

РЕЗИНОВЫЕ ИЗОБРАЖЕНИЯ



АЛЕКСАНДР БЕСПОЯСОВ / НЕТОЛОГИЯ



АЛЕКСАНДР БЕСПОЯСОВ

преподаватель курса



bespoyasov@me.com



[@bespoyasov](https://t.me/bespoyasov)

ПЛАН ЗАНЯТИЯ

1. Принцип расчета размеров элемента
2. box-sizing
3. Функция calc()
4. Отличия в использовании контентных и фоновых изображений
5. Резиновые изображения
6. Свойство background-size
7. Особенности HTML-элементов

ПРИНЦИП РАСЧЕТА РАЗМЕРОВ ЭЛЕМЕНТА

МАКЕТ С ДВУМЯ КОЛОНКАМИ

Дизайнер сделал макет фиксированной ширины в 960px . Макет содержит две колонки с текстом шириной в 250px и 700px .

```
1 .wrapper {  
2     width: 960px;  
3     margin: 0 auto;  
4 }  
5  
6 .menu {  
7     width: 250px;  
8     float: left;  
9     background-color: #018b5a;  
10 }  
11  
12 .content {  
13     width: 700px;  
14     float: right;  
15     background-color: #edc9dd;  
16 }  
17  
18 .link {  
19     color: #000000;  
20 }
```

ДВЕ КОЛОНКИ

Неплохо, но чего-то не хватает, а именно внутренних отступов.

Главная
О нас
Каталог работ
Вопросы и ответы
Контакты

Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.

ЗАДАДИМ ВНУТРЕННИЕ ОТСТУПЫ

Зададим обоим элементам `padding` по `20px` со всех сторон.

```
1 .menu {  
2     width: 250px;  
3     float: left;  
4     background-color: #018b5a;  
5     padding: 20px;  
6 }  
7  
8 .content {  
9     width: 700px;  
10    float: right;  
11    background-color: #ede9dd;  
12    padding: 20px;  
13 }
```

ВЕРСТКА СЛОМАЛАСЬ

Как видим, колонки съехали, сломав верстку.



Чтобы понять, почему так получилось, вспомним блочную модель браузера.



Блочная модель – алгоритм, по которому браузер рассчитывает размеры области, в которой находится элемент. В ней задействованы следующие свойства: ширина контента `width`, высота контента `height`, внутренние отступы `padding`, рамка `border` и внешние отступы `margin`.

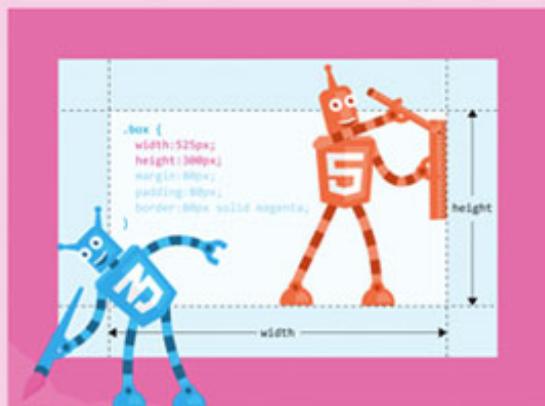
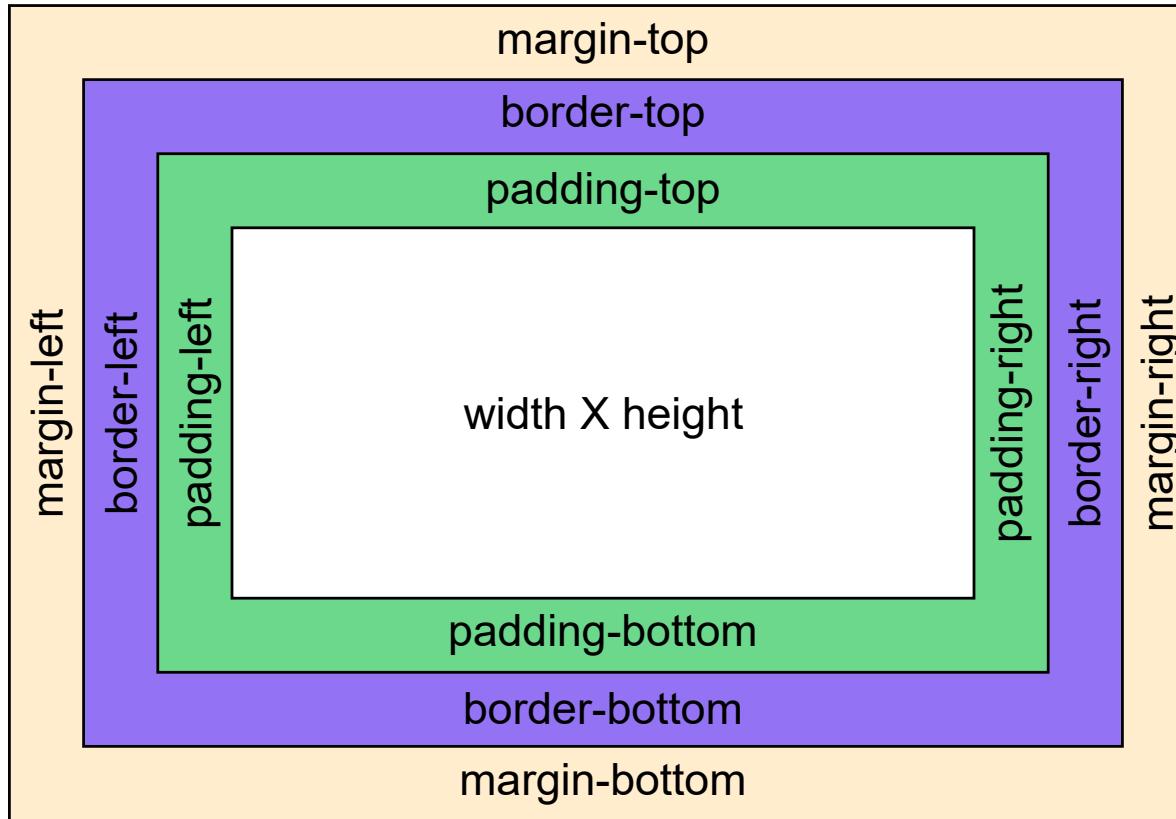


СХЕМА БЛОЧНОЙ МОДЕЛИ



РАСЧЕТ РАЗМЕРОВ ПО ГОРИЗОНТАЛИ

Чтобы определить, сколько места занимает элемент по горизонтали, браузер суммирует свойства, задействованные в модели, по следующей формуле:

```
width + padding-left + padding-right + border-left +  
border-right + margin-left + margin-right
```

