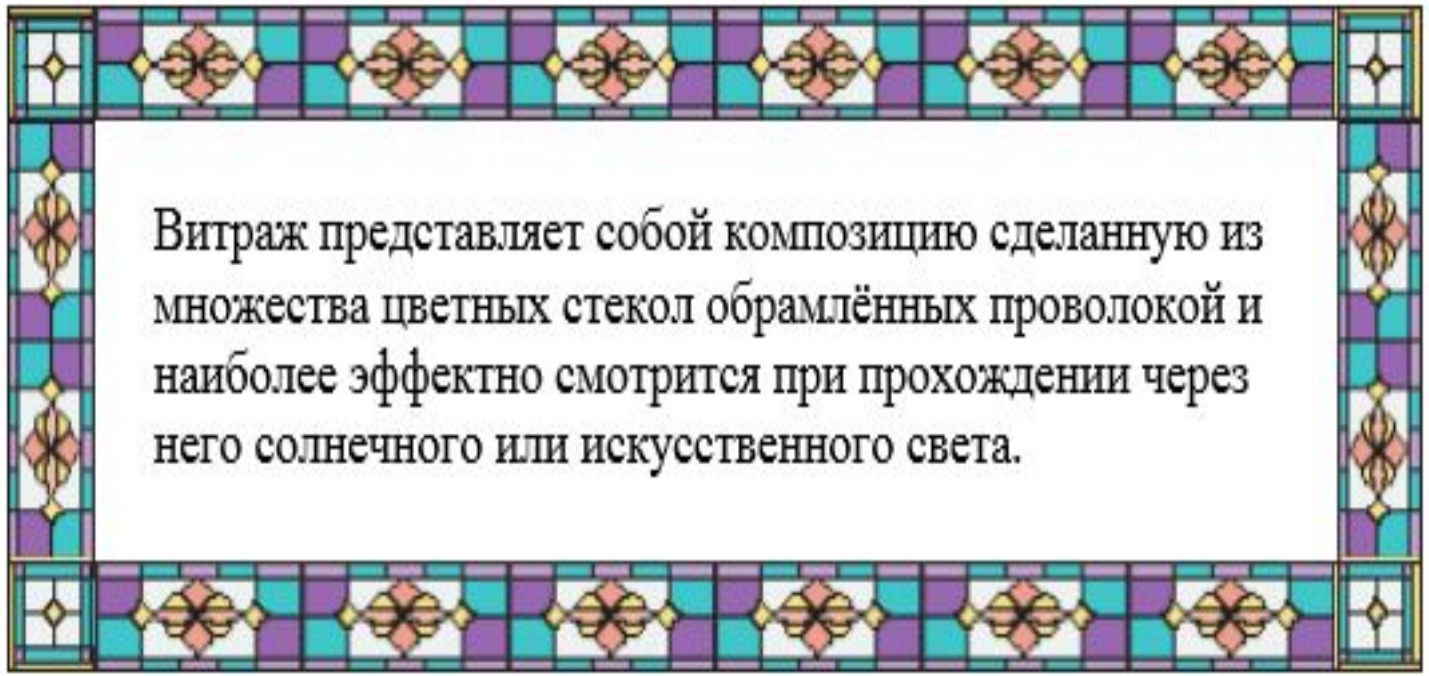
```
div {
    border: 30px solid #40c4c8;
    padding: 20px;
    -moz-border-image: url(images/bg-image.png) 30 round round;
    -webkit-border-image: url(images/bg-image.png) 30 round round;
-o-border-image: url(images/bg-image.png) 30 round round;
border-image: url(images/bg-image.png) 30 round round;
}
```

-moz-border-image: Firefox до версии 15.0

-webkit-border-image: Safari, Chrome до версии 15.0, Android и iOS

-o-border-image: Opera до версии 15.0

```
div {
border: 30px solid #40c4c8;
padding: 20px;
border-image:
url(/example/image/bg-image.png) 30 round round;   }
```



Свойство outline

Универсальное свойство, задаёт толщину, стиль и цвет **на всех четырёх** сторонах блока.

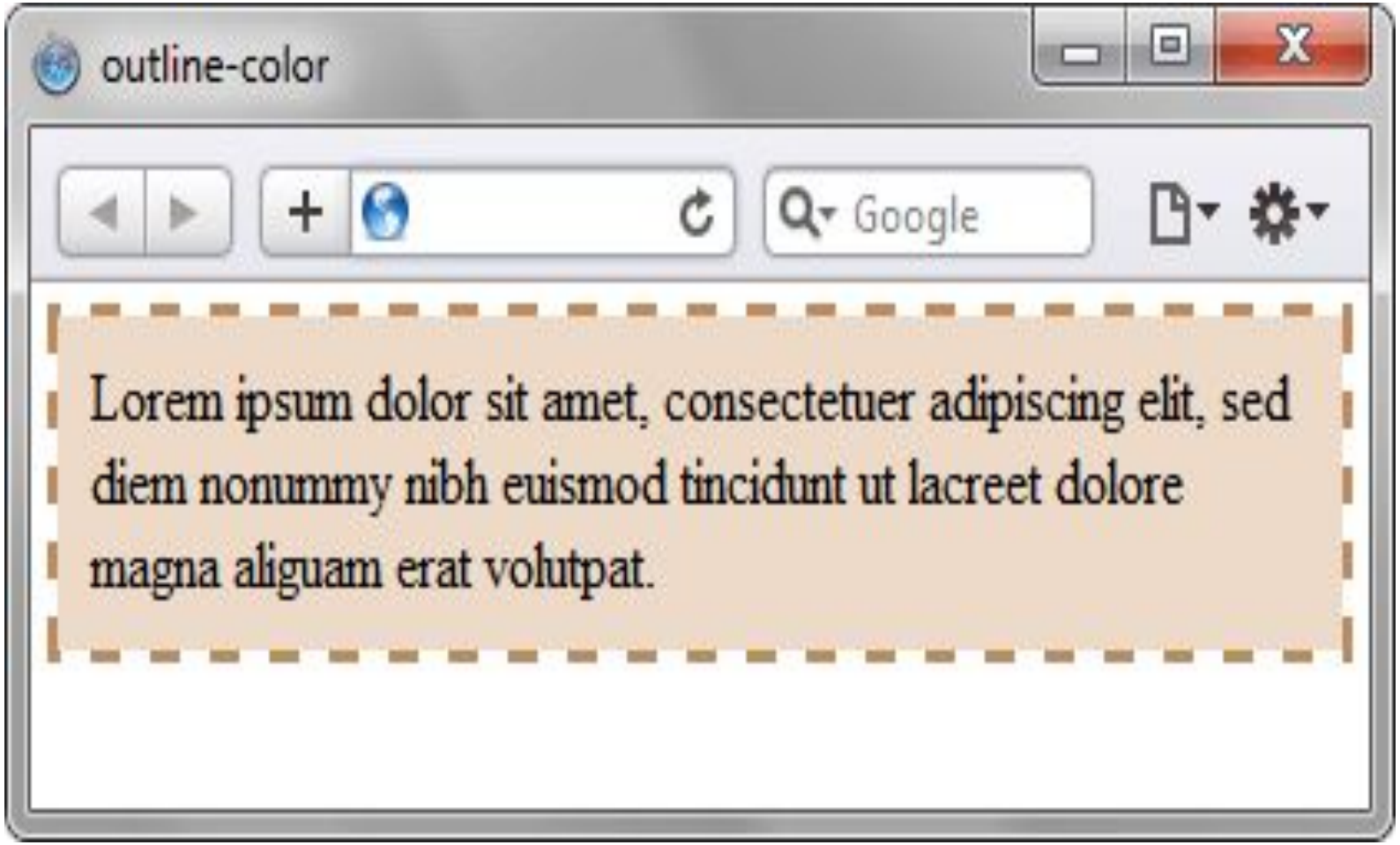
В отличие от линии, задаваемой через `border`, свойство `outline` **не влияет на вычисленную ширину блока**.

outline: outline-color || outline-style || outline-width | inherit

Свойство outline

*) Есть отдельные свойства outline-width, outline-style, outline-color и ещё свойство outline-offset.

```
#block {
    outline-style: dashed; /* Пунктирная граница */
    outline-color: #be8b5e; /* Цвет границы */
    padding: 10px; /* Поля вокруг текста */
    background: #eedac8; /* Цвет фона */
}
```


outline-color
Google
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diem nonummy nibh euismod tincidunt ut lacreet dolore magna aliguam erat volutpat.

`outline-offset:` — Устанавливает расстояние между рамкой, созданной с помощью свойства `outline`, и краем или границей элемента добавленной через `border`.

Синтаксис

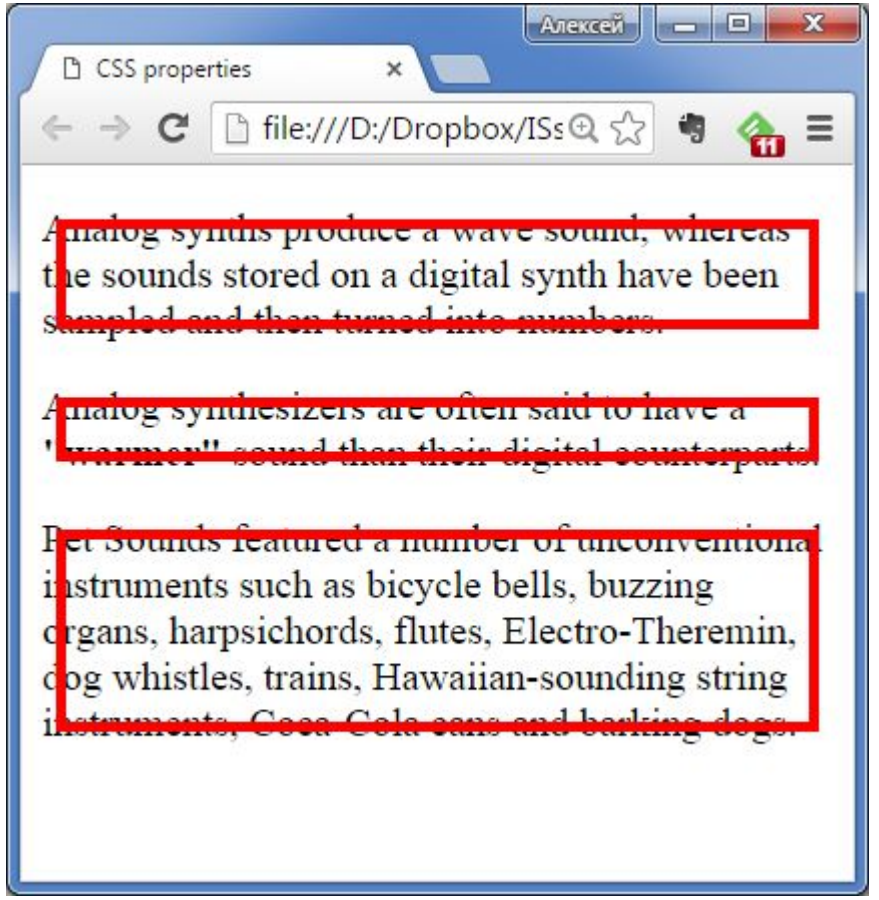
```
outline-offset: <размер> | inherit
```

`<размер>` Задаёт расстояние от края элемента до рамки. Отрицательное значение отображает рамку внутри элемента, положительное — вокруг элемента.

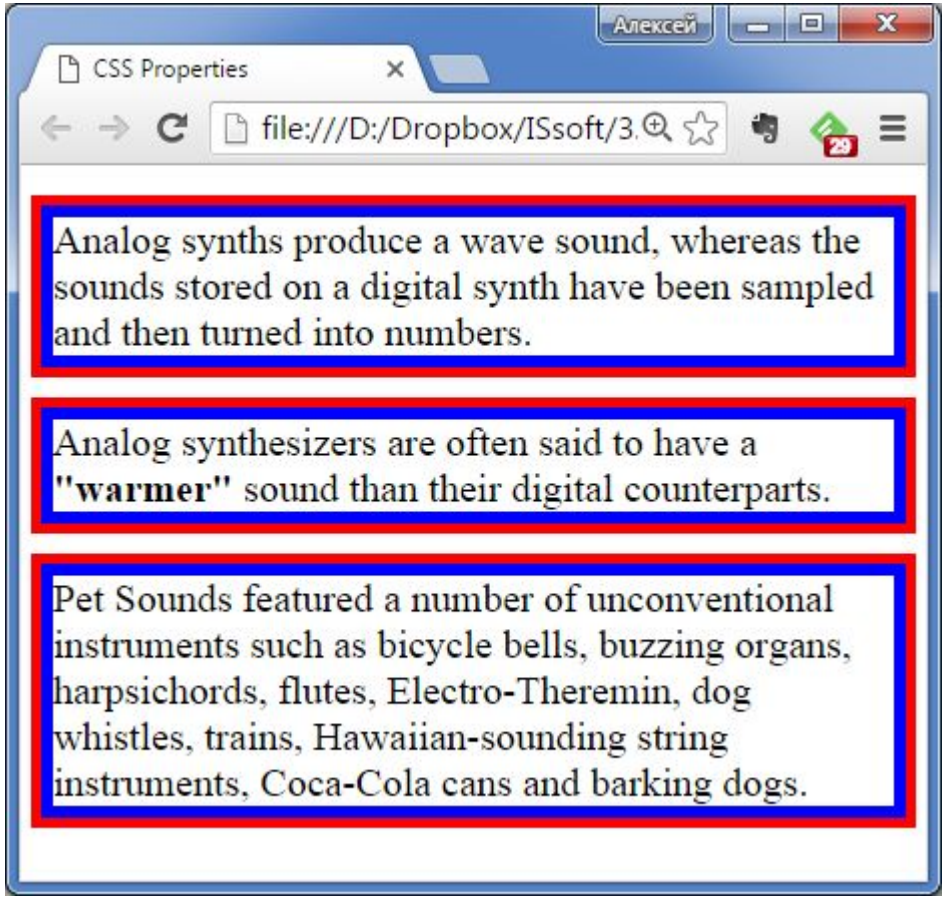
```
.clue {
    background: url(images/leather.jpg); /* Фоновый
рисунок */
    outline: 2px dashed rgba(255,255,255,0.8); /*
Пунктирная рамка */
    outline-offset: -10px; /* Выводим рамку внутри
элемента */
    padding: 10px; /* Поля */
    min-height: 100px; /* Минимальная высота */
  }
```


outline-offset

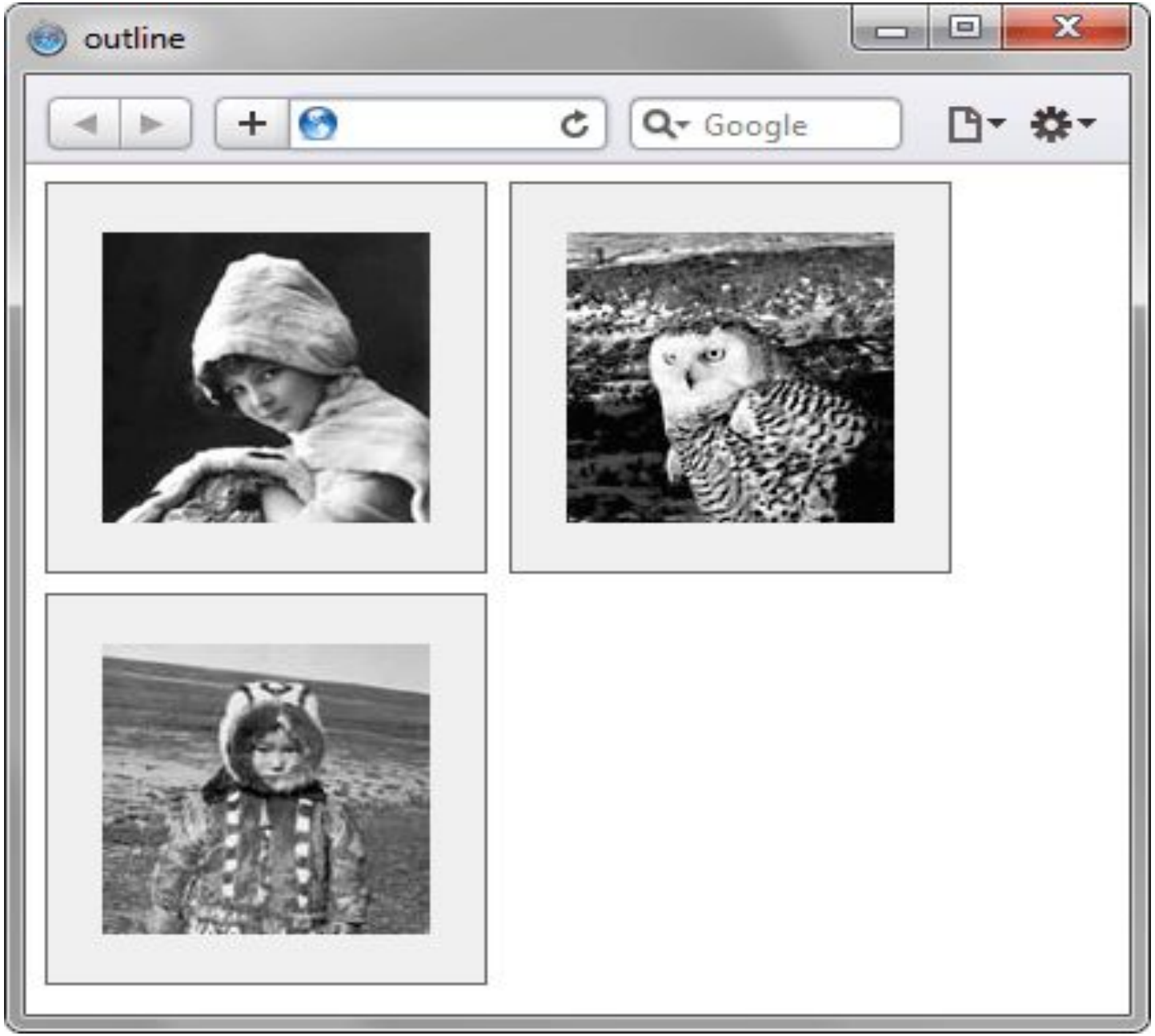
```
p {
  outline: thick solid red;
  outline-offset: -10px;
}
```



```
p {
  border: 5px solid blue;
  outline: thick solid red;
}
```



```
.photo img {
    padding: 20px; /* Поля вокруг изображения */
    margin-right: 10px; /* Отступ справа */
    margin-bottom: 10px; /* Отступ снизу */
    outline: 1px solid #666; /* Параметры рамки */
    background: #f0f0f0; /* Цвет фона */
    float: left; /* Обтекание по правому краю */
  }
```


outline
Google

Фон блока

Цвет фона задаётся свойством `background-color` (по умолчанию ***transparent*** - *прозрачный*), а фоновая картинка – свойством `background-image`.

Для настройки отображения картинки служат свойства `background-repeat`, `background-size`, `background-position`, `background-clip`, `background-origin`.

Свойство `background` – сокращённая форма для задания параметров фона

(`-attachment`||`-image`||`-position`||`-repeat`||`-color`).

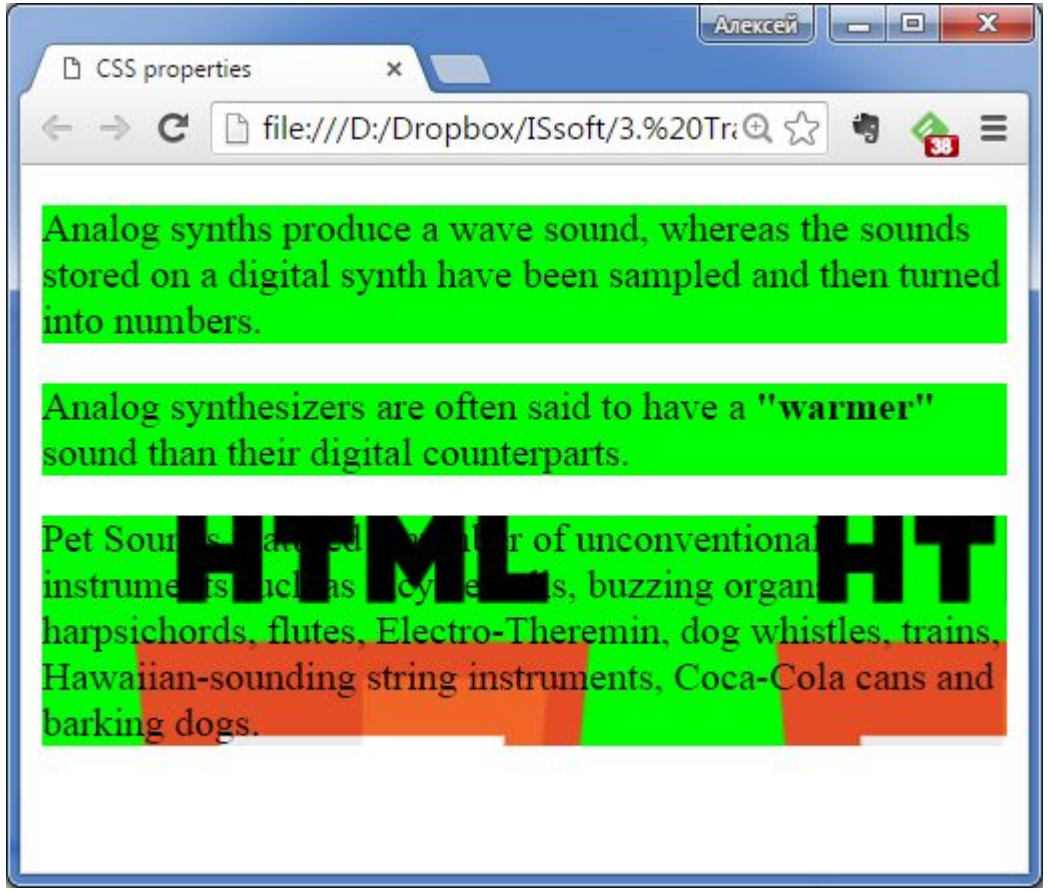
Фон блока

`background-attachment`	Устанавливает, будет ли прокручиваться фоновое изображение вместе с содержимым элемента.
`background-repeat`	Определяет, как будет повторяться фоновое изображение. Можно установить повторение рисунка только по горизонтали, по вертикали или в обе стороны.
`background-size`	Масштабирует фоновое изображение согласно заданным размерам.
`background-position`	Задаёт начальное положение фонового изображения.
`background-clip`	Определяет, как цвет фона или фоновая картинка должна выводиться под границами блока. Эффект заметен при прозрачных или пунктирных границах блока.
`background-origin`	Определяет область позиционирования фонового рисунка (относительно границы, относительно края элемента с учетом толщины границы, …).