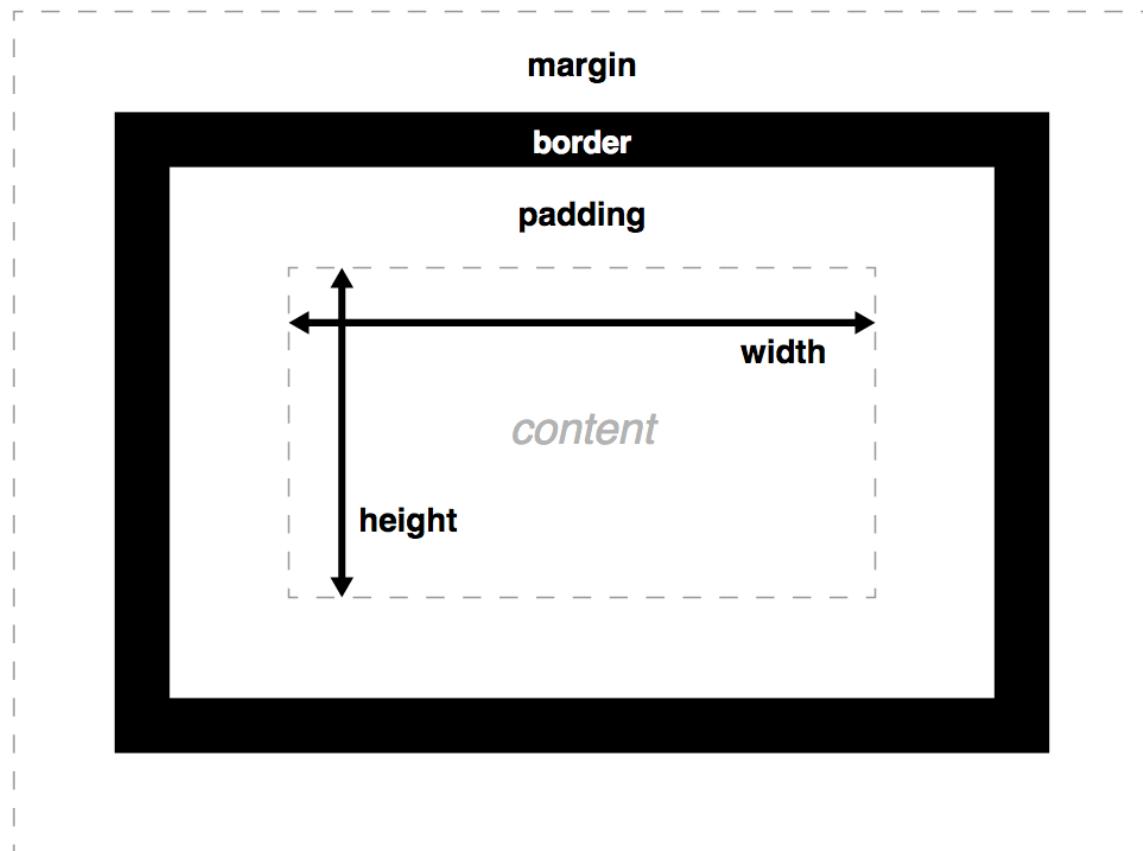
РАСЧЕТ РАЗМЕРОВ ПО ВЕРТИКАЛИ

Для расчета размеров по вертикали браузер поступает аналогичным образом:

```
height + padding-top + padding-bottom + border-top +  
border-bottom + margin-top + margin-bottom
```

ВАЖНЫЙ НЮАНС

Здесь важный нюанс состоит в том, что `width` и `height` устанавливают значение для размеров контента блока, а не для самого блока.



СЧИТАЕМ РАЗМЕРЫ КОЛОНК

Теперь вернемся к примеру и «побудем браузерами» – подставим значение свойств для колонок в блочную модель.

Левая колонка:

$$250\text{px} + 20\text{px} + 20\text{px} + 0 + 0 + 0 + 0 = 290\text{px}$$

Правая колонка:

$$700\text{px} + 20\text{px} + 20\text{px} + 0 + 0 + 0 + 0 = 740\text{px}$$

СУММА КОЛОНOK

А теперь суммируем значения:

290px + 740px = 1030px

Получили 1030px !

А у родителя – 960px , поэтому браузер перенес правый блок на вторую строчку.

ПОДГОНЯЕМ РАЗМЕРЫ

Для того, чтобы исправить это, нужно из ширины вычесть значения левого и правого отступов:

```
1 .menu {  
2     width: 210px; /* 250 - 20 - 20 */  
3     float: left;  
4     background-color: #018b5a;  
5     padding: 20px;  
6 }  
7  
8 .content {  
9     width: 660px; /* 700 - 20 - 20 */  
10    float: right;  
11    background-color: #edc9dd;  
12    padding: 20px;  
13 }
```

ПОЧИНИЛИ ВЕРСТКУ

Колонки встали на свои места!

The screenshot shows a web page with a dark green sidebar on the left containing navigation links: Главная, О нас, Каталог работ, Вопросы и ответы, and Контакты. The main content area to the right contains placeholder text (Lorem Ipsum) explaining its history and widespread use in design.

Главная
О нас
Каталог работ
Вопросы и ответы
Контакты

Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.

НОВОЕ СВОЙСТВО В CSS3

В стандарте CSS3 появилось свойство `box-sizing`, которое позволяет более удобно работать с размерами элементом.



box-sizing – свойство, позволяющее переключить алгоритм расчета размеров элементов.

Оно имеет несколько значений.