Фон блока – пример 1

```
p {
  background-color: lime;
}

p.three {
  background-image: url(HTML5_Logo.png);
}
```

Фон блока – пример 1



background-attachment

`fixed` Делает фоновое изображение элемента неподвижным.

`scroll` Позволяет перемещаться фону вместе с содержимым.

`inherit` Наследует значение родителя.

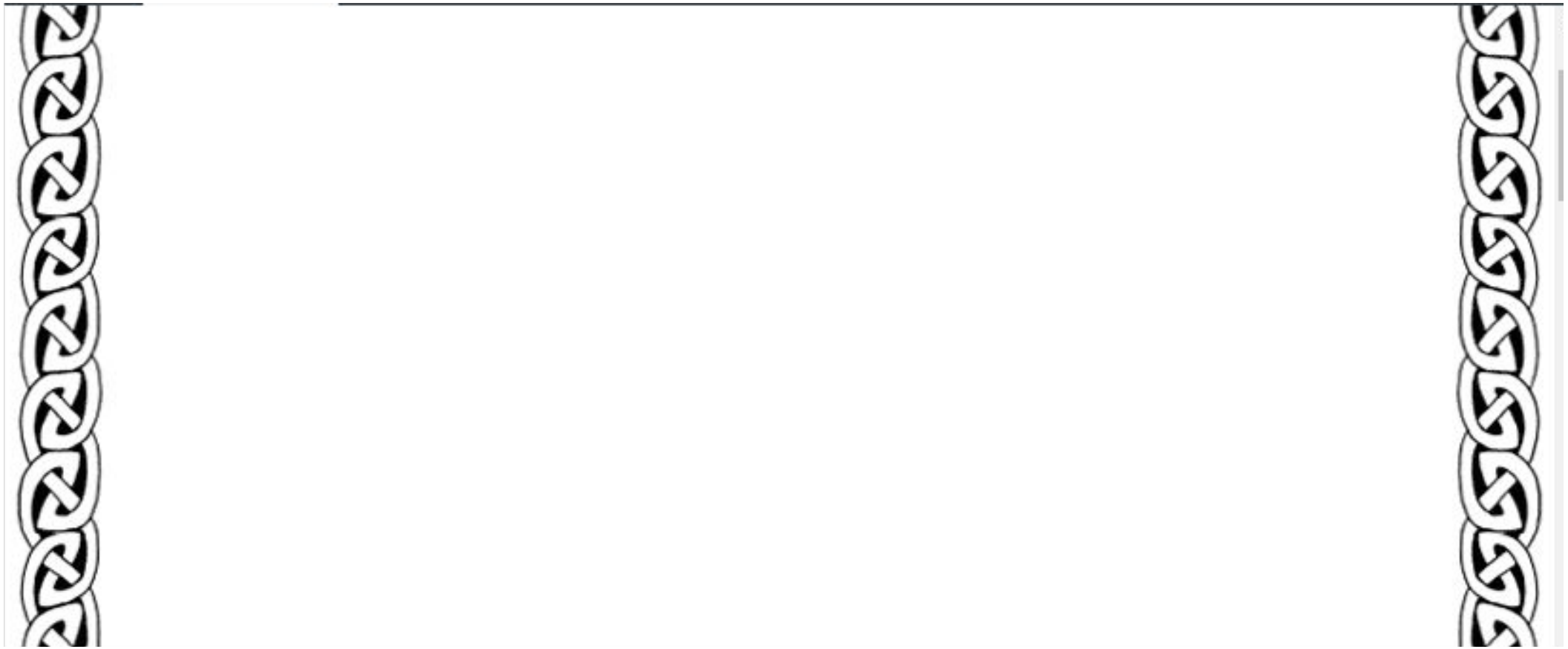
`local` Фон фиксируется с учётом поведения элемента. Если элемент имеет прокрутку, то фон будет прокручиваться вместе с содержимым, но фон выходящий за рамки элемента остаётся на месте.

CSS2.1

```
background-image: url(images/help.png);
/* Путь к фоновому изображению */
    background-attachment: fixed;
/* Фиксируем фон веб-страницы */
```

CSS3

```
background-image: url(images/pattern-left.png),
                              url(images/pattern-right.png);
    background-repeat: repeat-y, repeat-y;
    background-position: left, right;
    background-attachment: fixed, fixed;
```



background-repeat

`no-repeat` Устанавливает одно фоновое изображение в элементе без его повторений, положение которого определяется свойством `background-position` (по умолчанию в левом верхнем углу). Аналогично `no-repeat no-repeat`.

`repeat` Фоновое изображение повторяется по горизонтали и вертикали. `repeat-x` Фоновый рисунок повторяется только по горизонтали. `repeat-y` Фоновый рисунок повторяется только по вертикали.

background-repeat

`inherit` Наследует значение родителя.

`space` Изображение повторяется столько раз, чтобы полностью заполнить область; если это не удаётся, между картинками добавляется пустое пространство.

`round` Изображение повторяется так, чтобы в области поместилось целое число рисунков; если это не удаётся сделать, то фоновые рисунки масштабируются.

CSS 2.1

```
background-image: url(images/bg_grey.png);
/* Путь к фоновому рисунку */
    background-position: left bottom; /* Положение фона
*/
    background-repeat: repeat-x;
/* Повторяем фон по горизонтали */
```

CSS3

```
background-image: url(images/pattern-left.png),
url(images/pattern-right.png);
    background-position: left, right;
    background-repeat: repeat-y, repeat-y;
```

background-size

<значение> Задает размер в любых доступных для CSS единицах

<проценты> Задает размер фоновой картинки в процентах от ширины или высоты элемента.

auto Если задано одновременно для ширины и высоты (auto auto), размеры фона остаются исходными; если только для одной стороны картинки (100px auto), то размер вычисляется автоматически исходя из пропорций картинки.

background-size

`cover` Масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.

`contain` Масштабирует изображение с сохранением пропорций таким образом, чтобы картинка целиком поместилась внутрь блока.

background-position

```
<позиция> = [left | center | right | <проценты> | <размер>]
|| [top | center | bottom | <проценты> | <размер>].
```

У свойства `background-position` два значения, положение по горизонтали (может быть – `left`, `center`, `right`) и вертикали (может быть – `top`, `center`, `bottom`). Также можно задавать в процентах, пикселах или других единицах. Если применяются ключевые слова, то порядок их следования не имеет значения, при процентной записи вначале задается положение рисунка по горизонтали, а затем, через пробел, положение по вертикали.

top left = left top = 0% 0% (в левом верхнем углу)
top = top center = center top = 50% 0% (по центру вверху)
right top = top right = 100% 0% (в правом верхнем углу)
left = left center = center left = 0% 50% (по левому краю и по центру)
center = center center = 50% 50% (по центру)
right = right center = center right = 100% 50% (по правому краю и по центру)
bottom left = left bottom = 0% 100% (в левом нижнем углу)
bottom = bottom center = center bottom = 50% 100% (по центру внизу)
bottom right = right bottom = 100% 100% (в правом нижнем углу)

```
body {
    background-image: url(images/mybg.png);
/* Путь к фоновому рисунку */
    background-position: right bottom;
/* Положение фона */
    background-repeat: no-repeat;
/* Отменяем повторение фона */
   }
```

background-clip

Определяет, как цвет фона или фоновая картинка должна выводиться под границами. Эффект заметен при прозрачных или пунктирных границах.

```
background-clip: [padding-box | border-box | content-box]
```

`padding-box` Фон отображается внутри границ.

`border-box` Фон выводится под границами.

`content-box` Фон отображается только внутри контента.

padding-box

border-box

content-box

```
.example {
    background: #5f392f url(images/gear.png); /* Фоновый
рисунок */
    border: 10px dotted red; /* Параметры рамки */
    background-clip: border-box; /* Фон под рамкой */
    padding: 10px; /* Поля */
    color: #fff; /* Цвет текста */
    min-height: 48px; /* Минимальная высота */

     }
```

background-origin

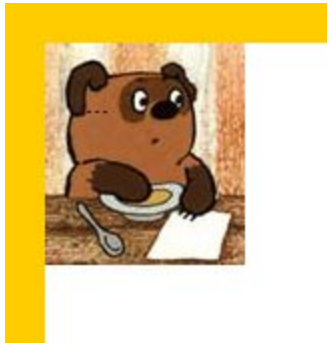
Свойство `background-origin` определяет область позиционирования фонового рисунка. Это свойство не применяется, когда значение `background-attachment` задано как `fixed`.

`padding-box` Фон позиционируется относительно края элемента с учетом толщины границы.

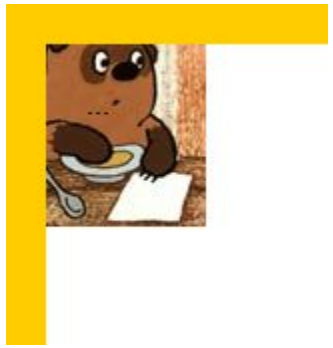
`border-box` Фон позиционируется относительно границы, при этом линия границы может перекрывать изображение.

`content-box` Фон позиционируется относительно содержимого элемента.

```
.example {
    border: 20px solid #fc0;
    padding: 20px;
    height: 200px;
    background: url(images/figure.jpg) no-repeat;
    background-origin: content-box;
  }
```



padding-box



border-box

content-box

Градиенты

Градиент – плавный переход от одного заданного цвета к другому. Бывают *линейные* и *радиальные* градиенты.

Чтобы работать с градиентом нужно задать два цвета и точки перехода.

В CSS градиенты используются для фона (являются значениями для `background-image` или `background`).

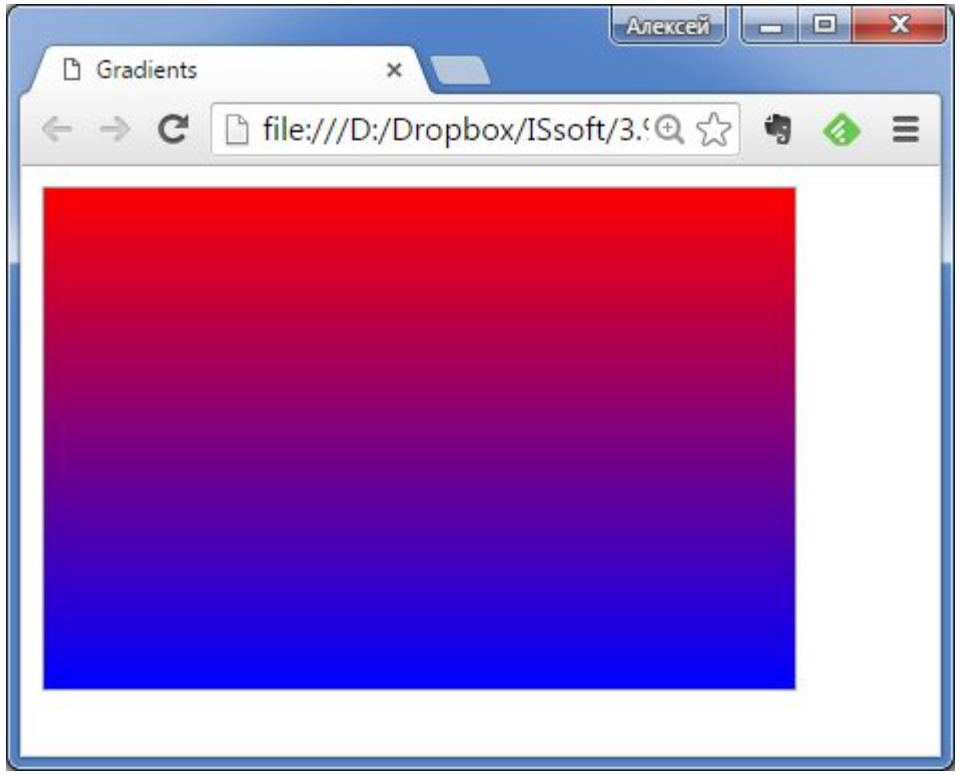
Линейный градиент

Простейший вариант – функция `linear-gradient()` с двумя аргументами-цветами. Это градиент сверху вниз:

```
.grd {
    width: 300px;
    height: 200px;
    border: 1px solid #aaa;
    background-image: linear-gradient(red, blue);
}
```

```
<div class="grd"></div>
```

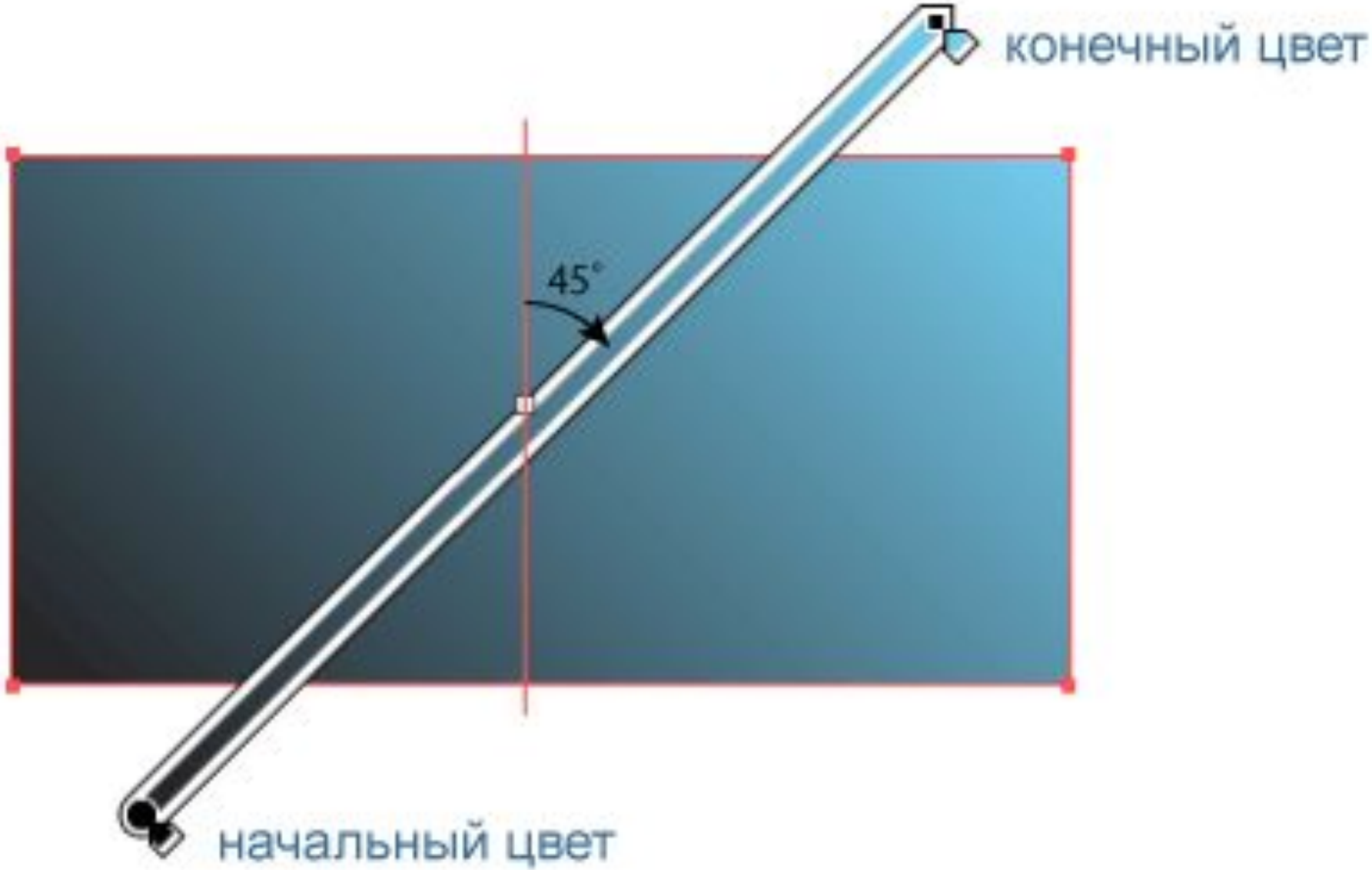
Линейный градиент



Линейный градиент

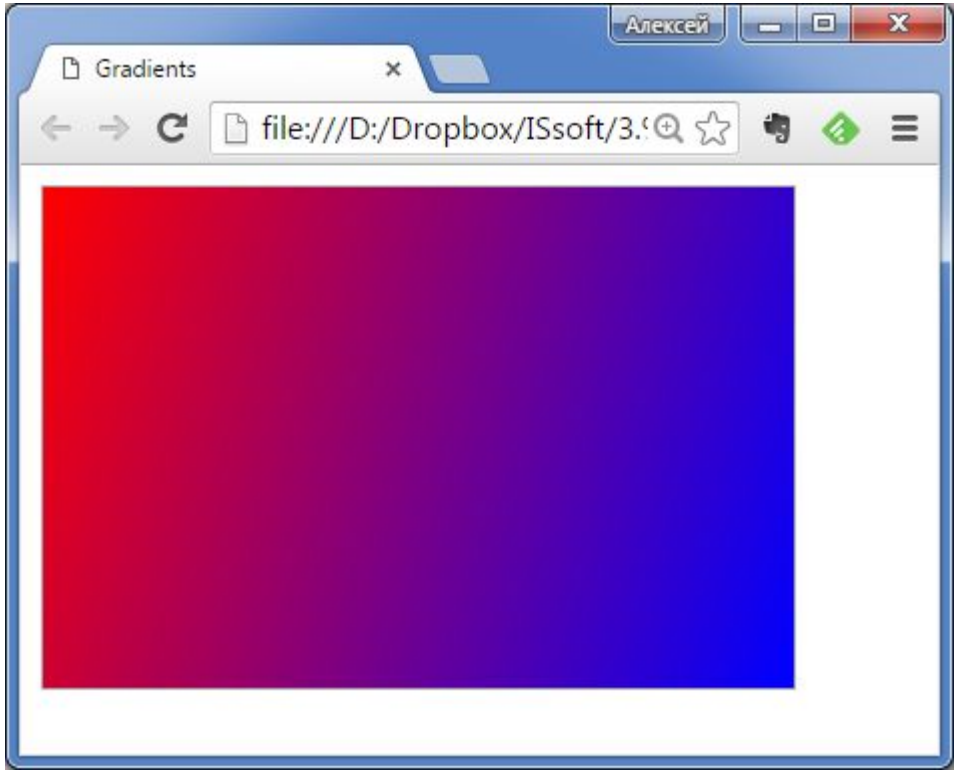
Направление градиента можно задать первым аргументом функции. Записывают

- либо угол (по часовой, 0^o – это **снизу вверх**– 0deg), цвет меняется **к** этому углу.
- либо слово `to` и значения `top`, `bottom` и `left`, `right`, а также их сочетания через пробел.

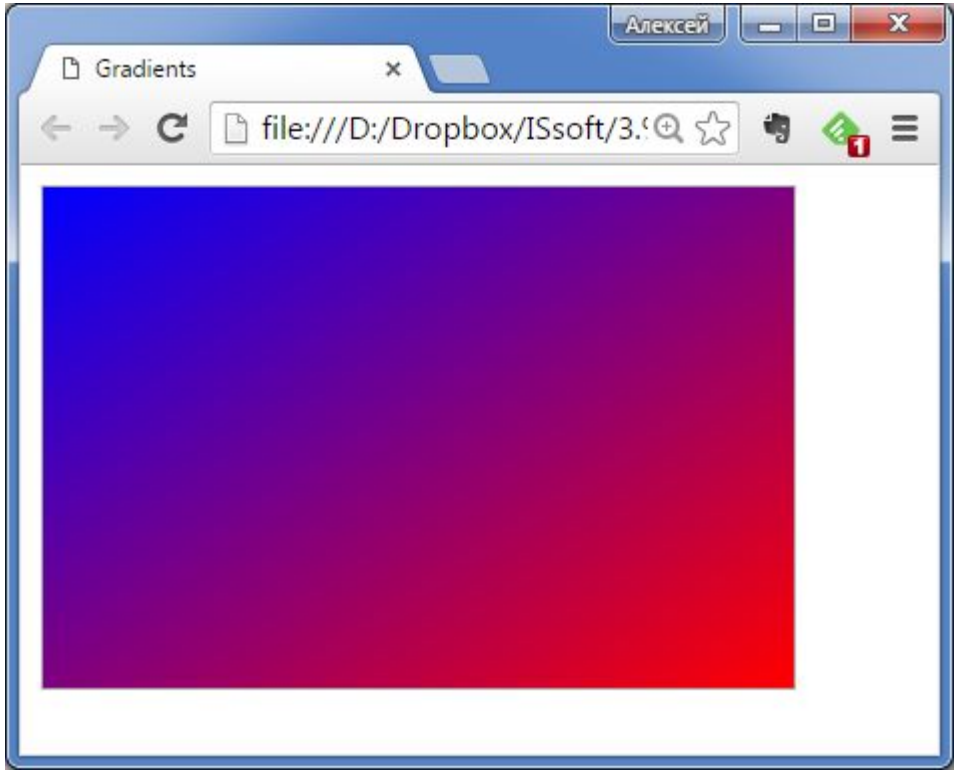

конечный цвет
45°
начальный цвет

Линейный градиент

```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background-image: linear-gradient(110deg, red, blue);
}
```



Линейный градиент



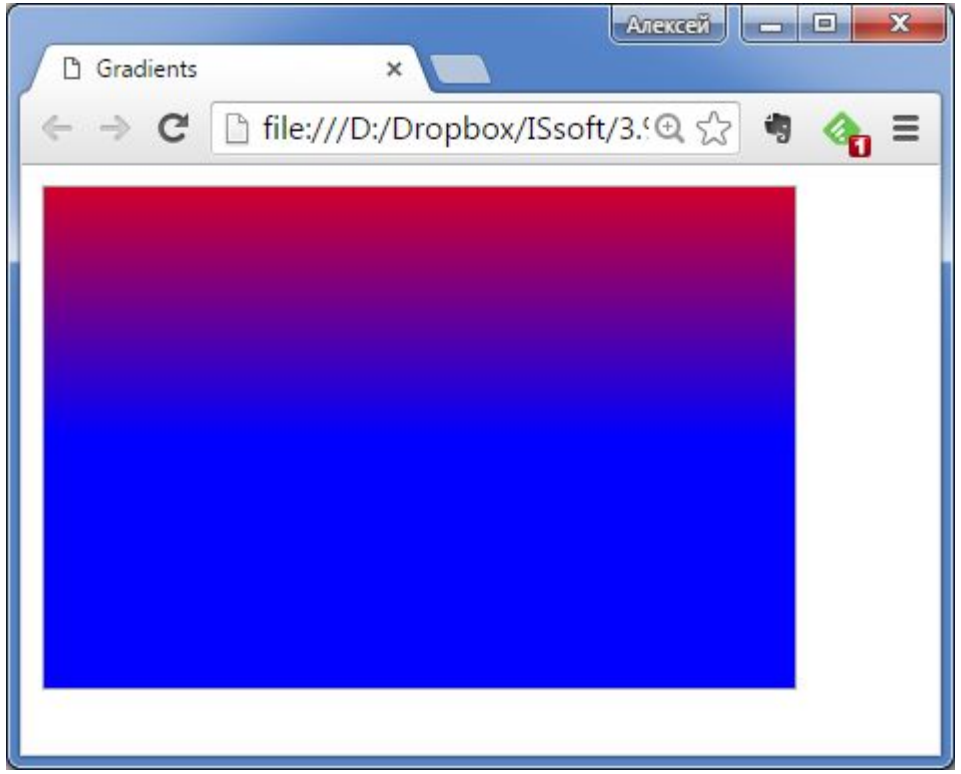

```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background-image: linear-gradient(to left top,red,blue);
}
```

Линейный градиент

После указания цвета, через пробел можно задать **положение точки распространения** цвета на векторе градиента абсолютным размером или процентами.

Линейный градиент

```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background-image: linear-gradient(red -20px, blue 50%);
}
```

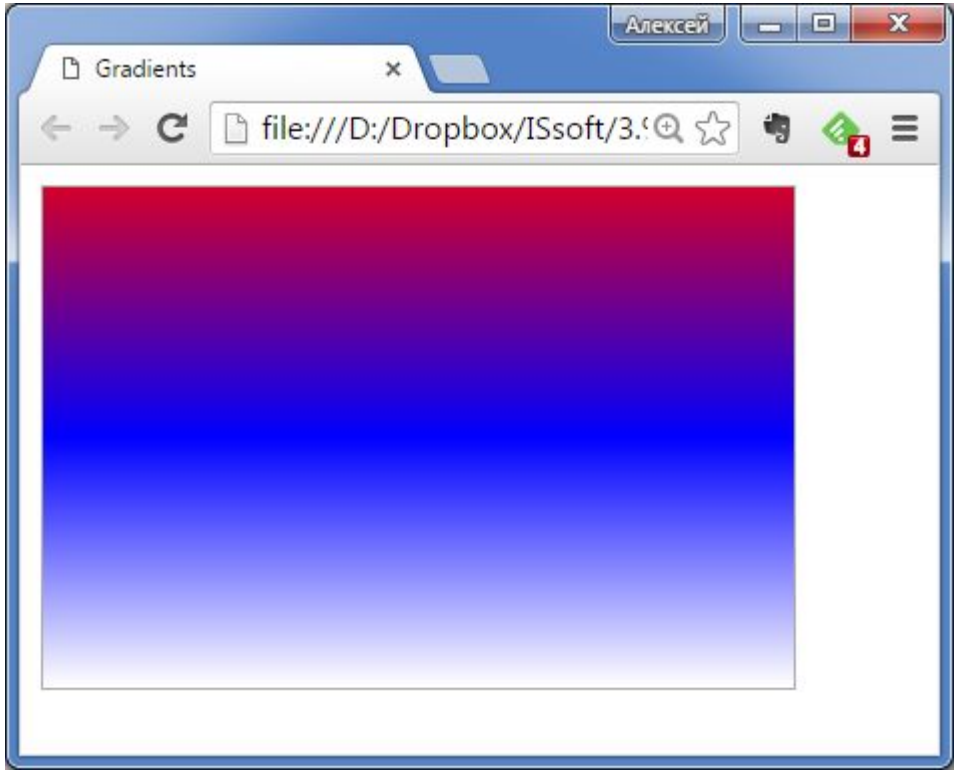


Линейный градиент

Для построения градиента можно использовать не два, а более цветов.

И для каждого цвета можно задавать позицию на векторе градиента.

Линейный градиент




```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background: linear-gradient(red -20px, blue 50%, white);
}
```

```
background: linear-gradient(to top,
                            rgba(30,87,153,0),
                            rgba(30,87,153,0.8),
                            rgba(30,87,153,1),
                            rgba(30,87,153,1),
                            rgba(41,137,216,1),
                            rgba(30,87,153,1),
                            rgba(30,87,153,1),
                            rgba(30,87,153,0.8),
                            rgba(30,87,153,0));
```

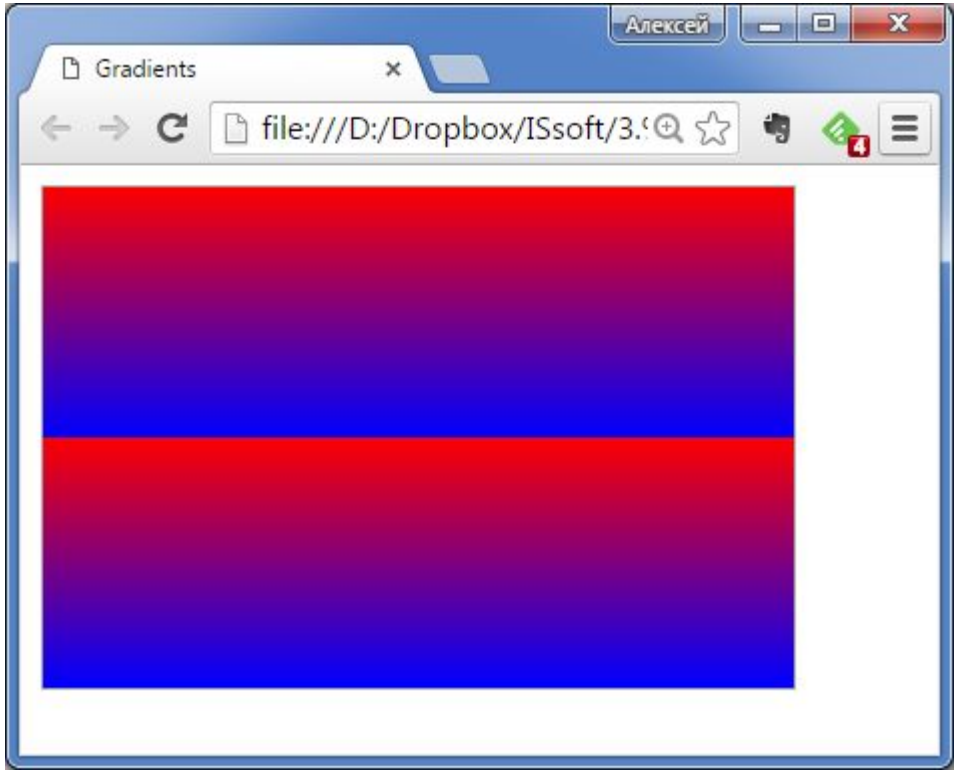
Линейный градиент

Функция `repeating-linear-gradient()` создаёт бесконечно повторяющийся линейный градиент, образуя фоновый узор.

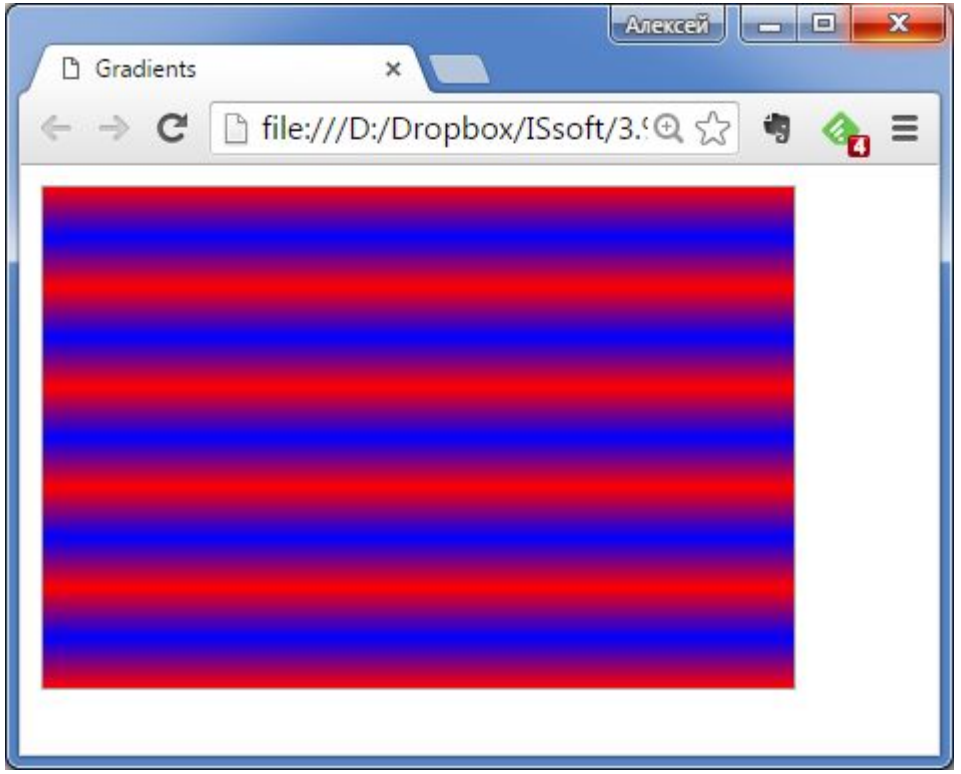
По своему действию похожа на `linear-gradient()` и имеет те же параметры. Обычно для цветов указывается **смещение на векторе градиента.** Цвета последовательно сменяют друг друга.

Линейный градиент

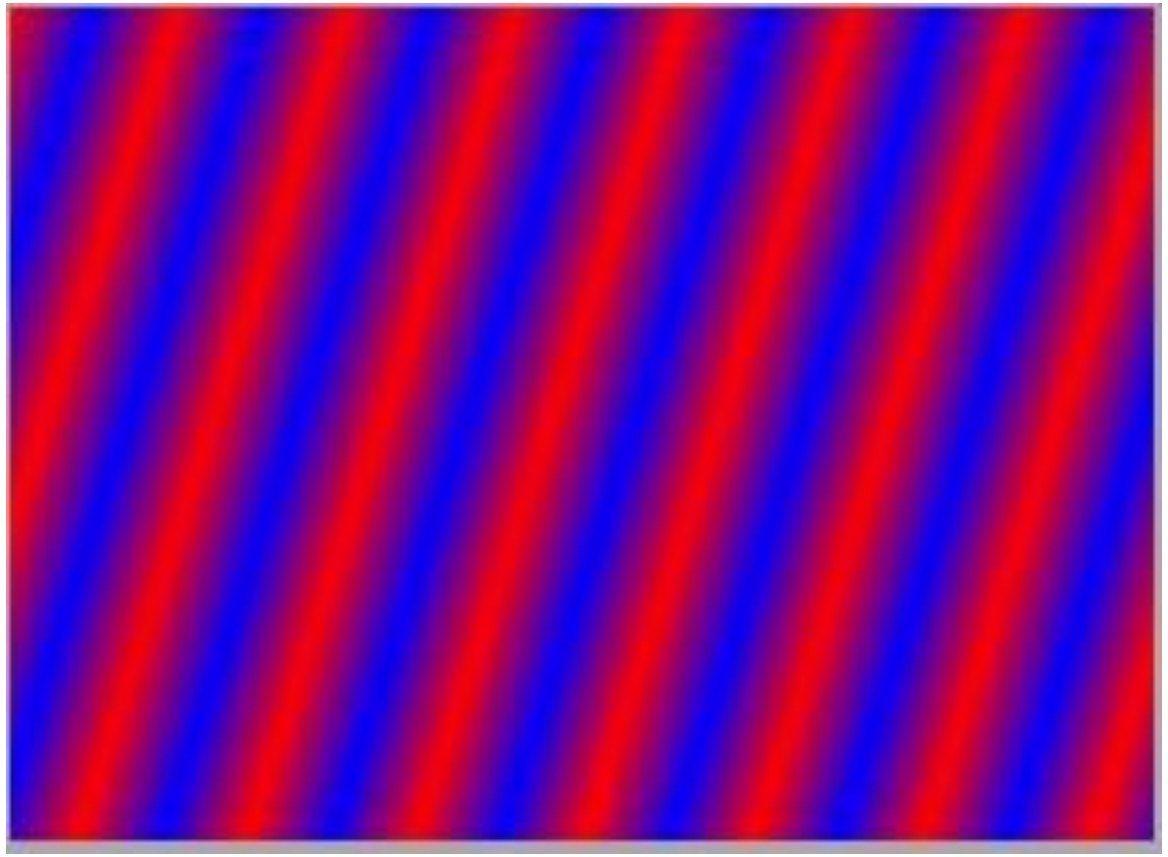
```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background: repeating-linear-gradient(red, blue 50%);
}
```



Линейный градиент




```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background: repeating-linear-gradient(red, blue 20px, red 40px);
}
```



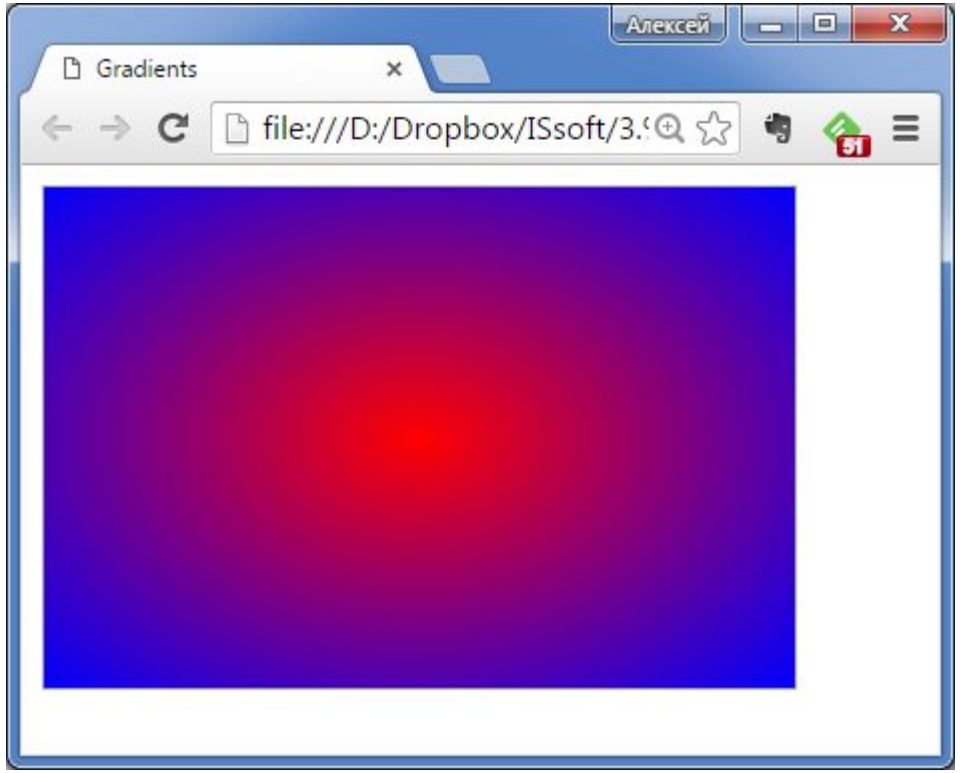
```
background:repeating-linear-gradient(110deg, red, blue 20px, red 40px);
```

Радиальный градиент

В радиальном градиенте переход между цветами происходит по окружности («круги на воде»). Для построения служит функция `radial-gradient()`:

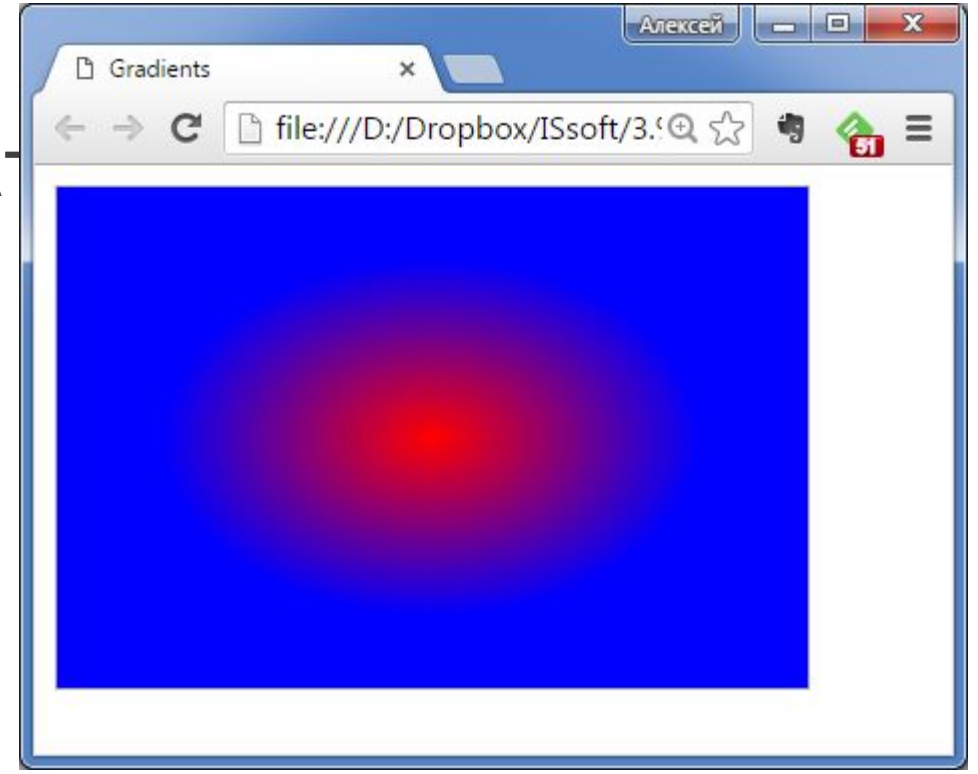
```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background-image: radial-gradient(red, blue);
}
```

Радиальный градиент



Радиальный градиент

Как и в случае линейного градиента, можно настраивать «остановки» для цветов:




```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background-image: radial-gradient(red, blue 50%);
}
```

Радиальный градиент

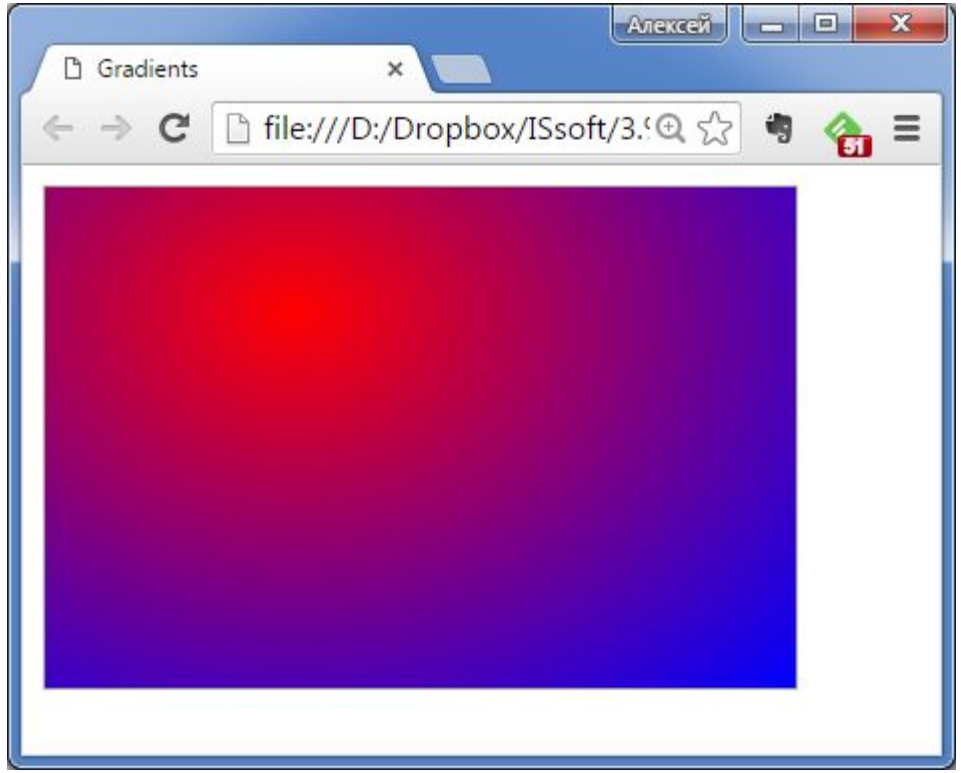
Ещё настраивается центр градиента. По умолчанию точкой отсчёта является середина (`at center center`). Её можно смещать аналогично `background-position`.