- ***content-box***
- ***border-box***



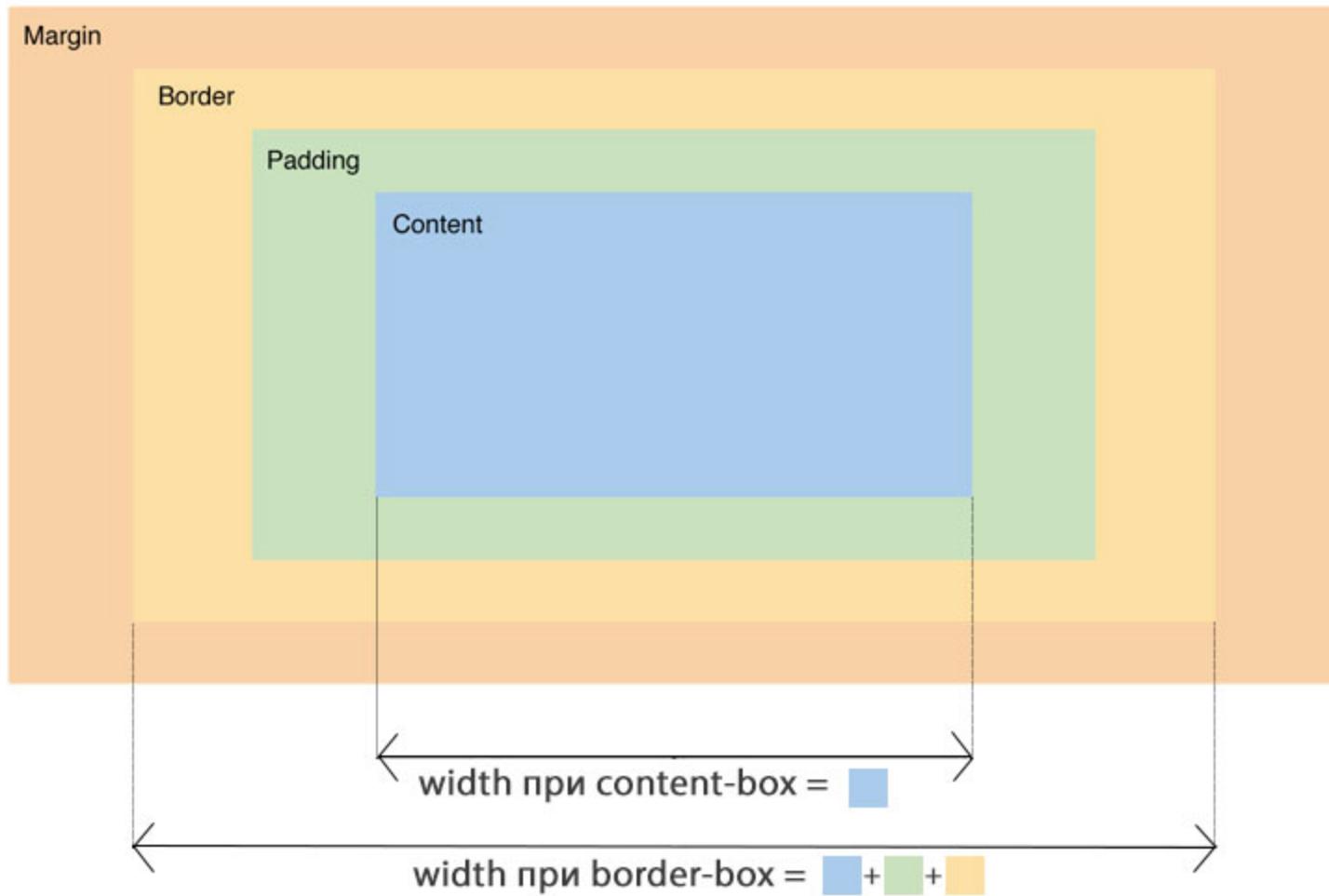
content-box

Значение `content-box` является значением по умолчанию. При таком значении браузер работает по стандартной блочной модели, которую мы рассмотрели ранее.

border-box

А вот значение `border-box` изменяет порядок расчета размеров элемента так, что значения свойств `padding` и `border` уже входят в свойства `width` и `height`.

Соответственно, `width` и `height` перестают задавать размеры контентной области и начинают задавать размеры самого блока.



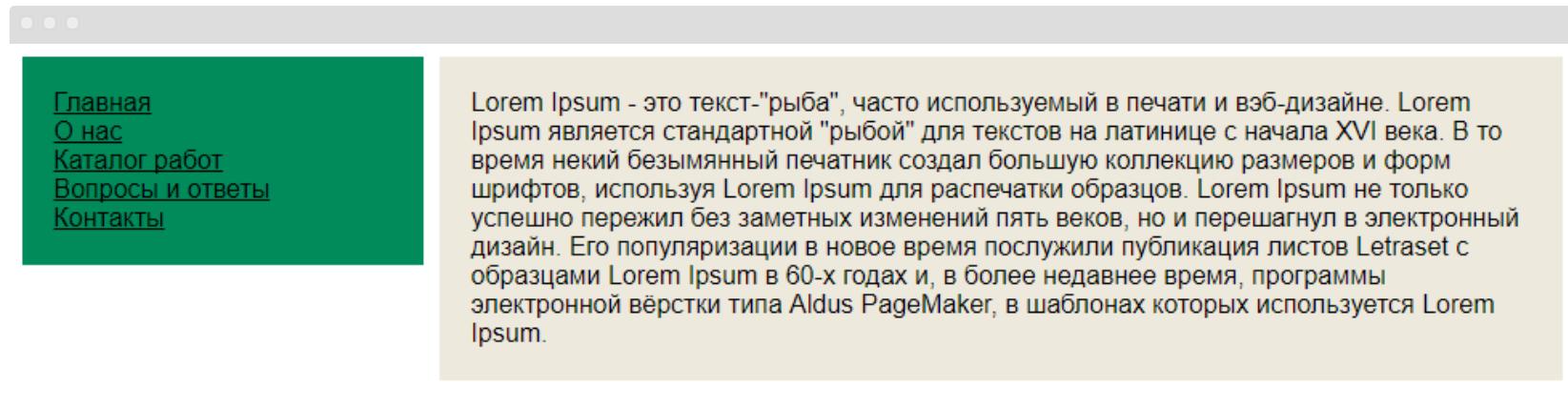
АЛЬТЕРНАТИВНОЕ РЕШЕНИЕ ЗАДАЧИ

Решим задачу с двумя колонками при помощи `box-sizing` со значением `border-box`:

```
1 .menu {  
2     width: 250px;  
3     float: left;  
4     background-color: #018b5a;  
5     padding: 20px;  
6     box-sizing: border-box;  
7 }  
8  
9 .content {  
10    width: 700px;  
11    float: right;  
12    background-color: #ede9dd;  
13    padding: 20px;  
14    box-sizing: border-box;  
15 }
```

ИЗМЕНИЛСЯ ПРИНЦИП РАСЧЕТА ШИРИНЫ

По изображению видно, что браузер включил значения `padding` в `width`, уменьшив контентную область.

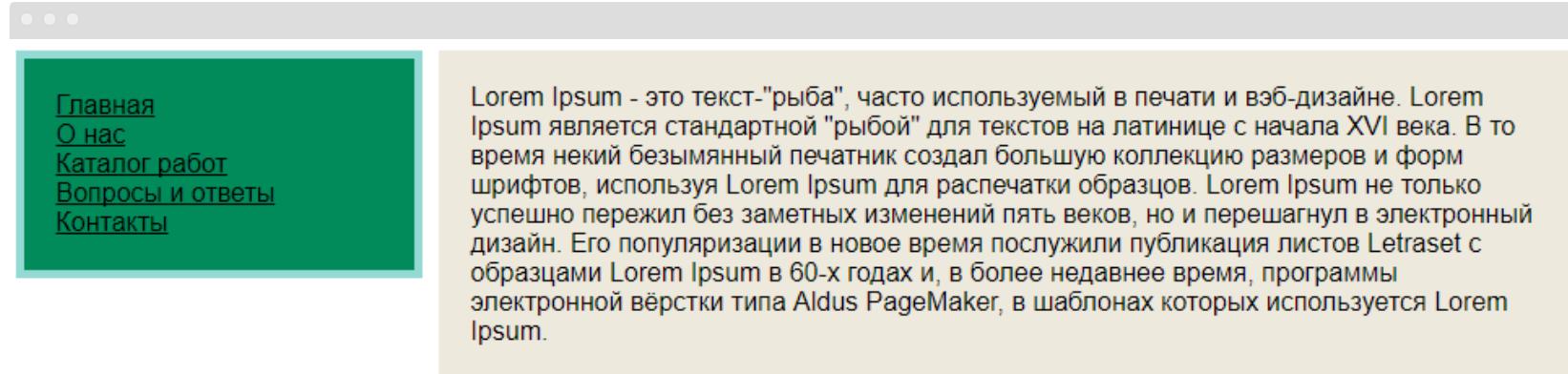


ДОБАВИМ border

Если добавить обводку блоку `.menu`, то ее ширина также будет включена в заданную ширину:

```
1 .menu {  
2     width: 250px;  
3     float: left;  
4     background-color: #018b5a;  
5     padding: 20px;  
6     box-sizing: border-box;  
7     border: 5px solid #94d9d4;  
8 }
```

ОБВОДКА ВКЛЮЧЕНА В ШИРИНУ



Lorem Ipsum - это текст-"рыба", часто используемый в печати и вэб-дизайне. Lorem Ipsum является стандартной "рыбой" для текстов на латинице с начала XVI века. В то время некий безымянный печатник создал большую коллекцию размеров и форм шрифтов, используя Lorem Ipsum для распечатки образцов. Lorem Ipsum не только успешно пережил без заметных изменений пять веков, но и перешагнул в электронный дизайн. Его популяризации в новое время послужили публикация листов Letraset с образцами Lorem Ipsum в 60-х годах и, в более недавнее время, программы электронной вёрстки типа Aldus PageMaker, в шаблонах которых используется Lorem Ipsum.