Первое значение определяет расположение по горизонтали, второе – по вертикали. Указываются после предлога `at` в любой единице измерения, в процентах или при использовании ключевых слов

```
[left|center|right] || [top|center|bottom]
```

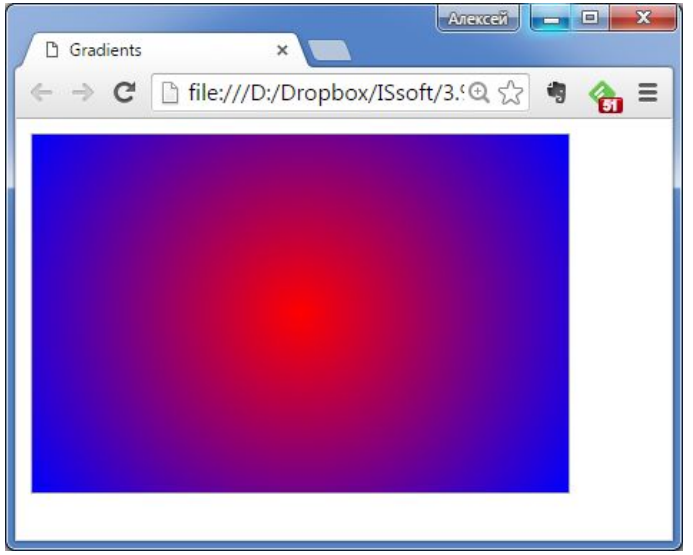
Радиальный градиент

```
.grd {
  width: 300px;
  height: 200px;
  border: 1px solid #aaa;
  background: radial-gradient(at 100px 50px, red, blue);
}
```



Радиальный градиент

В неквадратных блоках градиент работает как эллипс, но его поведение можно изменить на поведение круга:




```
.grd {
 width: 300px;
 height: 200px;
 border: 1px solid #aaa;
 background: radial-gradient(circle at 150px 100px, red, blue);
}
```

Радиальный градиент

Дополнительно можно настроить размер градиента. Делается это указанием специальных параметров после слова `circle` (или `ellipse`): `closest-side`, `closest-corner`, `farthest-side`, `farthest-corner`.

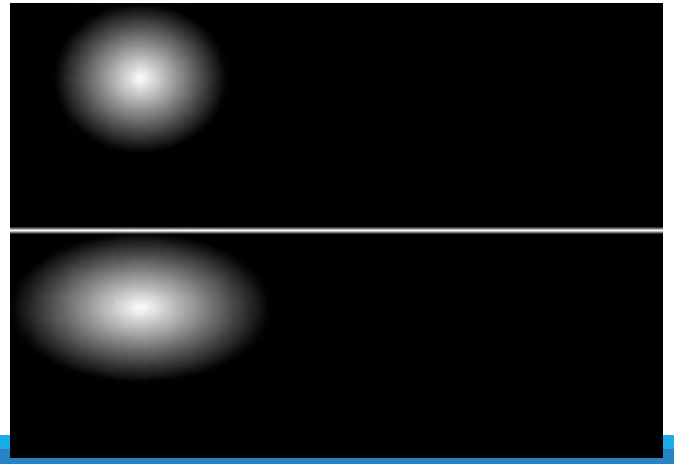
Вместо спец. параметров можно указать радиус градиента в единицах размера.

closest-side

Форма градиента совпадает с ближайшей к нему стороной блока.

```
background: radial-gradient(circle closest-side at 30px 20px, #fff, #000);
```

```
background: radial-gradient(ellipse closest-side at 30px 20px, #fff, #000);
```

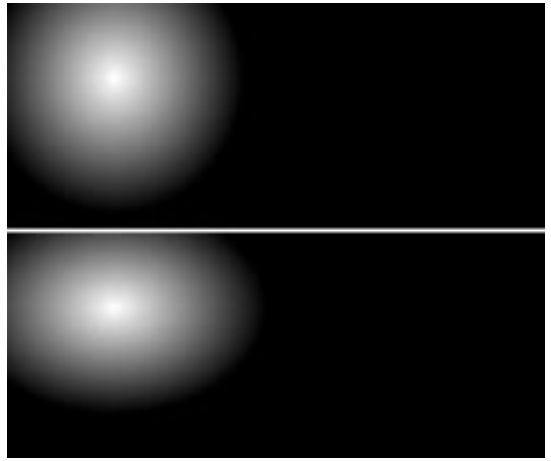


closest-corner

Форма градиента вычисляется на основании информации о расстоянии до ближайшего угла блока.

background: radial-gradient(circle closest-corner at 30px 20px, #fff, #000);

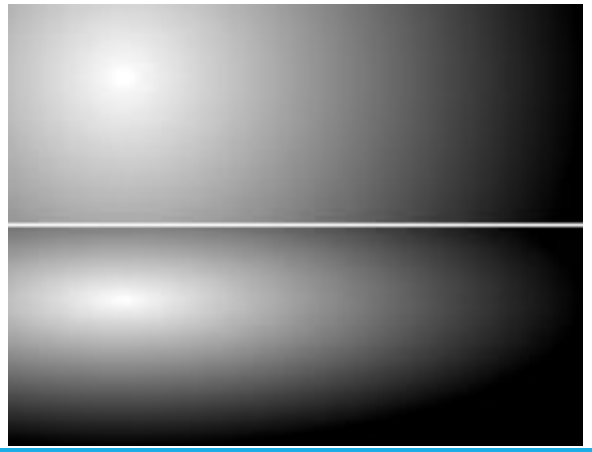
background: radial-gradient(ellipse closest-corner at 30px 20px, #fff, #000);



farthest-side

Градиент распространяется до дальней стороны блока.

```
background: radial-gradient(circle farthest-side at 30px 20px, #fff, #000);
background: radial-gradient(ellipse farthest-side at 30px 20px, #fff, #000);
```

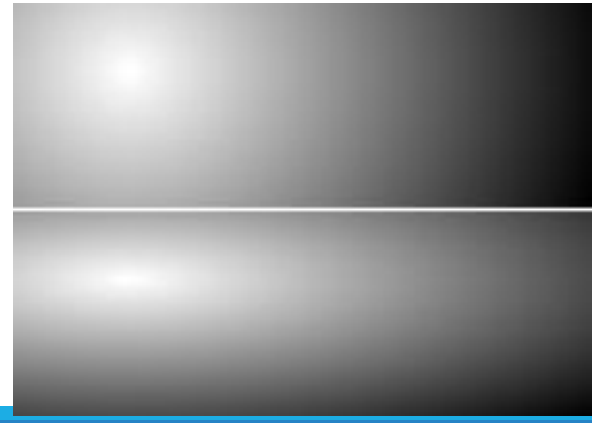


farthest-corner

Форма градиента вычисляется на основании информации о расстоянии до дальнего угла блока

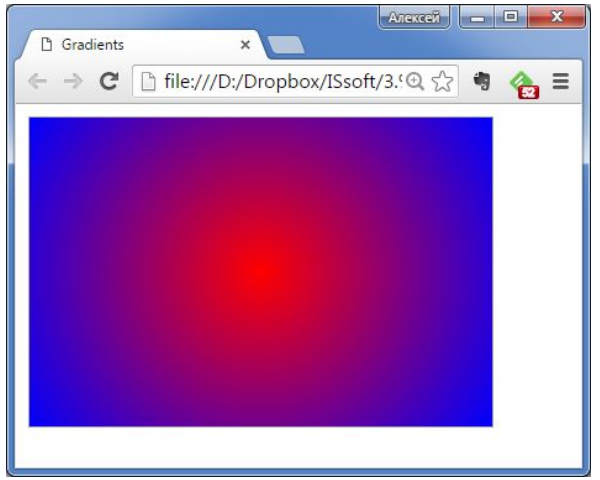
background: radial-gradient(circle farthest-corner at 30px 20px, #fff, #000);

background: radial-gradient(ellipse farthest-corner at 30px 20px, #fff, #000);

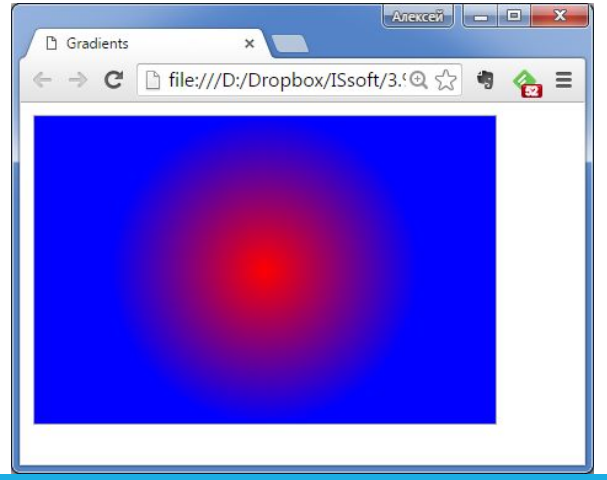


Радиальный градиент

```
background: radial-gradient(circle farthest-corner at 150px 100px, red, blue);
```



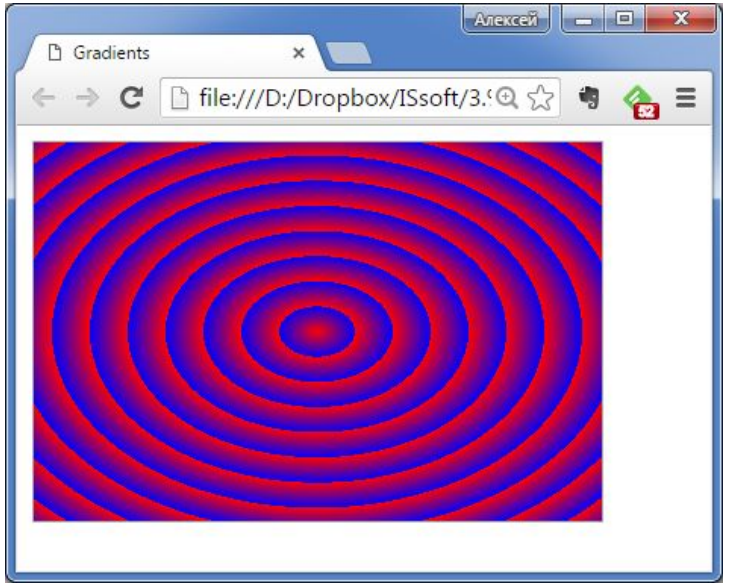

```
background: radial-gradient(circle closest-side at 150px 100px, red, blue);
```



Радиальный градиент

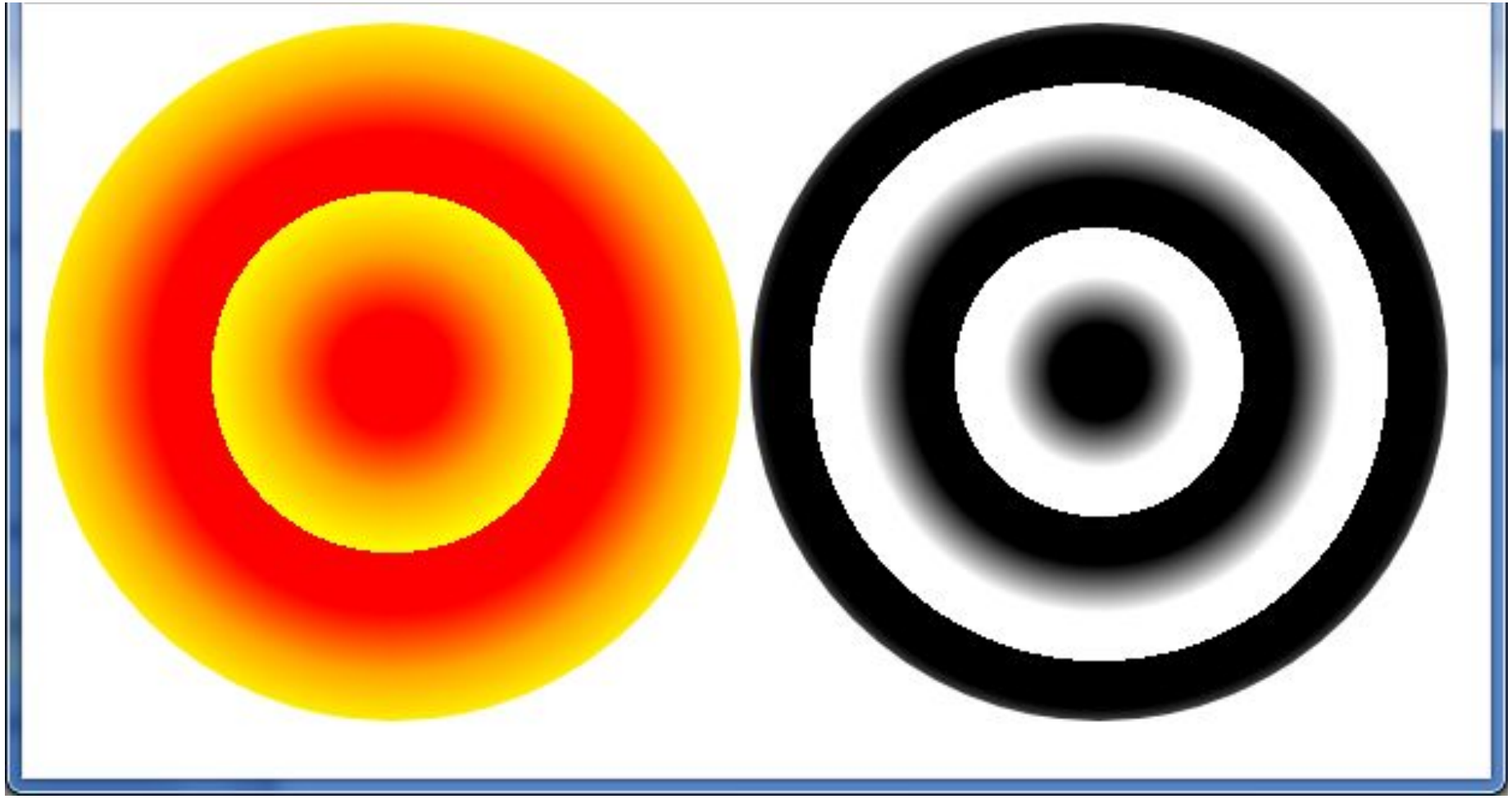
Функция `repeating-radial-gradient()` создаёт бесконечно повторяющийся радиальный градиент.

```
background: repeating-radial-gradient(red, blue 20px);
```

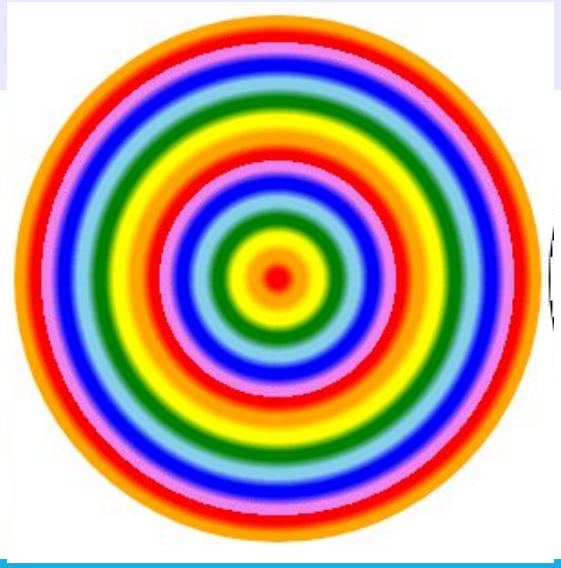


```
div {
height: 290px; /* устанавливаем высоту элемента */
width: 290px; /* устанавливаем ширину элемента */
display: inline-block; /* устанавливаем, что элементы <div>
становятся блочно-строчными (чтобы выстроились в
линейку) */
border-radius: 100%; /* задаем радиус скругления углов
элемента */
}
```

```
.test {
background-image: repeating-radial-gradient(red, red 25px, orange 50px, yellow 75px);
 /* повторяющийся градиент с точками остановки для каждого цвета через каждые 25px */
}
.test2 {
background-image: repeating-radial-gradient(black , black 20px, white 40px, white 60px);
/* повторяющийся градиент с точками остановки для каждого цвета через каждые 20px */
}
```



```
.test {background-image:
repeating-radial-gradient(red, red 5px,
orange 10px, orange 15px, yellow 20px, yellow 25px,
green 30px, green 35px, SkyBlue 40px, SkyBlue 45px,
blue 50px, blue 55px, violet 60px, violet 65px);
/* повторяющийся четырнадцатицветный градиент с
точками остановки для каждого цвета через каждые
5px */
```



Вендорные префиксы

Это приставки (перед именем CSS-свойства или функции), используемые производителями браузеров для экспериментальных CSS-свойств и функций.

`-o-` префикс для Opera

`-moz-` префикс для Mozilla

`-ms-` префикс для IE

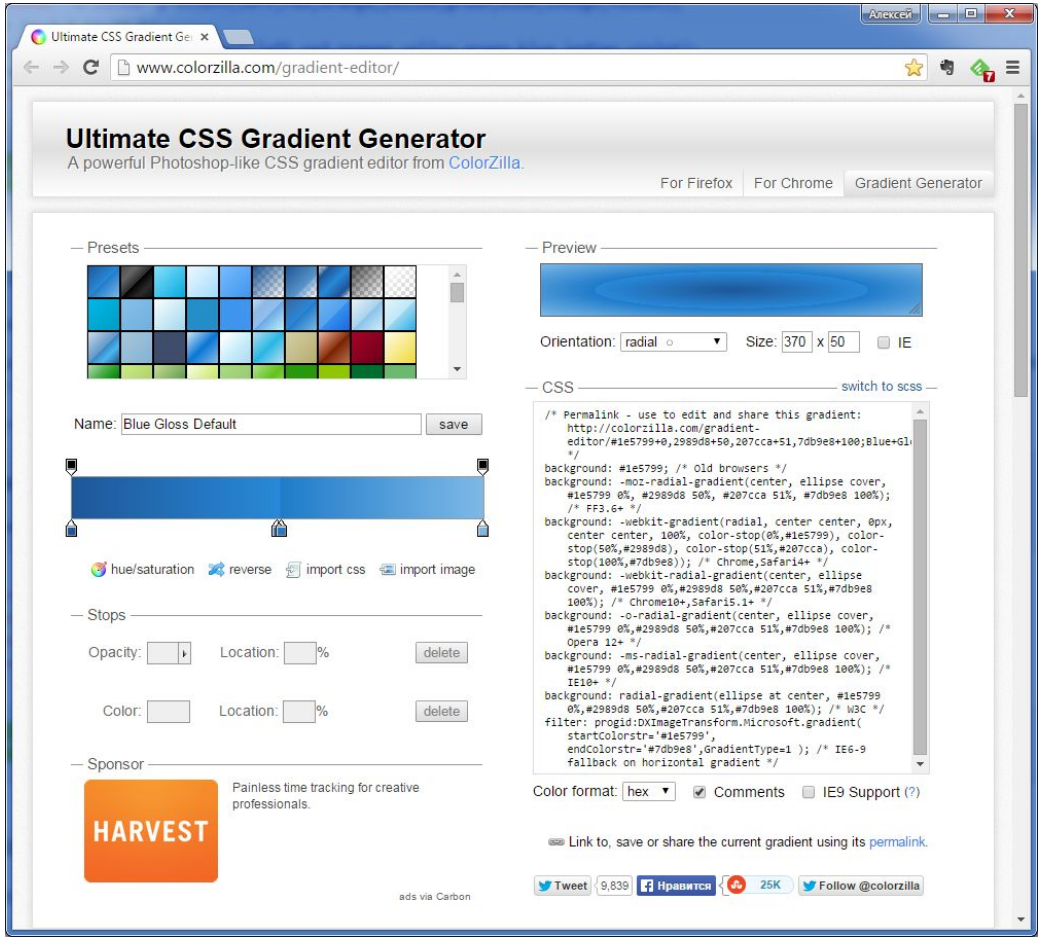
`-webkit-` префикс для Safari и Chrome

Вендорные префиксы и градиенты

Иногда встречается такой унаследованный код:

```
/* The old syntax, prefixed, for old browsers */
background: -moz-linear-gradient(top, blue, white);

/* The new syntax needed by standard-compliant browsers
(Opera 12.1, IE 10, Firefox 16, Chrome 26, Safari 6.1),
without prefix */
background: linear-gradient(to bottom, blue, white);
```

http://www.colorzilla.com/gradient-editor/



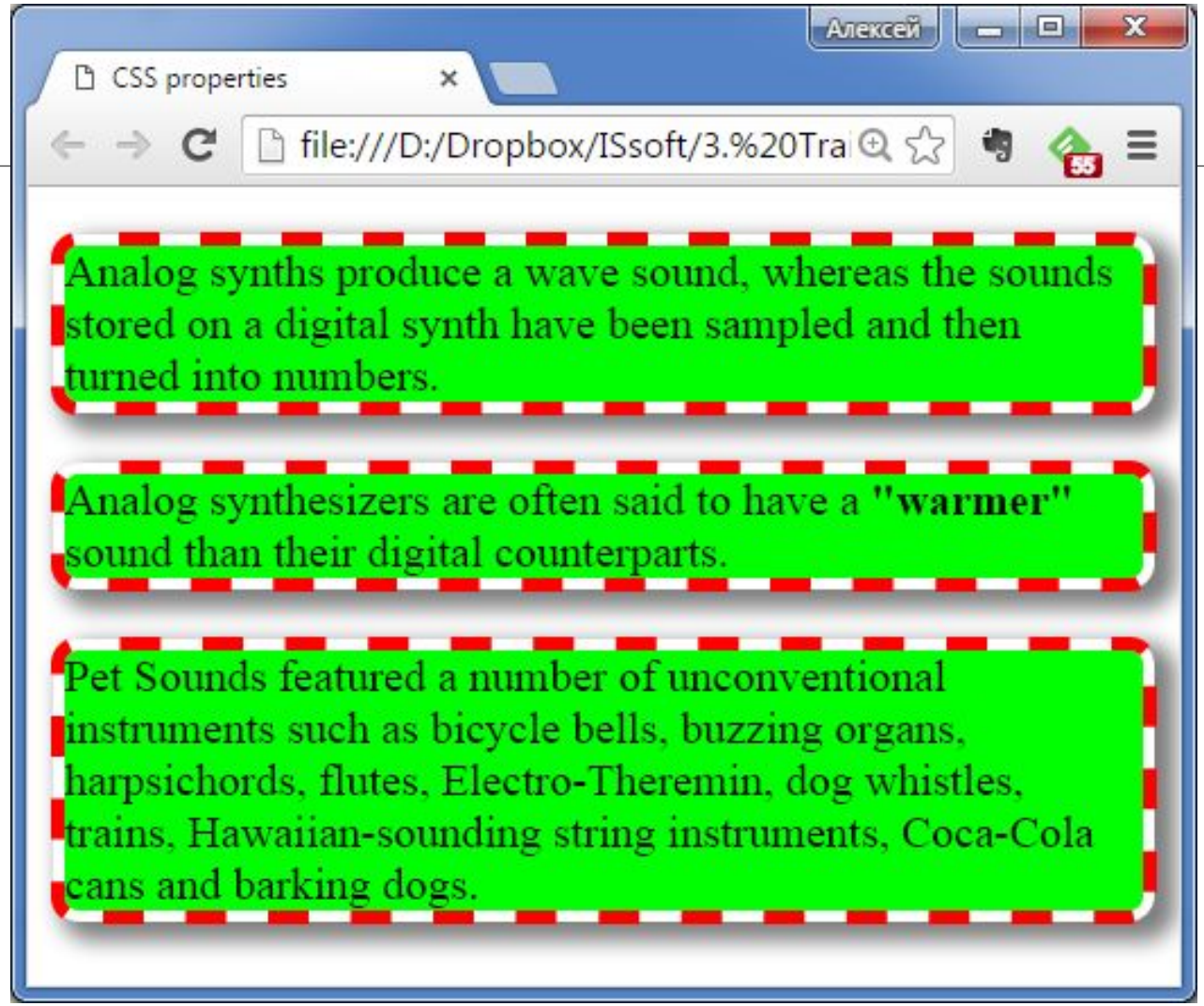
Тень блока

Свойство `box-shadow` добавляет к блоку тень (или даже несколько теней).

Синтаксис для тени такой:

<сдвиг по x> <сдвиг по y> <радиус размытия> <растяжение> <цвет>

```
p {
    border: 5px dashed red;
    border-radius: 10px;
    background-color: lime;
    background-clip: content-box;
    box-shadow: 5px 4px 10px 2px gray;
}
```


Алексей
CSS properties
file:///D:/Dropbox/ISsoft/3.%20Tra
Analog synths produce a wave sound, whereas the sounds stored on a digital synth have been sampled and then turned into numbers.
Analog synthesizers are often said to have a "warmer" sound than their digital counterparts.
Pet Sounds featured a number of unconventional instruments such as bicycle bells, buzzing organs, harpsichords, flutes, Electro-Theremin, dog whistles, trains, Hawaiian-sounding string instruments, Coca-Cola cans and barking dogs.

Размер блока. Отступы и поля

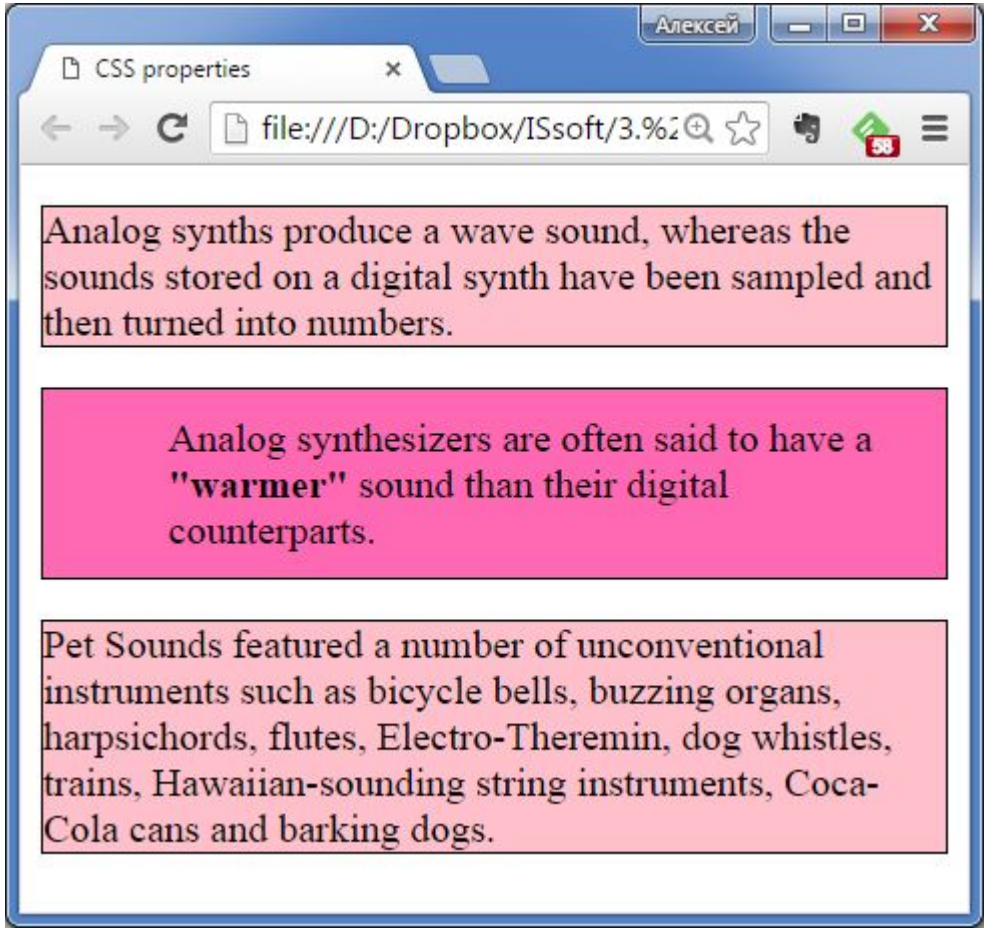
Отступы блока настраиваются через свойство `padding`, поля – через свойство `margin`.

При этом могут использоваться отдельные свойства для сторон блока (например, `padding-top`). Или можно указать одно-два-три-четыре значения через пробел.

Значения свойств – любая единица размера или проценты. Для `margin` можно указать `auto` (размер будет автоматически рассчитан браузером).

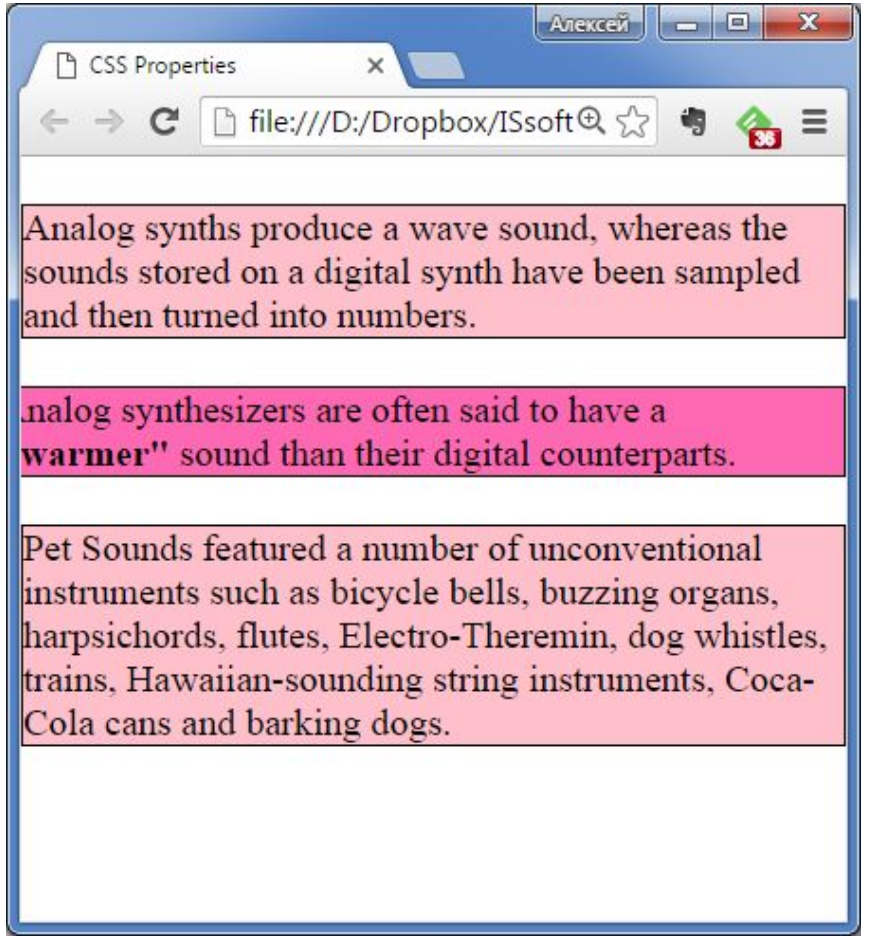
****Отступы и поля – пример 1

```
p {
  border: 1px solid black;
  background-color: pink;
}

p.two {
 background-color: hotpink;
 padding: 10px 20px 10px 50px;
}
```



Отступы и поля – пример 2

```
* {
  padding: 0;
  margin: 0;
}
p {
  border: 1px solid black;
  background-color: pink;
  margin: 20px 0;
}
p.two {
  background-color: hotpink;
  margin-left: -10px;
}
```



Ширина и высота

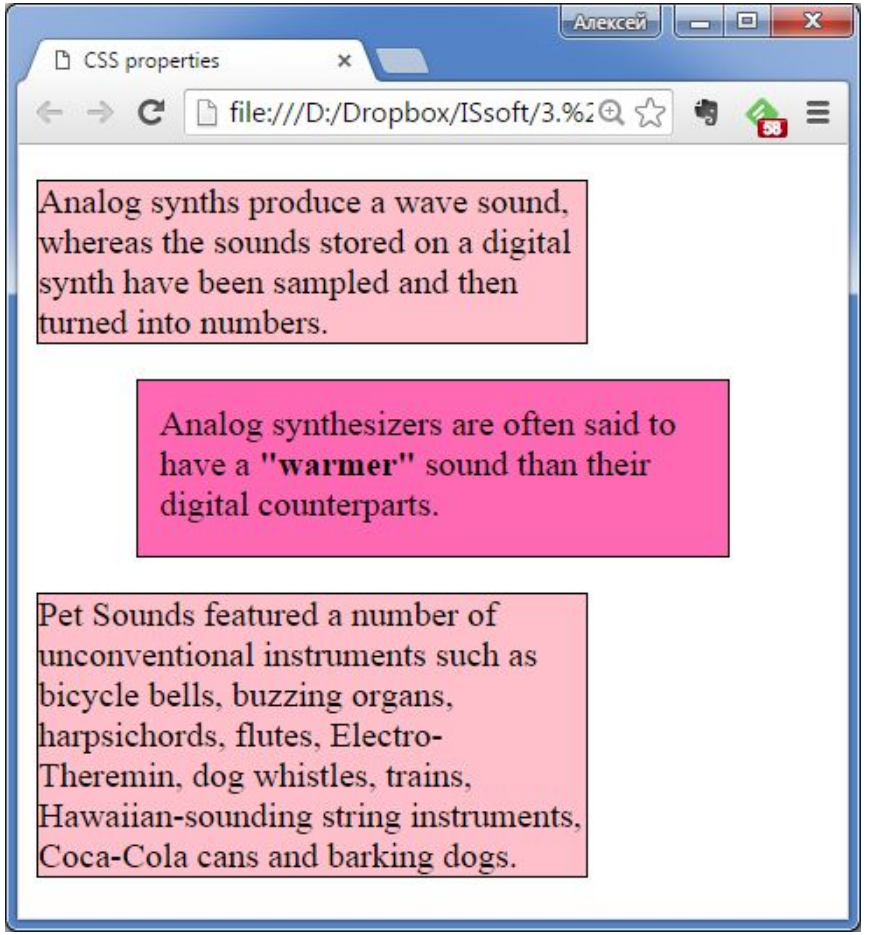
У блока можно принудительно задать ширину (свойство `width`) и высоту (свойство `height`).

Для значений этих свойств можно использовать любую допустимую в CSS единицу измерения.

Если у блока задана ширина, а значение левого и правого поля установлены в значение `auto`, то блок центрируется на странице.

Ширина и высота – пример

```
p {
    border: 1px solid black;
    background-color: pink;
    width: 250px;
}

p.two {
    background-color: hotpink;
    padding: 10px;
    margin-left: auto;
    margin-right: auto;
    height: 60px;
}
```



Работа с шириной

Есть три свойства, связанные с шириной, которые у блока можно установить в значение `auto`:

`margin-left`, `width`, `margin-right`.

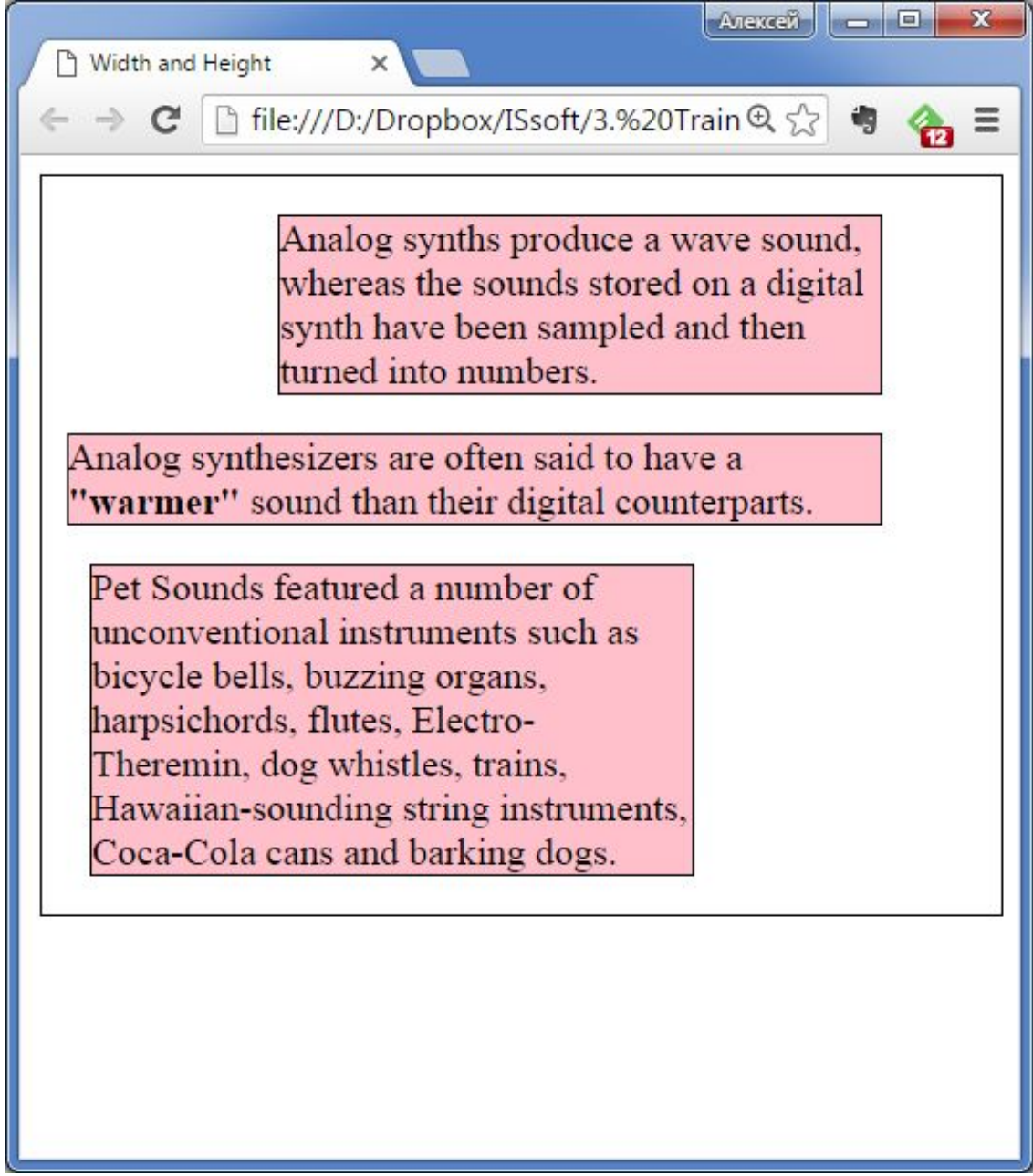
Кстати, по умолчанию

```
margin-left: 0;
width: auto;
margin-right: 0;
```

Пусть `auto` используется **ровно для одного** из свойств.

В этом случае `auto`-свойство получает значение, чтобы общая ширина блока была равна общей ширине родительского элемента.

```
body{border: 1px solid black;
     width: 400px;}

p {border: 1px solid black;
   background-color: pink;}

p.one {margin-left: auto;
       width: 250px;
       margin-right: 50px;}

p.two {margin-left: 20px;
       width: auto; /*by default*/
       margin-right: 50px;}

p.three{margin-left:30px;
        width:250px;
        margin-right:auto;}
```

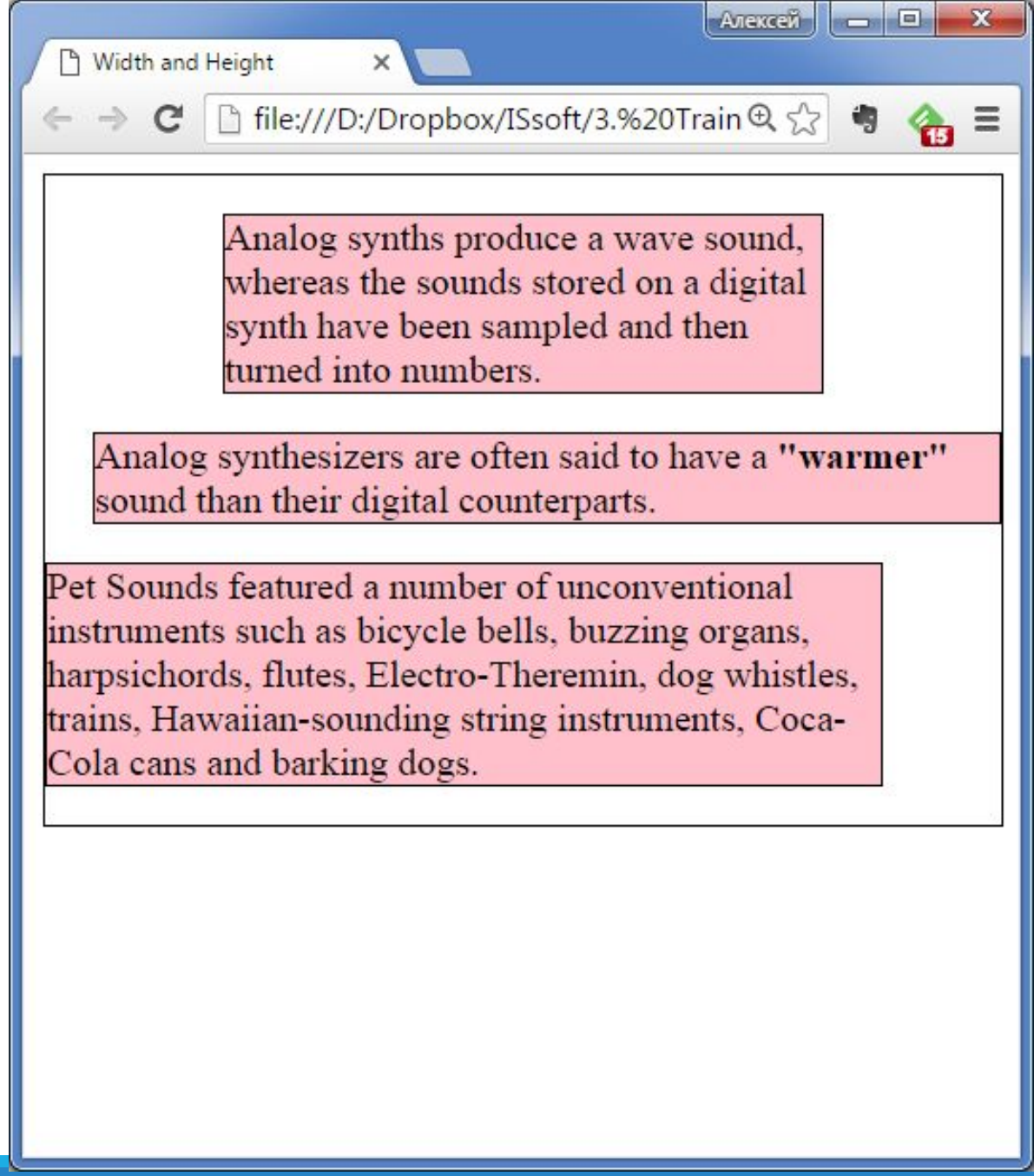


Пусть `auto` используется **ровно для двух** свойств.

Если это `margin-*`, то блок центрируется в родителе.