функция calc()

ДВЕ КОЛОНКИ НА ВЕСЬ ЭКРАН

Допустим, у нас есть резиновый макет с двумя колонками, занимающими всю ширину окна браузера и отступом в 20px.



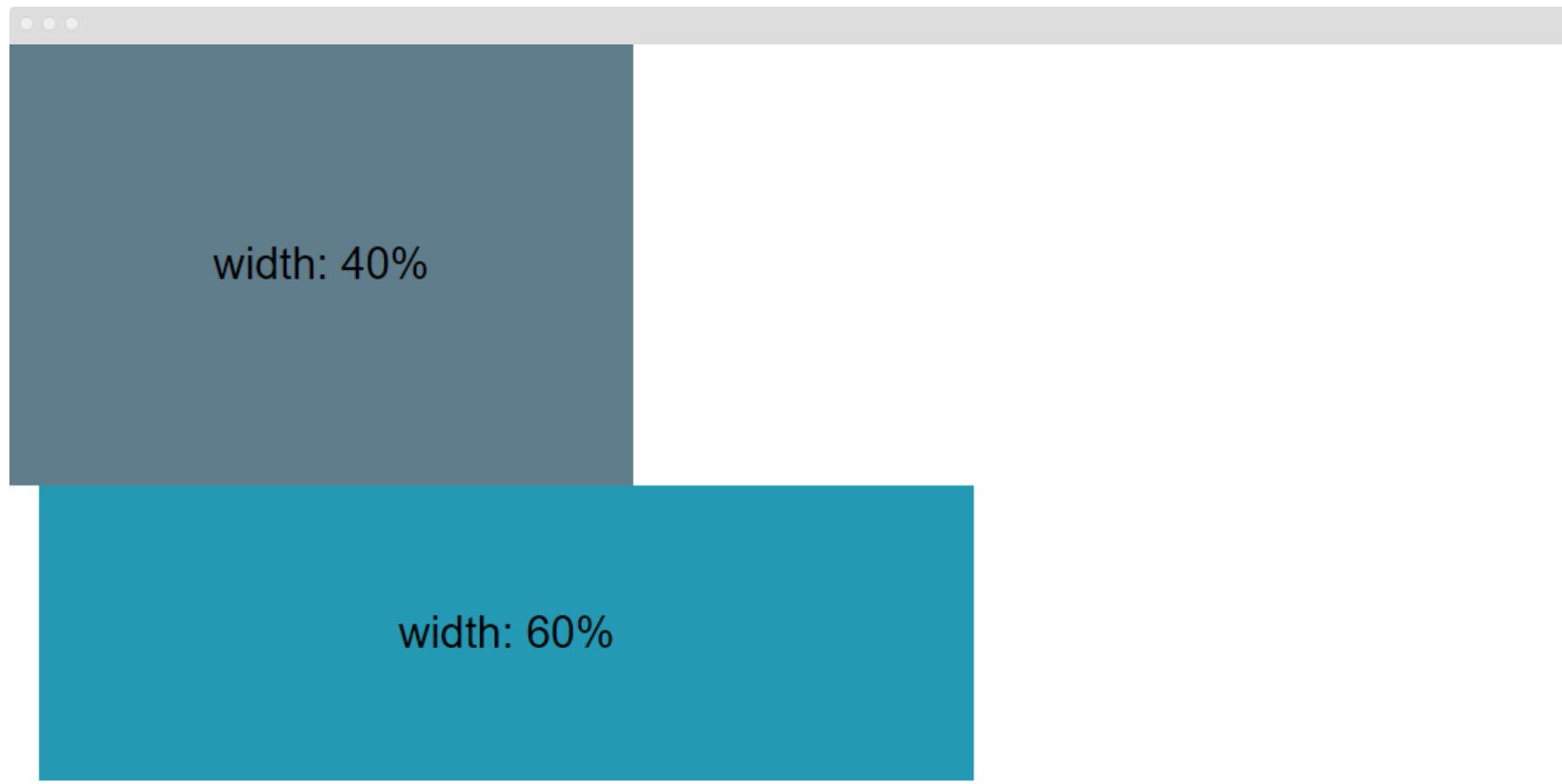
НАПИШЕМ СТИЛИ

Следующий код выглядит логично при заданных условиях:

```
1 .sidebar {  
2     height: 300px;  
3     width: 40%;  
4     background-color: #607D8B;  
5     float: left;  
6 }  
7  
8 .content {  
9     height: 200px;  
10    width: 60%;  
11    background-color: #2499b3;  
12    margin-left: 20px;  
13    float: left;  
14 }
```

ВТОРАЯ КОЛОНКА НЕ ПОМЕЩАЕТСЯ

В стилях не учитывается то, что обе колонки занимают в сумме 100% ширины окна браузера, а значит, отступ в 20px неминуемо «столкнет» правую колонку вниз.