Если это `width` и `margin-left` (или `width` и `margin-right`), то `auto` для `margin-*` будет означать `0`.

```
body{border: 1px solid black;
     width: 400px;}

p{border: 1px solid black;
  background-color: pink;}

p.one{margin-left: auto;
      width: 250px;
      margin-right: auto;}

p.two{margin-left: 20px;
      width: auto;/*by default*/
      margin-right: auto;}

p.three{margin-left: auto;
        margin-right: 50px;}
```

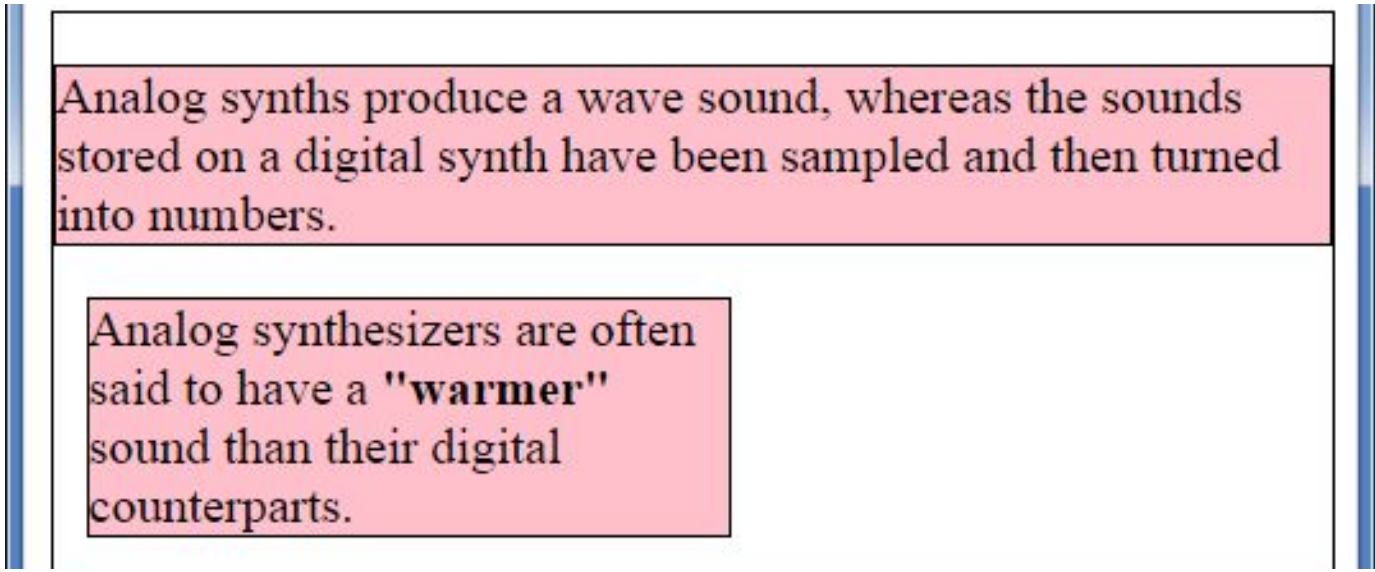


Пусть `auto` используется **для всех трёх** свойств.

В этом случае оба `margin-*`, устанавливаются в `0`, а свойство `width` становится равным ширине родителя.

Если же заданы **конкретные значения** для всех трёх свойств (`margin-left`, `width`, `margin-right`), то `margin-right` принудительно получает значение `auto`.

```
body{border: 1px solid black;
     width: 400px;}

p { border: 1px solid black;
    background-color: pink;}

p.one{margin-left: auto;
      width: auto;
      margin-right: auto;}

p.two{margin-left: 10px;
      width: 200px;
      margin-right: 50px;}
```



Особый случай: *заменяемые* элементы (например, `img`).

У таких элементов при установке `width` в `auto` реальное значение `width` берётся из замещающего контента.

Работа с высотой

Есть три свойства, связанные с высотой, которые у блока можно установить в значение `auto`: `margin-top`, `height`, `margin-bottom`.

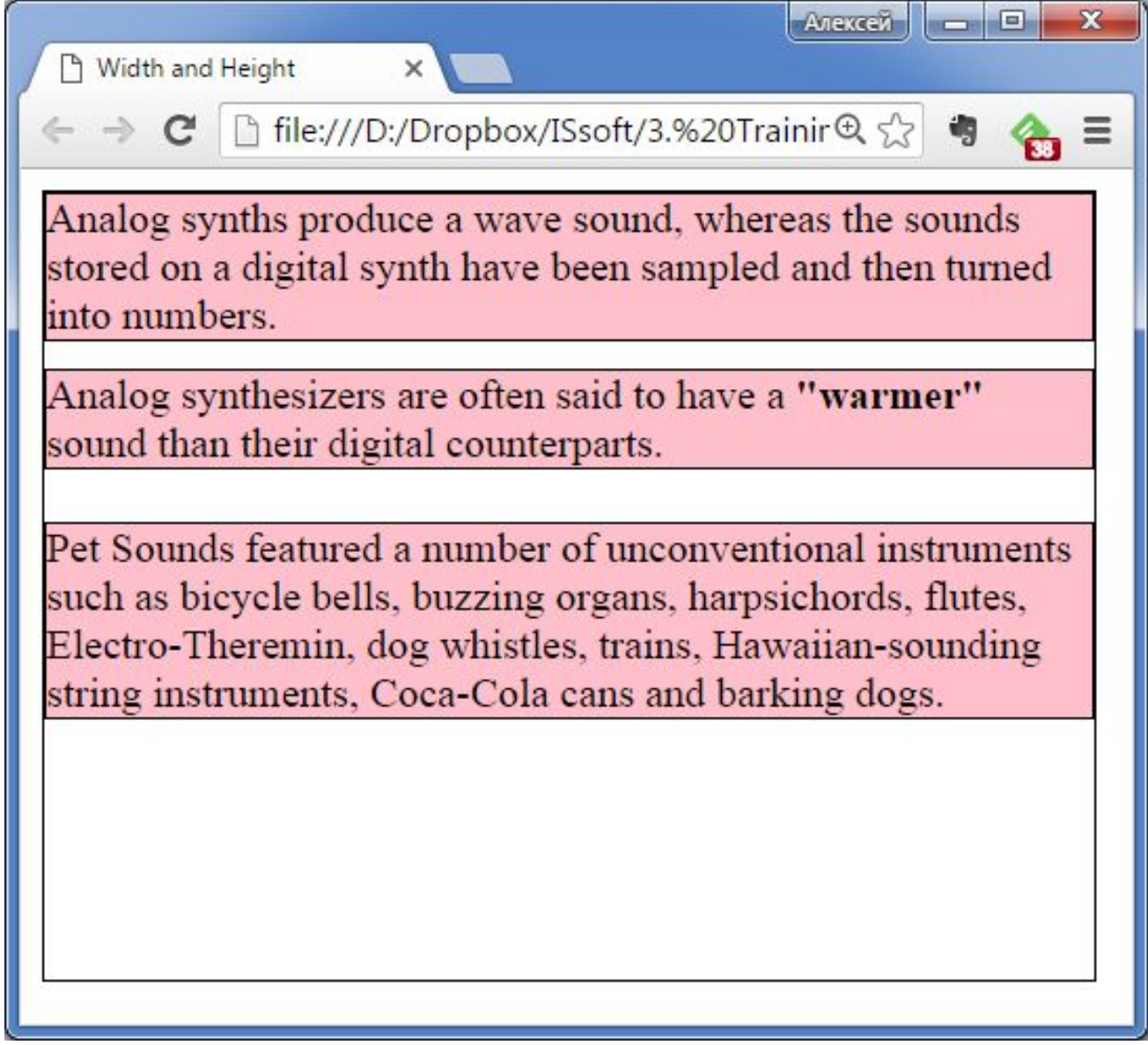
Кстати, по умолчанию

```
margin-top: 0;
height: auto;
margin-bottom: 0;
```

Установка любого из `margin-top` или `margin-bottom` в `auto` эквивалентна обнулению!

И помним про вертикальное схлопывание отступов.

```
body{border: 1px solid black;
     width: 400px;
     height: 300px;}

p {border: 1px solid black;
   background-color: pink;}

p.one{margin-top: auto;
      height: auto;
      margin-bottom: auto;}

p.two{margin-top: 10px;
      margin-bottom: 20px;}

p.threetwo{margin-top: 20px;
           margin-bottom: 20px;}
```

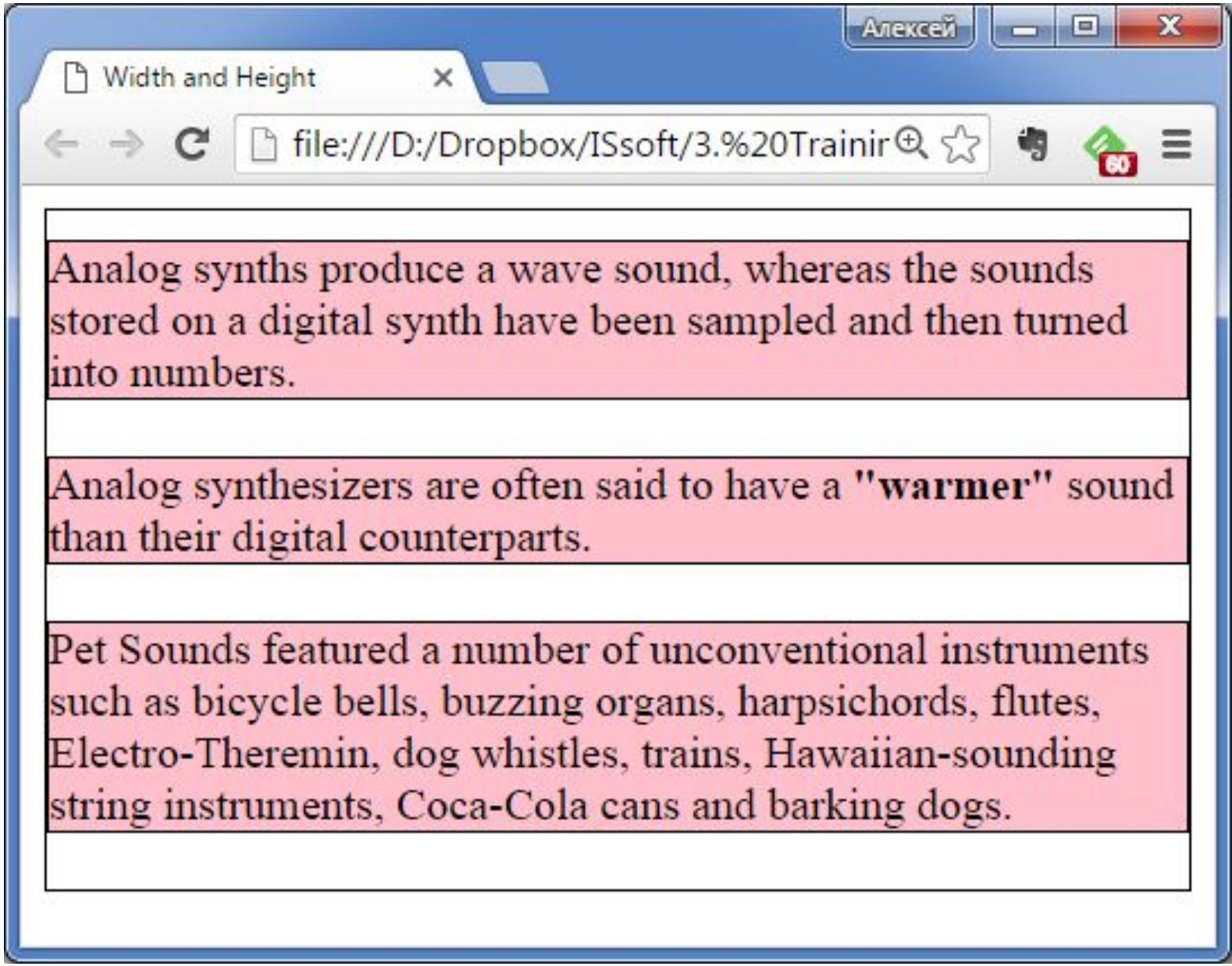


Пусть у блока `height` установлено в `auto`. Фактическая высота будет зависеть от наличия у этого блока границ и отступов (`padding-*`):

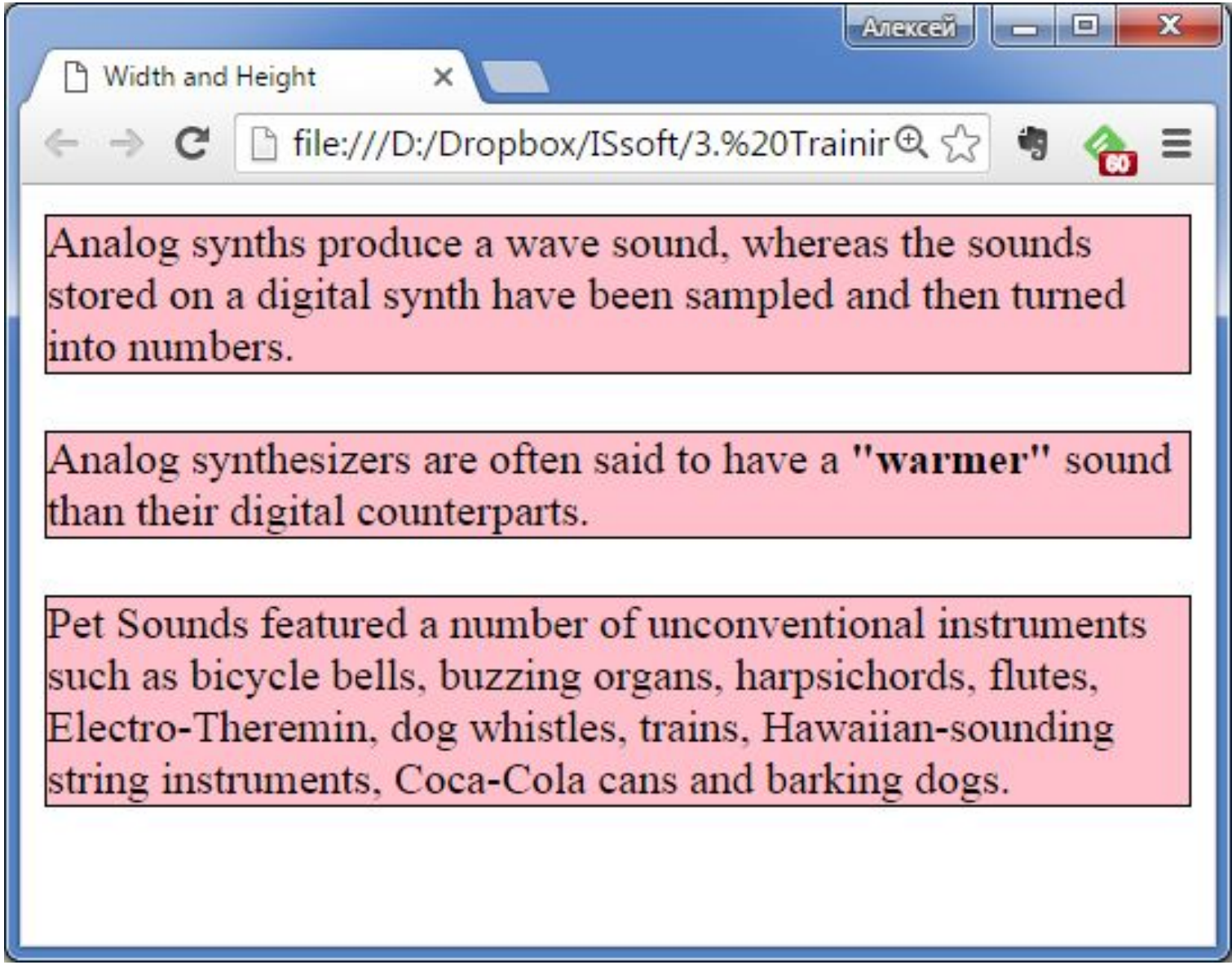
1. Границы и (или) отступы есть: учитываем вертикальные `margin` детей

2. Границ и отступов нет: не учитываем вертикальные `margin` детей

```css
body {
  border: 1px solid black;
}

p {
  border: 1px solid black;
  background-color: pink;
  margin-top: 10px;
  margin-bottom: 20px;
}
```



```
body {

}

p {
  border: 1px solid black;
  background-color: pink;
  margin-top: 10px;
  margin-bottom: 20px;
}
```

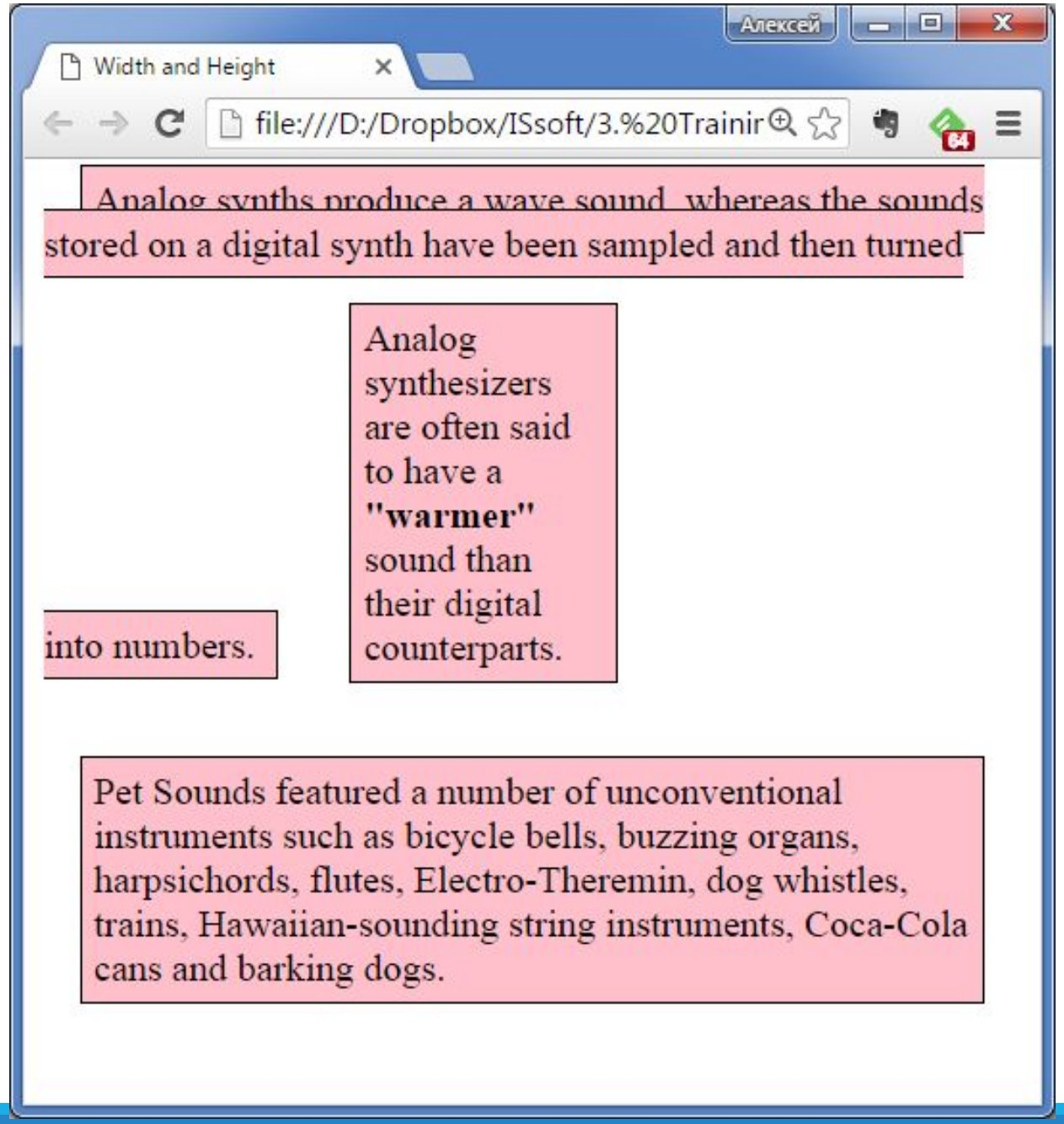


Всё, что описывалось раньше справедливо **для блочных элементов**.

Для элементов `display: inline (делает элемент строчным)` свойства `width` и `height` вообще не играют роли!

Однако эти свойства работают, если `display: inline-block`. Элемент ведет себя как блочный, включая все свойства блочной модели , но отображается на строке с другими элементами, а не начинается с новой строки по умолчанию.

```css
p {
    border: 1px solid black;
    background-color: pink;
    padding: 5px;
    margin: 15px;
}
p.one {
    display: inline;
    width: 100px;
    height: 20px;
}
p.two {
    display: inline-block;
    width: 100px;
}
```



Свойство `overflow` управляет отображением, если содержимое выходит за размеры блока (длиннее, чем его контейнер) :

`visible`показывается все содержимое, даже за пределами установленной высоты и ширины

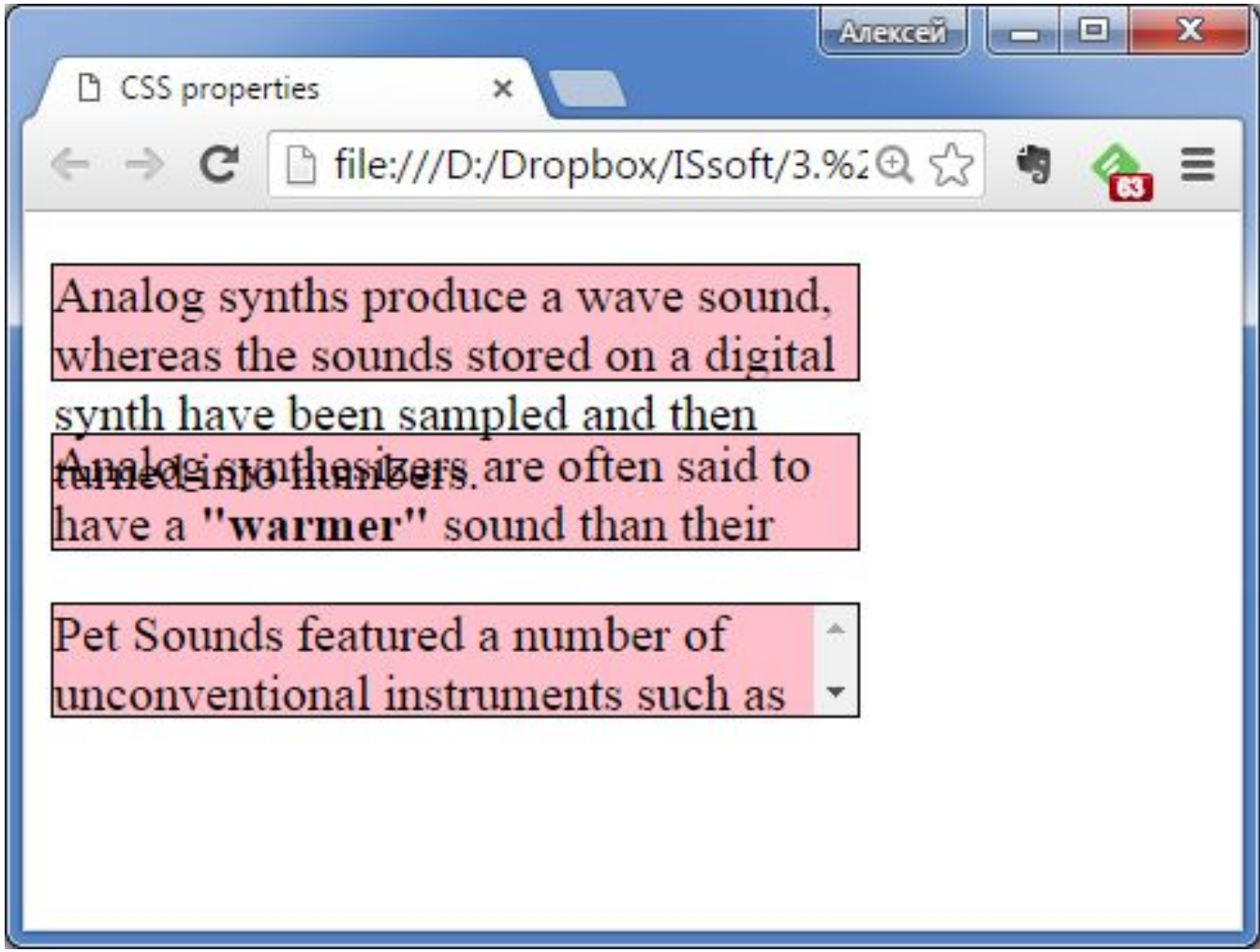
`hidden` отображается только область внутри элемента, остальное будет скрыто

`scroll` всегда добавляются полосы прокрутки

`auto` полосы прокрутки добавляются при необходимости , если содержимое будет переполнено

Есть отдельно overflow-x и overflow-y

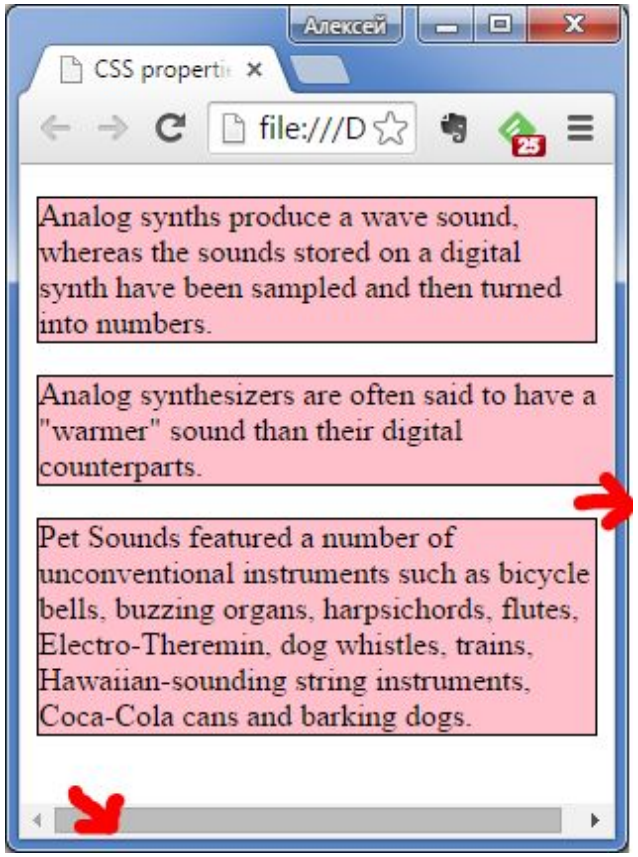
```
p {
    border: 1px solid black;
    background-color: pink;
    width: 250px;
    height: 35px;
}

p.one { overflow: visible; }

p.two { overflow: hidden; }

p.three { overflow: auto; }
```

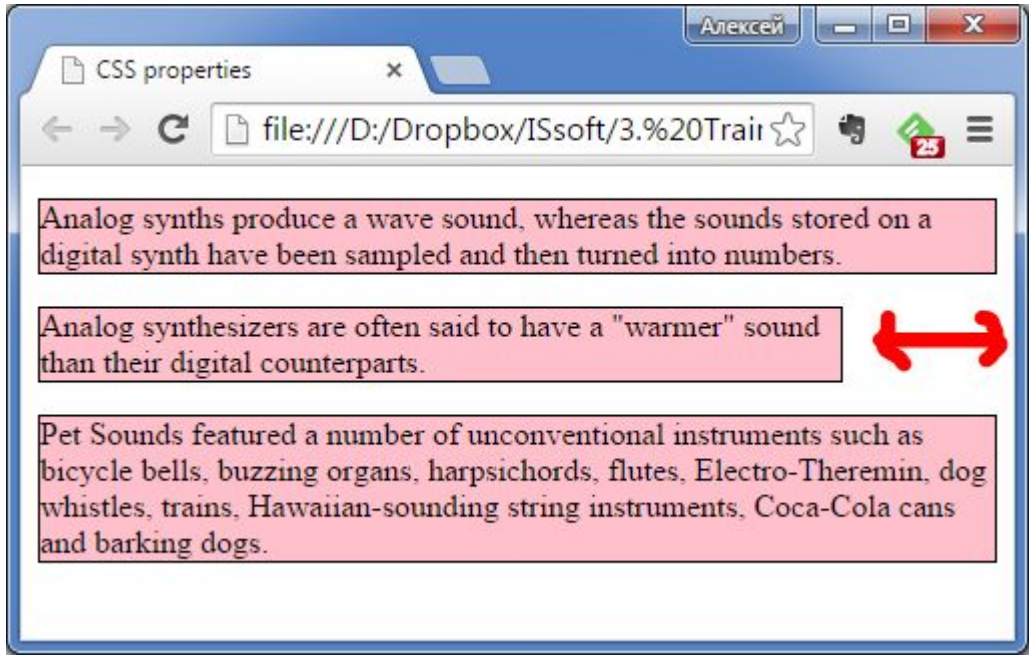


Если у блока не задана ширина, он растягивается вместе со своим родительским элементом (например, окном браузера).

Контролировать границы растяжения можно при помощи свойств `min-width` и `max-width`.

Существуют также свойства `min-height` и `max-height`.






```
p.two {
    min-width: 300px;
    max-width: 400px;
}
```

Свойство box-sizing

Общая ширина блока= **width** + **padding** + **border** + **margin** (аналогично высота). Это алгоритм по умолчанию.

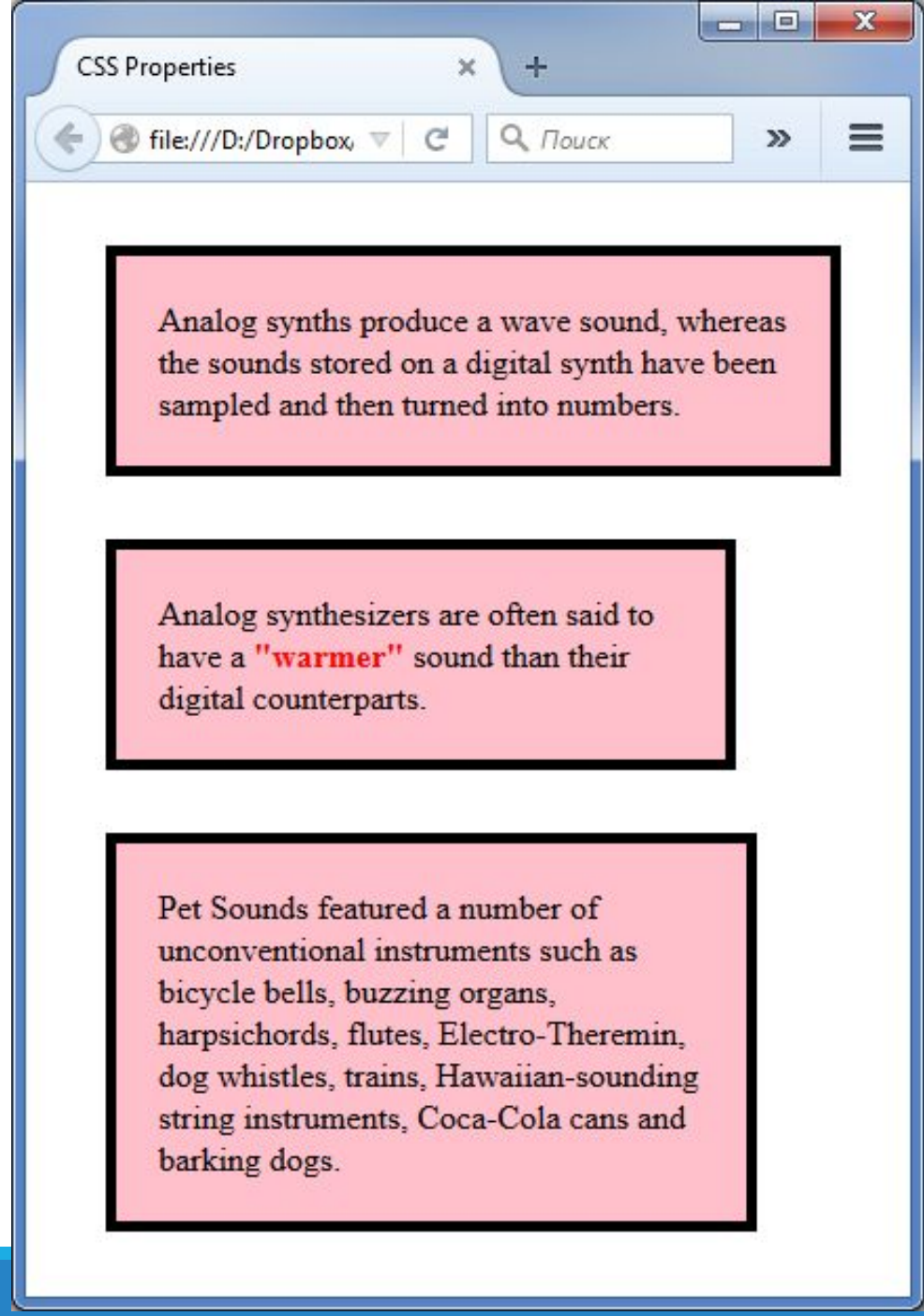
Свойство `box-sizing` позволяет изменить алгоритм:

`content-box` алгоритм по умолчанию

`border-box` width и height включают контент, padding, border, но не margin (IE в режиме совместимости)

`padding-box` (убрали из спецификации) width и height включают контент и padding, но не margin и border (поддерживает только Firefox)

```
p {
    border: 5px solid black;
    background-color: pink;
    width: 300px;
    padding: 20px;
    margin: 30px;
}

p.one {box-sizing: content-box;}
p.two {box-sizing: border-box; }
p.three {box-sizing: padding-box;}
```

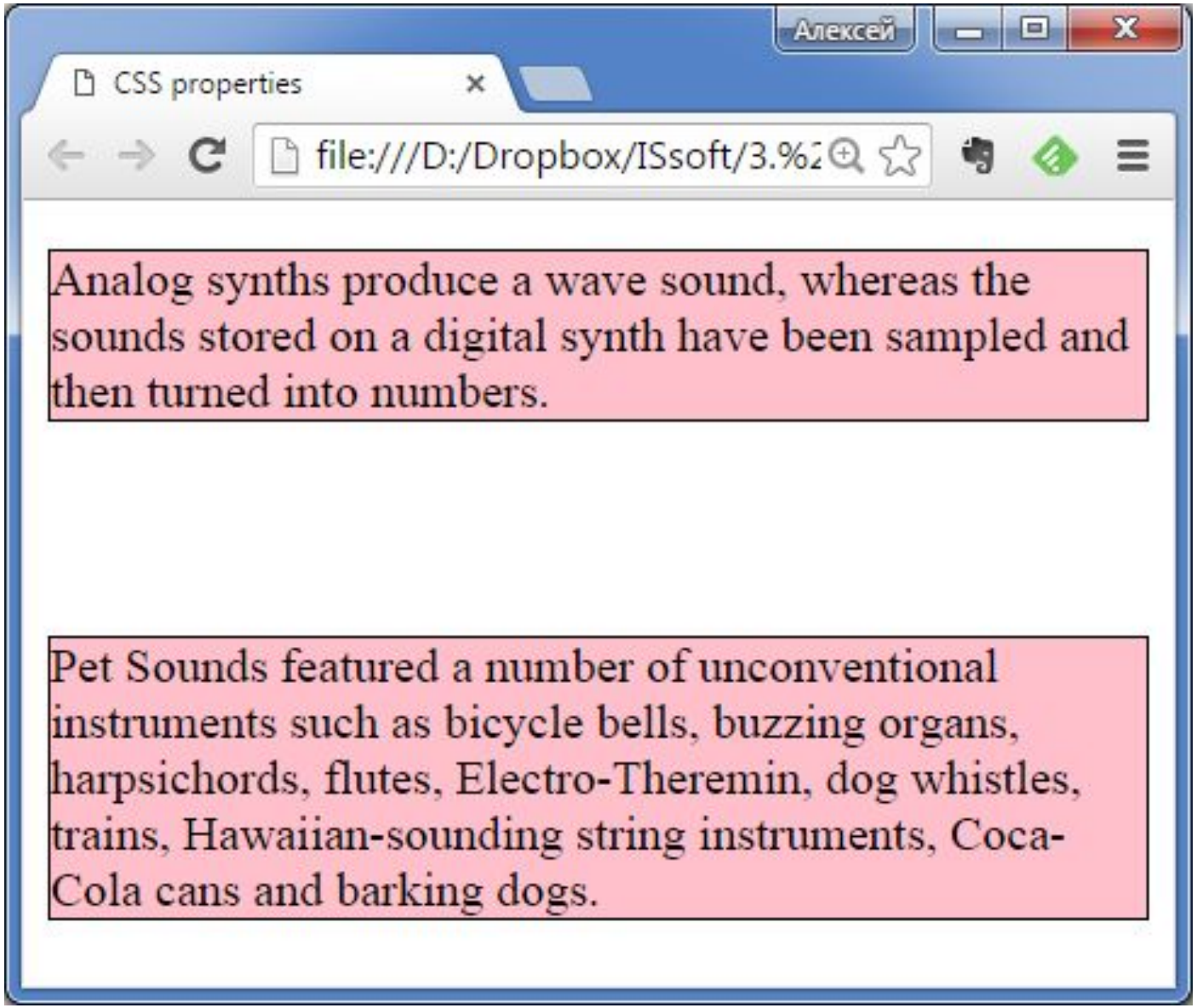


Видимость элемента

Видимость элемента контролируется свойством `visibility`.

При значении `visible` элемент отображается, при значении `hidden` элемент становится невидимым (вернее – *полностью прозрачным*, так как продолжает участвовать в формировании страницы).

```
p.two {
    visibility: hidden;
}
```



Свойство `display` контролирует видимость **и поток элемента**. *Некоторые* из его возможных значений:

`none` элемент не виден, и веб-страница формируется так, словно элемента и не было

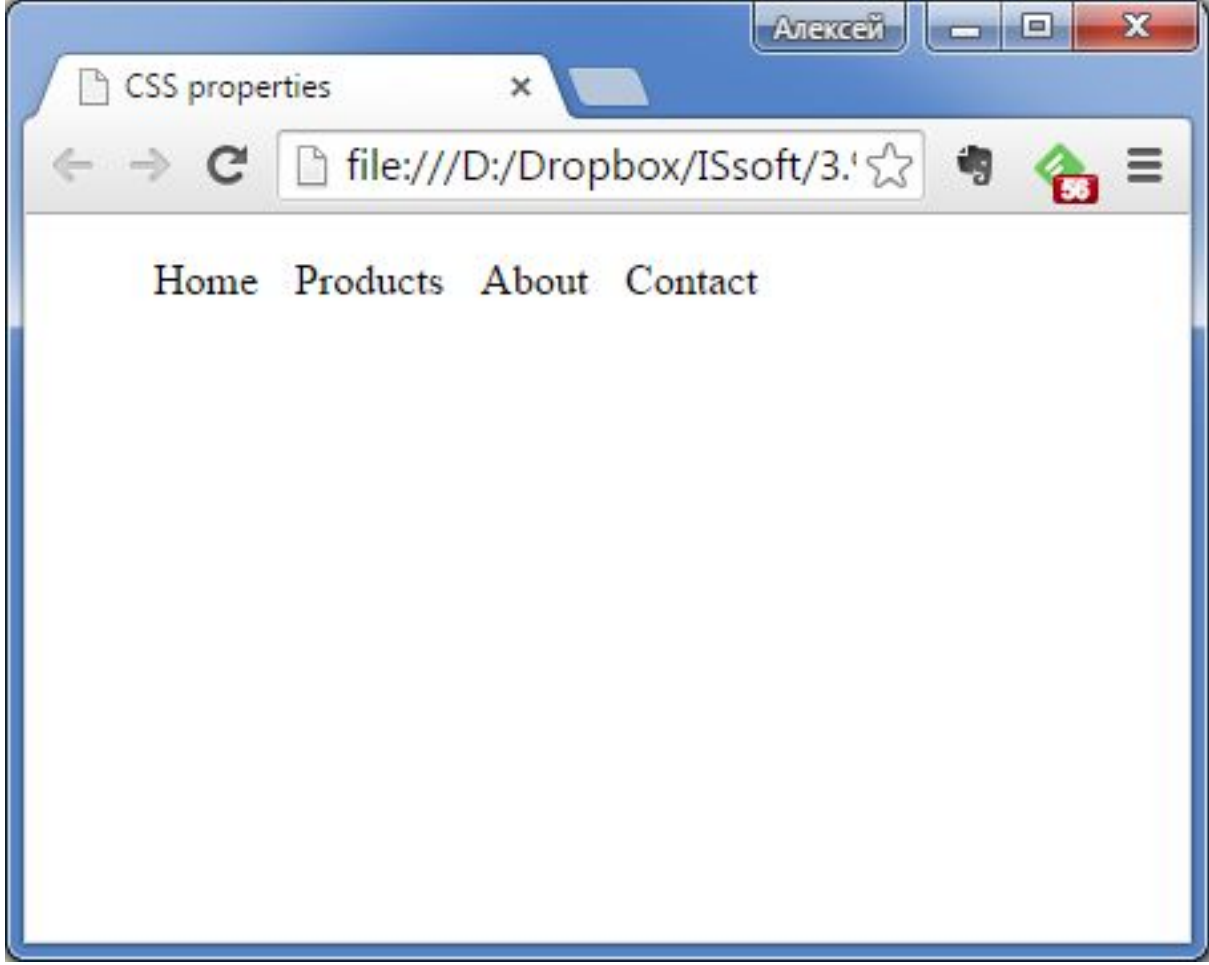
`block` элемент показывается как блочный (`div`)

`inline` элемент отображается как встроенный - строчный (`span`)

`list-item` элемент выводится как блочный и добавляется маркер списка

`inline-block` – элемент ведет себя как блочный, включая все свойства блочной модели ,но будет отображаться в строке с другими элементами

```css
li {
        display: inline;
        margin-right: 10px;
    }
    li.coming-soon {
        display: none;
    }
```




```html
<ul>
    <li>Home</li>
    <li>Products</li>
    <li class="coming-soon">Services</li>
    <li>About</li>
    <li>Contact</li>
</ul>
```

Позиционирование блоков

По умолчанию блоки позиционируются (располагаются) на странице слева направо и сверху вниз.

Все блочные элементы являются **гибкими:**

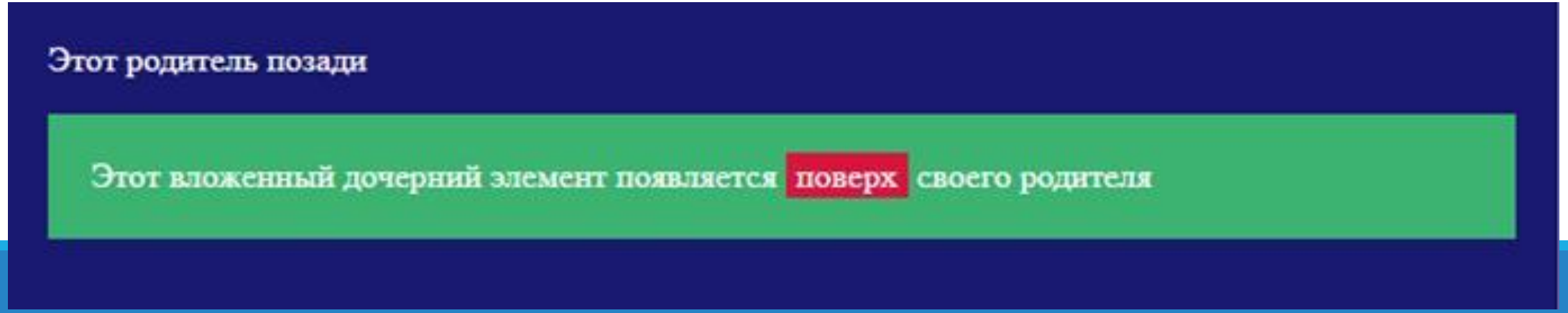
width: 100% – вся доступная ширина

Перенос слов

height: auto – соответствует размеру содержимого

Элементы HTML отображаются в том **порядке**, в котором они записаны **в коде – сверху вниз**.
Каждый элемент принадлежит воображаемому **слою**.
дочерние элементы появляются **поверх** своих родителей.
Чем **глубже** элемент по иерархии, тем *выше* в наложении.

```
div>
  Этот родитель позади
  <p>
    Этот вложенный дочерний элемент появляется <strong>поверх
</strong> своего родителя
  </p>
</div>
```



- height и width могут поменять изменчивость элемента;
- float нарушает поведение элемента, а также его окружения;
- значения absolute и fixed у свойства position удаляют элемент из потока;
- z-index может менять порядок наложения элементов.