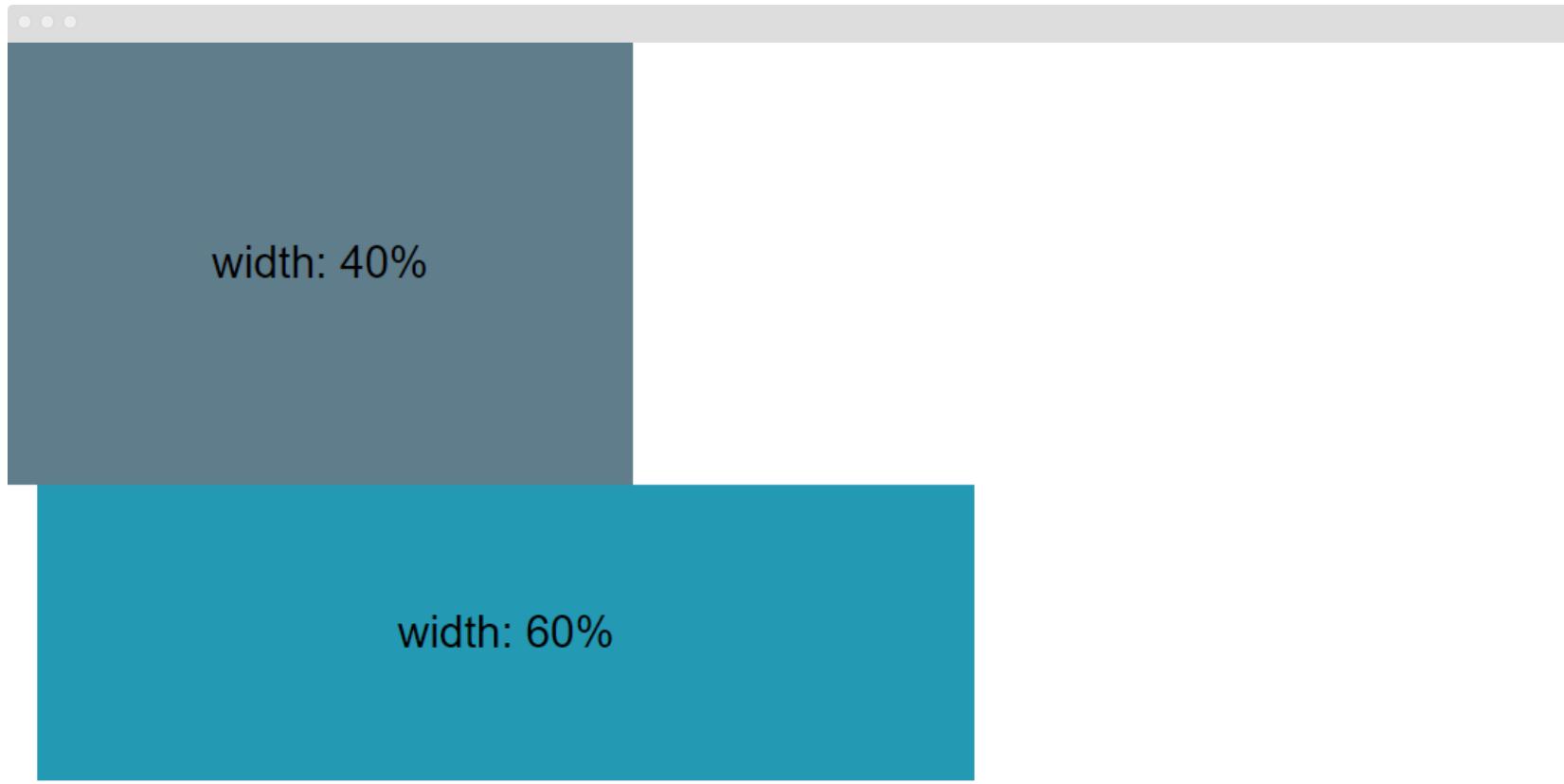
ДОБАВИМ box-sizing

Что произойдет, если блоку `.content` задать
`box-sizing: border-box`?

```
1 .sidebar {  
2     height: 300px;  
3     width: 40%;  
4     background-color: #607d8b;  
5     float: left;  
6 }  
7  
8 .content {  
9     height: 200px;  
10    width: 60%;  
11    background-color: #2499b3;  
12    margin-left: 20px;  
13    float: left;  
14    box-sizing: border-box;  
15 }
```

ПРОБЛЕМА НЕ РЕШЕНА

Ситуация совершенно не изменилась, ведь элемент `.content` имеет `margin-left`, а не `padding` или `border`.



НУЖНО ВЫЧЕСТЬ ЛЕВЫЙ ОТСТУП

В данном случае логичным кажется уменьшить ширину блока `.content` на значение `margin-left`, то есть на `20px`.

Именно здесь к нам на помощь приходит функция `calc()`.



Функция `calc()` используется для указания вычисляемого значения свойств, которые в качестве значений используют размер, угол, время или число.

Если значение не может быть вычислено, оно игнорируется.



МАТЕМАТИЧЕСКИЕ ОПЕРАЦИИ

В `calc()` можно делать следующие вычисления, используя выражения:

- Сложение `width: calc(20px + 20px);`
- Вычитание `padding: calc(10% - 10px);`
- Умножение `height: calc(20%*2);`
- Деление `width: calc(100%/3)` (*На ноль делить запрещено*).

Важно: знаки сложения и вычитания должны отделяться пробелом.

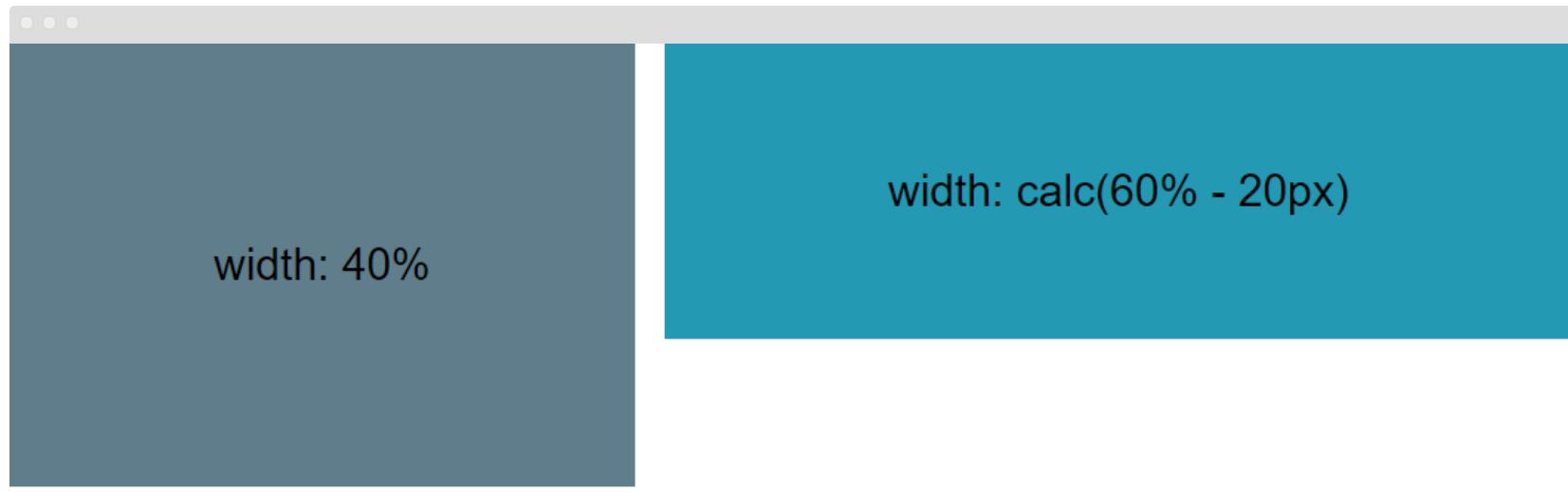
ВЫЧИТАЕМ ЛЕВЫЙ ОТСТУП

Теперь мы используем функцию `calc()` для задания ширины блока `.content`, тем самым установив его ширину таким образом, чтобы она равнялась 60% от ширины экрана браузера минус `20px`.

```
1 .sidebar {  
2     height: 300px;  
3     width: 40%;  
4     background-color: #607d8b;  
5     float: left;  
6 }  
7  
8 .content {  
9     height: 200px;  
10    width: calc(60% - 20px);  
11    background-color: #2499b3;  
12    margin-left: 20px;  
13    float: left;  
14 }
```

БЛОКИ ВСТАЛИ В РЯД

Как видно на иллюстрации, теперь блок `.content` не сползает вниз.