свойство **position** сообщает браузеру, какой тип позиционирования используется для элемента:
статический - **static** – по умолчанию классическое размещение в потоке (сверху вниз),
абсолютный – **absolute**,
относительный - **relative**,
фиксированный - **fixed**,
сочетание относительного и фиксированного позиционирования – **sticky**

значение свойства **position** **не наследуется**

Абсолютное позиционирование

```
position: absolute;
```

Позиционирование относительно заданного края предка элемента

предок **должен иметь** position отличное от **static** (по умолчанию),

иначе (если у родителя значение position:static или родителя нет) смещение **относительно, указанного края окна браузера**

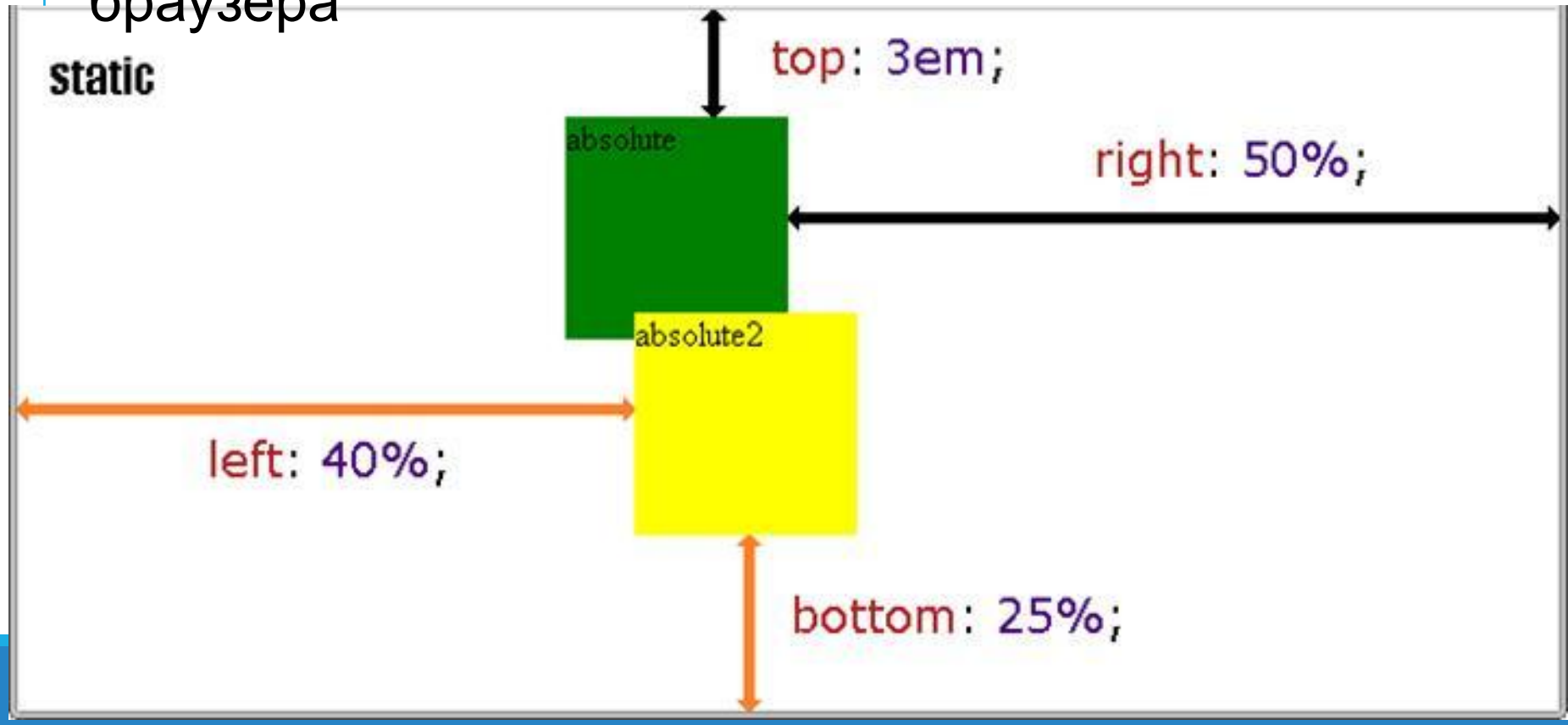
свойства, управляющие смещением позиционированного элемента от края родителя или окна:top , right , bottom , left .

```
div {
position: absolute;
width: 100px;
height: 100px;}
```

```
.absolute {
top: 3em; /* смещение от верхнего края */
right: 50%; /* смещение от правого края */
background-color: green; /* цвет заднего фона */
}
```

элементы позиционируются относительно окна браузера

```
.absolute2 {
bottom: 25;
left: 40%;
background-color:
yellow;
}
```



Относительное позиционирование

`position:relative;` относительно текущей позиции элемента (исходного места в нормальном потоке)

```
.static {   /*static по умолчанию*/
height: 50px;
background-color: red;
}

.relative {
position: relative;
height: 100px;
top: 50px;
left: 100px;
background-color: green;
}
```

```
<div class = "static">
static
</div>
<div class = "relative">
relative
</div>
<div class = "static">
static
</div>
```

браузер зарезервировал место под элемент, где он находился бы до перемещения

Фиксированное позиционирование

`position:fixed;` **всегда** относительно заданного края окна браузера

элемент остается на одном месте при прокрутке страницы!

```
html, body {height: 100%;
    margin: 0; /*убрали
встроенные стили браузера*/
        }
.fixed {position: fixed;
    height: 100%;
    width: 15%;
    background-color: red;
    right: 0;   }
.container {height: 2000px; }
```

```
<body>
    <div class = "fixed">
    fixed
    </div>
    <div class = "container">
</div>
</body>
```

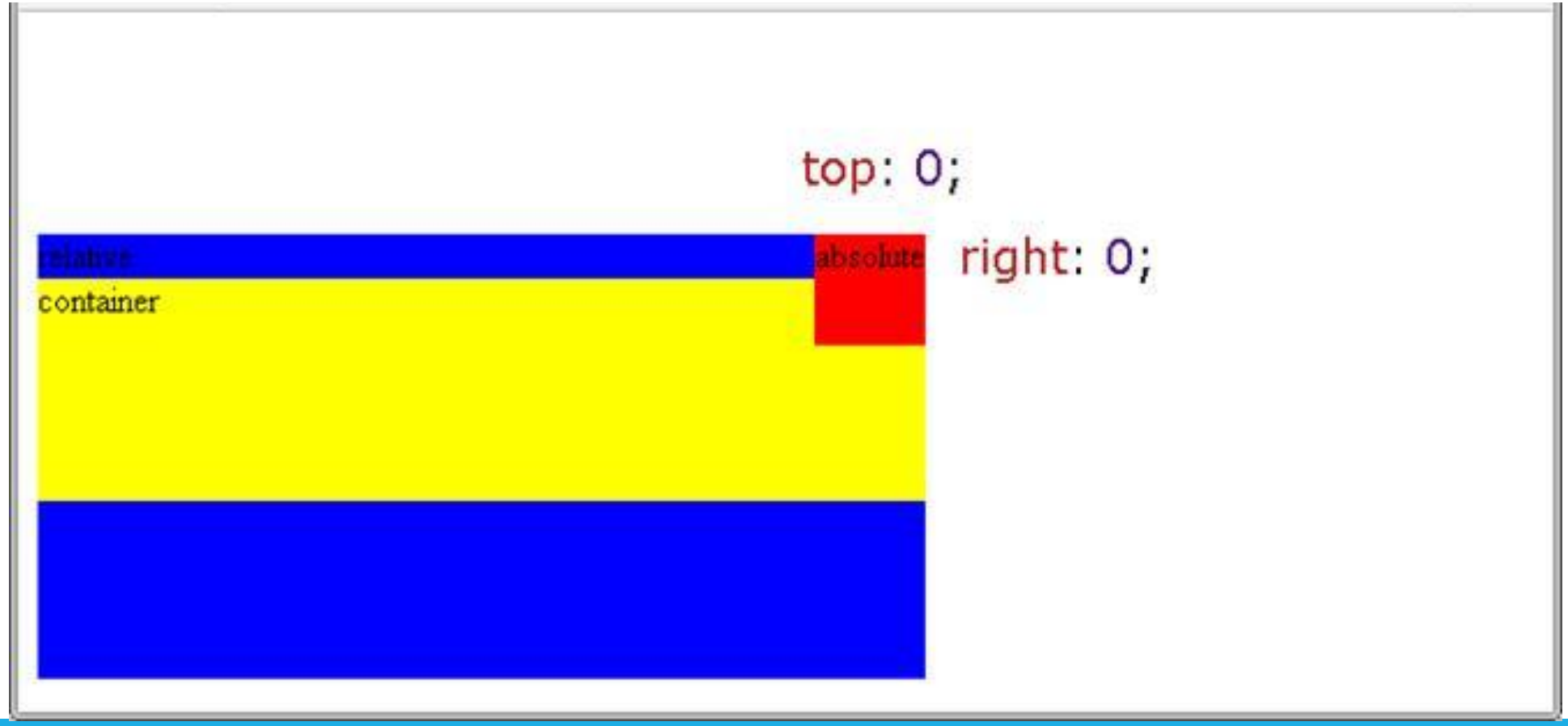

fixed

Абсолютное позиционирование относительно предка

```
.relative {position: relative;
margin-top: 100px; /* внешний отступ */
width: 400px; height: 200px;
background-color: blue; }

.container {height: 100px; /* static  по ум*/
background-color: yellow; }

.absolute {position: absolute;
top: 0;
right: 0;
width: 50px;
height: 50px;
background-color: red;    }
```

```
<div class = "relative">    relative
    <div class = "container">
        container
        <div class = "absolute">
        absolute        </div>
    </div>
</div>
```

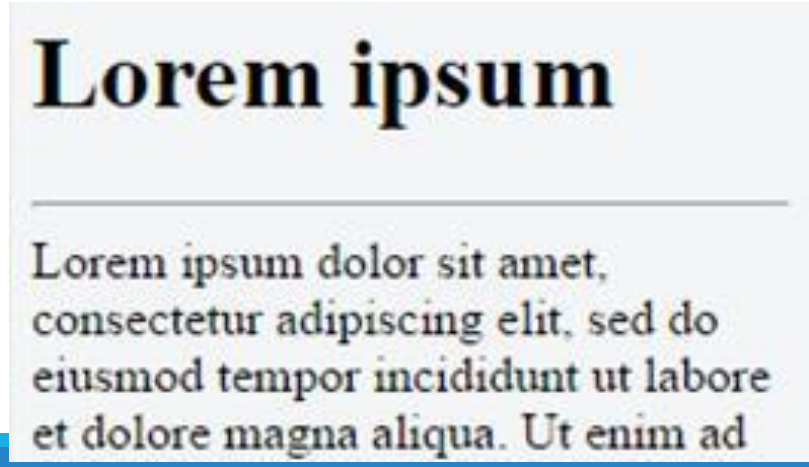
позиционируется не относительно окна браузера, не относительно родительского элемента, а относительно своего предка, который имеет позиционирование, отличное от статического



sticky – это сочетание относительного и фиксированного позиционирования. Элемент рассматривается как позиционированный относительно, пока он не пересекает определённый порог, после чего рассматривается как фиксированный. Обычно применяется для фиксации заголовка на одном месте, пока содержимое, к которому относится заголовок, прокручивается на странице. (плохо поддерживается)

```
#one {
  background-color: #f3f5f6;
  position: sticky;
  top: 0;
}
```

```
<h1 id="one">Lorem
ipsum</h1>
<hr>
Lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed do
eiusmod tempor
```



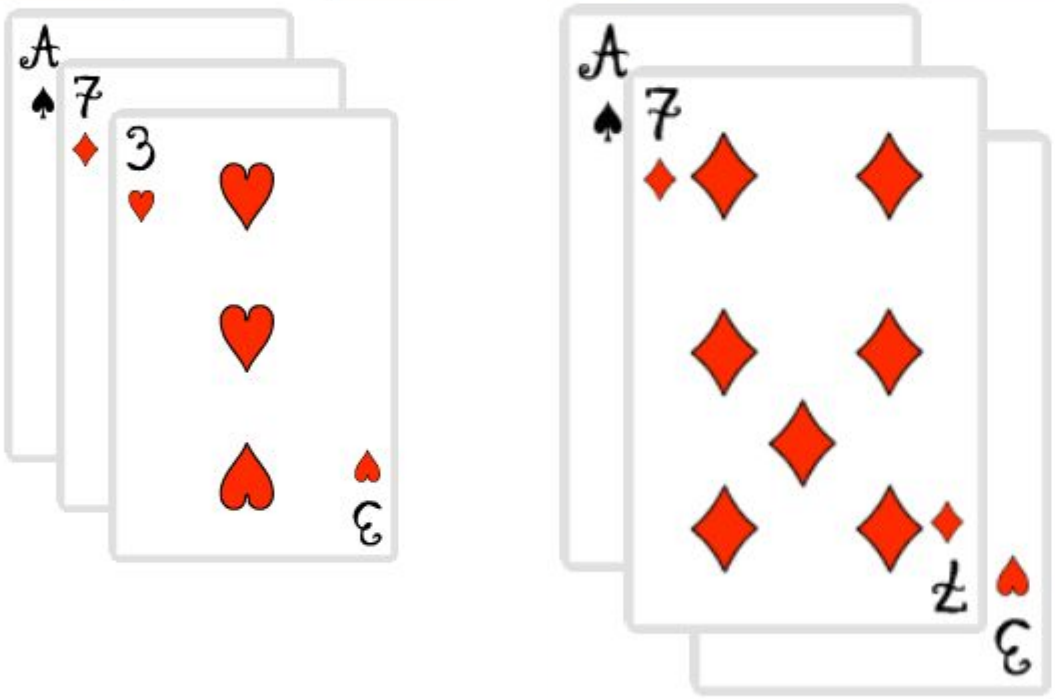
z-index — определяет порядок расположения позиционированных элементов по оси Z (порядок наложения элементов друг на друга).

Работает только с позиционируемыми элементами (position:absolute|fixed|relative).

Чем больше значение, тем выше находится элемент по сравнению с теми элементами, у которых оно меньше.

При равных z-index на переднем плане находится тот элемент, который в коде HTML описан ниже.

```
.card { position: relative; }
.three { top: 50px; left: 55px; z-index: 5; }
.seven { left: -120px; top: 25px; z-index: 2;}
.ace { left: -295px; z-index: 1; }
.card:hover { z-index: 10; }
```



при наведении курсора на карту она выходит на передний край, частично перекрывая остальные собой остальные изображения

Плавающие блоки

Плавающий блок «прикрепляется» к указанной стороне своего родителя, а остальные элементы его обтекают с других сторон.

Чтобы сделать блок плавающим, используется свойство `float` (возможные значения: `left`, `right`, `none`).

Плавающим может быть любой блок, кроме абсолютно позиционированных (`position: absolute|fixed`).

```
img {width: 30%;
/* от родительского элемента
*/}
h2 {
text-align: center;}
.floatLeft {
float: left;
padding: 5px 5px 5px 0;}
.floatRight {
float: right;
padding: 5px 0 5px 5px;
 /* внутренние отступы */}
p {
text-align:justify;/*по
ширине*/
```

```
<h2>Панды</h2>
<img src = "panda1.jpg"
class = "floatLeft" >
<p>Большая панда,
...</p>
<img src = "panda2.jpg"
class = "floatRight" >
<p>Большие ...</p>
```

Пример работы с плаваю

Панды

Большая панда, или бамбуковый медве́дь (лат. Ailuropoda melanoleuca) — вид млекопитающих из семейства медвежьих (Ursidae) со своеобразной чёрно-белой окраской шерсти, обладающих некоторыми признаками енотов. Единственный современный вид рода Ailuropus подсемейства Ailuropodinae.

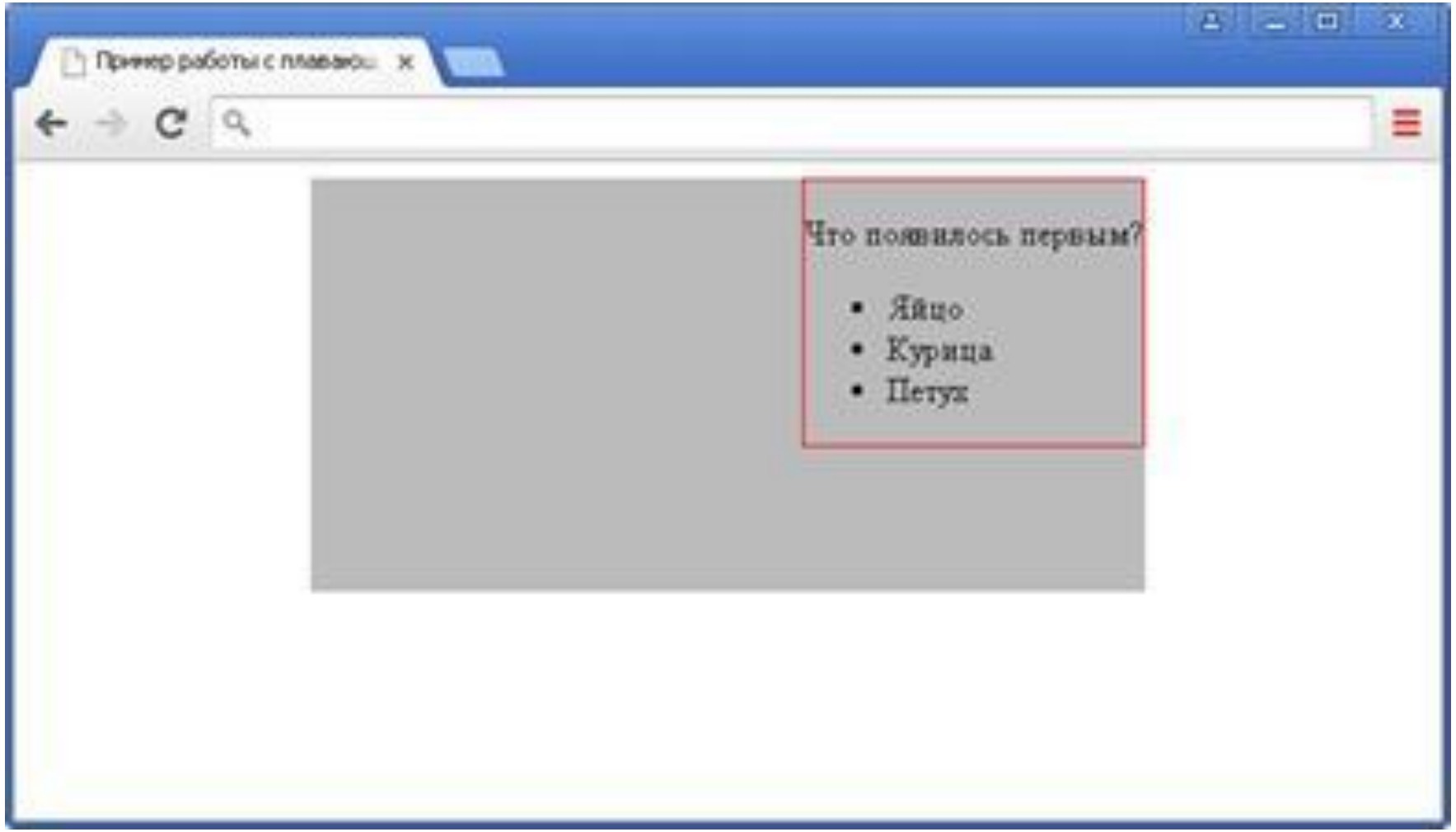
Большие панды обитают в горных регионах центрального Китая: Сычуань и Тибет. Со второй половины XX века панда стала чем-то вроде национальной эмблемы Китая. Его западное имя происходит от малой панды. Раньше его также называли пятнистым медведем (Ailuropus melanoleucus).

Особенности элемента, у которого `float` установлено в значение `left` или `right`:

- отображается как **блочный** (словно ему установили свойство `display: block`);
- по ширине **сжимается до размеров содержимого** (если у него явно не установлена ширина `width`);
- всё **остальное содержимое** страницы, идущее в HTML после элемента с `float`, обтекает его;
- если находится внутри контейнера и для контейнера заданы определенные значения ширины, то перемещение осуществляется к указанному краю внутри этого контейнера, а не по отношению к окну браузера.

```css
.container {
width: 60%;
height: 200px;
background-color: #BBB;
margin: 0 auto;
/*центрирование*/
}
.floatRight {
float: right;
border: 1px solid red;
}
```

```html
<div class = "container">
  <div class = "floatRight">
  <p>Что появилось первым?</p>
      <ul>
        <li>Яйцо</li>
        <li>Курица</li>
        <li>Петух</li>
      </ul>
  </div>
</div>
```


Пример работы с плаваю
Что появилось первым?
Яйцо
Курица
Петух

Свойство `clear` определяет, какие стороны плавающего блока не могут соседствовать с другими плавающими блоками.

Значения свойства	Где должен располагаться блок
`left`	не разрешены с левой стороны
`right`	не разрешены с правой стороны
`both`	не разрешены с обоих сторон элемента
`none`	ограничений нет (это значение по умолчанию)

```
img {width  : 100px;
height  : 100px;
float : left; }
.primer {clear  : left; }
…
<img src = "nich.jpg" alt = "nich">

<p>Съешь же ещё этих мягких
французских булок да выпей чаю.
</p>

<p class = "primer">Съешь же ещё
этих мягких французских булок
да выпей чаю.
<p>
```