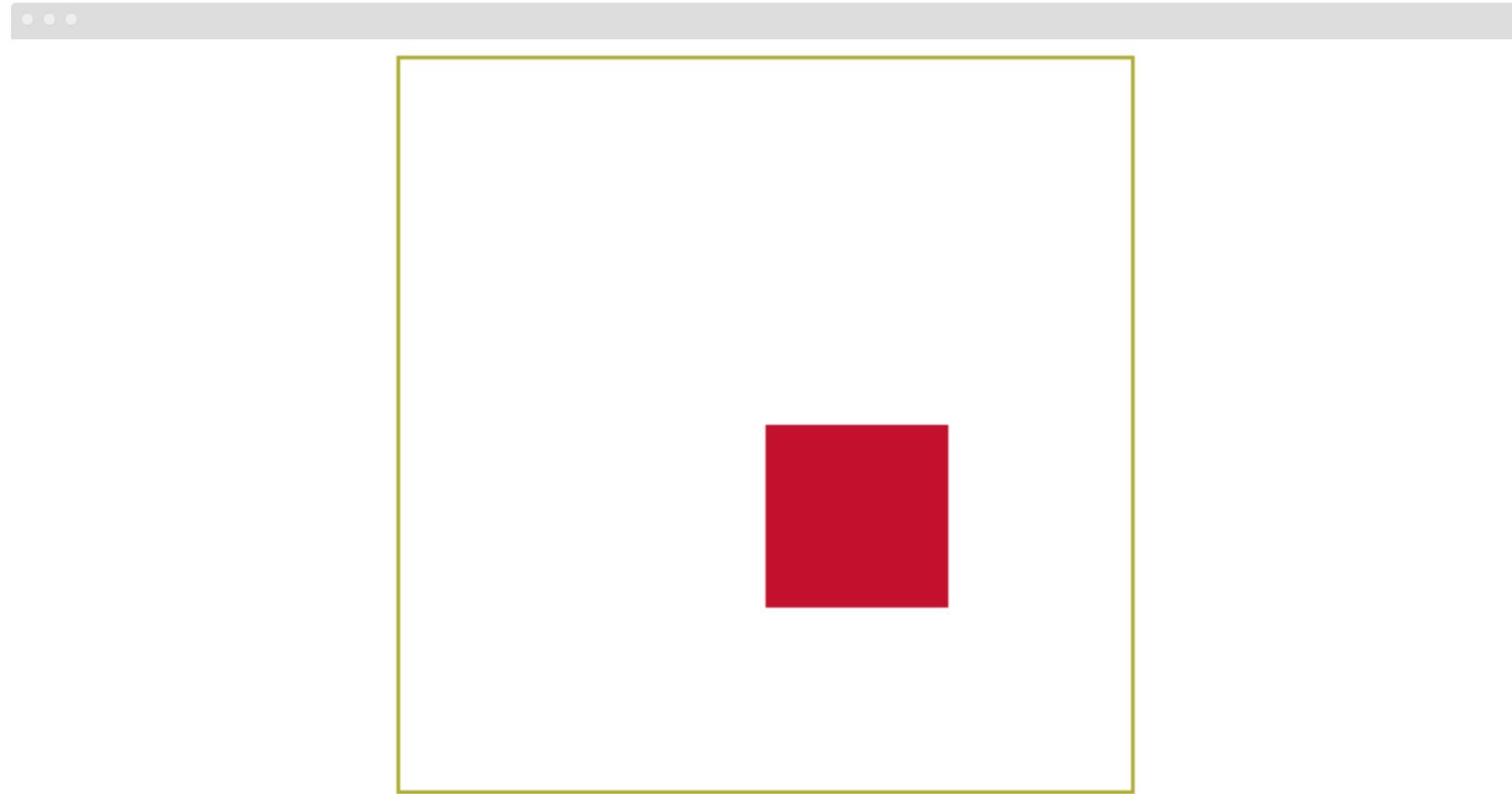
СТРОГО ПО ЦЕНТРУ

Поставлена задача расположить элемент по центру вертикально и горизонтально относительно родительского контейнера:

```
1 .wrapper {  
2     position: relative;  
3     width: 400px;  
4     height: 400px;  
5     margin: 0 auto;  
6     border: 2px solid #afac34;  
7     box-sizing: border-box;  
8 }  
9  
10 .popup {  
11     width: 100px;  
12     height: 100px;  
13     position: absolute;  
14     left: 50%;  
15     top: 50%;  
16     background-color: #c3102d;  
17 }
```

НЕ ПО ЦЕНТРУ

Почему элемент с классом `.popUp` не позиционируется по центру?



ПРИЧИНЫ СМЕЩЕНИЯ

Это произошло из-за того, что значения `top: 50%` и `left: 50%` рассчитываются, основываясь на ширине и высоте родительского контейнера, и в данном случае равны `200px` каждый.

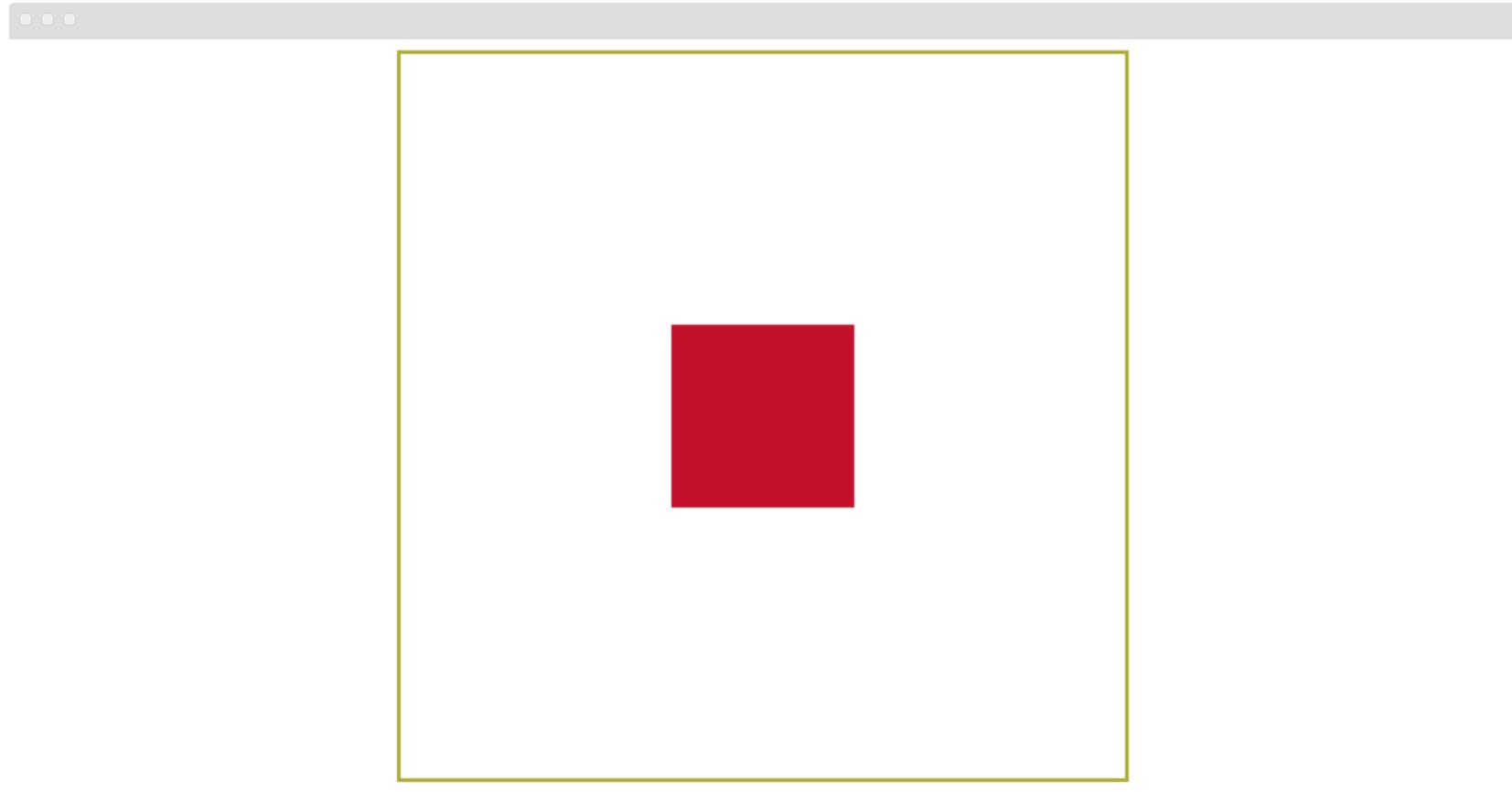
ИСПОЛЬЗУЕМ calc()

Чтобы поставить блок `.popup` строго по центру вертикали и горизонтали, нужно взять половину ширины/высоты родительского контейнера и отнять половину ширины/высоты самого блока `.popup`.

```
1 .wrapper {  
2     position: relative;  
3     width: 400px;  
4     height: 400px;  
5     margin: 0 auto;  
6     border: 2px solid #afac34;  
7     box-sizing: border-box;  
8 }  
9  
10 .popup {  
11     width: 100px;  
12     height: 100px;  
13     position: absolute;  
14     left: calc(50% - 100px/2);  
15     top: calc(50% - 100px/2);  
16     background-color: #c3102d;  
17 }
```

ТЕПЕРЬ ПО ЦЕНТРУ

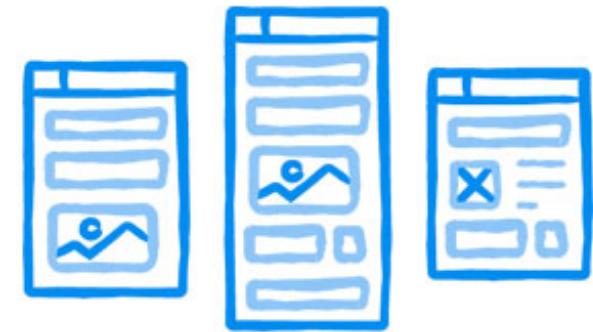
Теперь `.popUp` позиционируется строго по центру.



КОНТЕНТНЫЕ И ФОНОВЫЕ ИЗОБРАЖЕНИЯ



*Контентные изображения – термин происходит от английского слова **content**, что на русский переводится как содержимое.*



Благодаря таким изображениям мы можем донести до пользователя нашего сайта полезную информацию.

Для добавления изображения на страницу используется HTML-тег ``.



Фоновые (декоративные) изображения –
используются в декоративных целях и не несут
для пользователя полезной информации.

*То есть, если скрыть такое изображение, то
пользователь все равно будет в состоянии
воспринимать информацию на сайте.*

ИЗОБРАЖЕНИЕ ТОВАРА

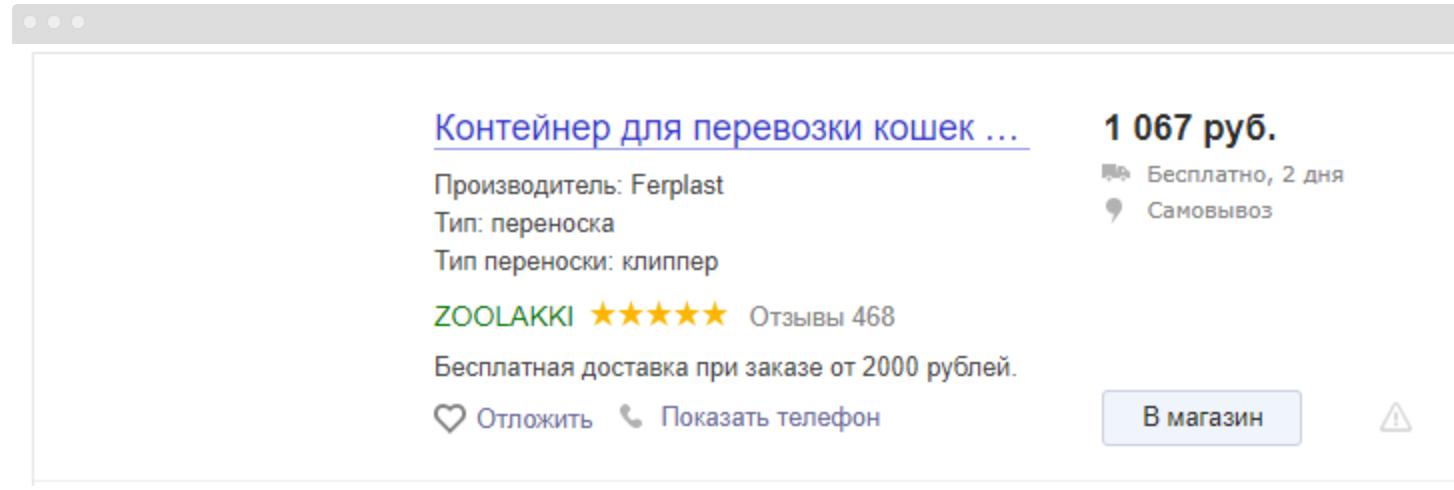
На изображении показана типичная карточка товара в интернет-магазине. Красным прямоугольником выделено изображение товара.

Это изображение является контентным, потому что, если его скрыть, то пользователь не сможет получить полную информацию о товаре.

The screenshot shows a product card for a pet carrier. On the left, there is a small image of a white and green plastic carrier with a red border around it. To the right of the image, the product title is partially visible: "Контейнер для перевозки кошек ...". Below the title, the manufacturer is listed as "Производитель: Ferplast", the type as "Тип: переноска", and the carrier type as "Тип переноски: клиппер". Underneath this information, the brand name "ZOOLAKKI" is followed by five yellow stars and the number "Отзывы 468". A note about free shipping for orders over 2000 rubles is present. At the bottom, there are buttons for "Отложить" (Bookmark) and "Показать телефон" (Show phone), a "В магазин" (To store) button, and a warning icon.

УБИРАЕМ КАРТИНКУ – ТЕРЯЕМ СМЫСЛ

Согласитесь, информативность этого блока пострадала.



АТРИБУТ `alt`

Иногда браузер не может загрузить изображение. Причины этого могут быть разными, будь то неправильно пришедший ответ от сервера или банальная опечатка в пути к файлу. От таких случаев верстальщик обязан подстраховываться и оставлять некое описание к изображению.

У тега `` есть полезный атрибут `alt`, который задает альтернативный текст к изображению, который отображается, когда с загрузкой изображения что-то пошло не так.

ПИШЕМ ЧТО НА КАРТИНКЕ

Изображение не загрузилось, но осталась информация о нем.

The screenshot shows a product listing for a 'Контейнер для перевозки кошек Ferplast ATLAS 10'. The product image is missing, but the title, price, and other details are visible. The title is 'Контейнер для перевозки кошек ...' (Container for transporting cats ...). The price is '1 067 руб.' (1,067 rubles). The producer is 'Ferplast'. The type is 'переноска' (carrier) and the transport type is 'клиппер' (clippie). The seller is 'ZOOLAKKI' with a 5-star rating and 468 reviews. Free delivery is offered for orders over 2000 rubles. There are buttons for 'Отложить' (Put on hold), 'Показать телефон' (Show phone), 'В магазин' (To store), and a warning icon.

Контейнер для перевозки кошек Ferplast ATLAS 10

Контейнер для перевозки кошек ...

1 067 руб.

Производитель: Ferplast

Тип: переноска

Тип переноски: клиппер

ZOOLAKKI ★★★★☆ Отзывы 468

Бесплатная доставка при заказе от 2000 рублей.

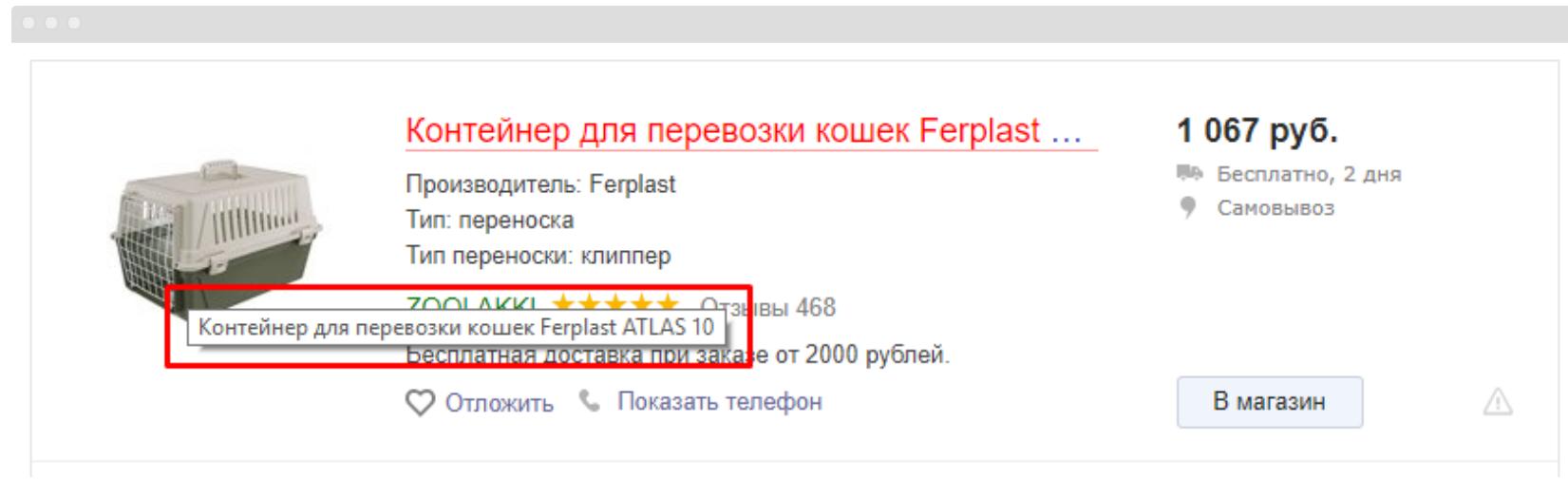
Отложить Показать телефон В магазин

ХОРОШО ДЛЯ ДОСТУПНОСТИ

Альтернативный текст помогает людям с ограниченными возможностями пользоваться вашим сайтом. Программы чтения с экрана зачитывают его вслух, и человек может представить изображение.

АТРИБУТ `title`

Еще один полезный, но не обязательный атрибут тега `` – `title`. Он позволяет отобразить текстовое описание при наведении курсора на изображение.



БЕЗ ИКОНОК СМЫСЛ НЕ ТЕРЯЕТСЯ

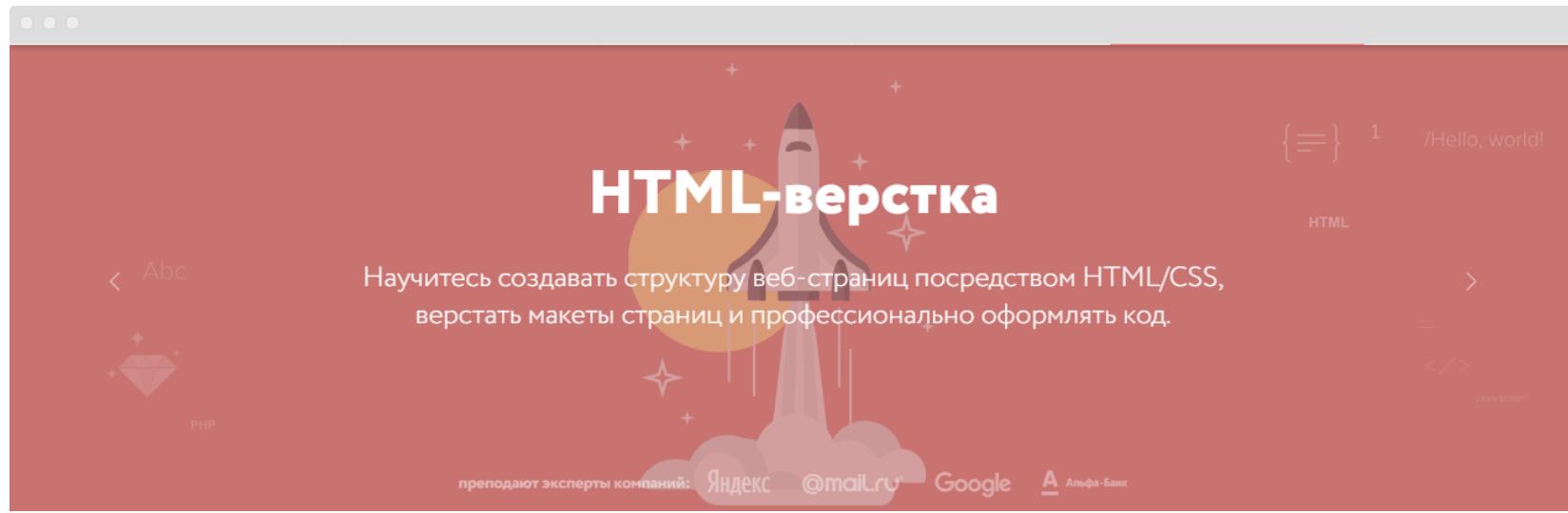
На изображении иконки в меню имеют декоративный характер. Если вдруг они пропадут, информация не потерянется, и пользователь всё равно сможет понять, какой раздел меню ему нужен.

Для реализации лучше всего использовать `background-image`.



ФОН ДЛЯ ТЕКСТА

Фоновые изображения могут использоваться в качестве подложки для информационного блока, вот, например, слайдер на сайте Нетологии:



ЗАДАЧА НА СМЕКАЛКУ

Что вы видите на этом скриншоте?



БЕЛЫЙ ФОН

Это была не шутка. На скриншоте был текст на белом фоне. Просто не загрузилась фоновая картинка.

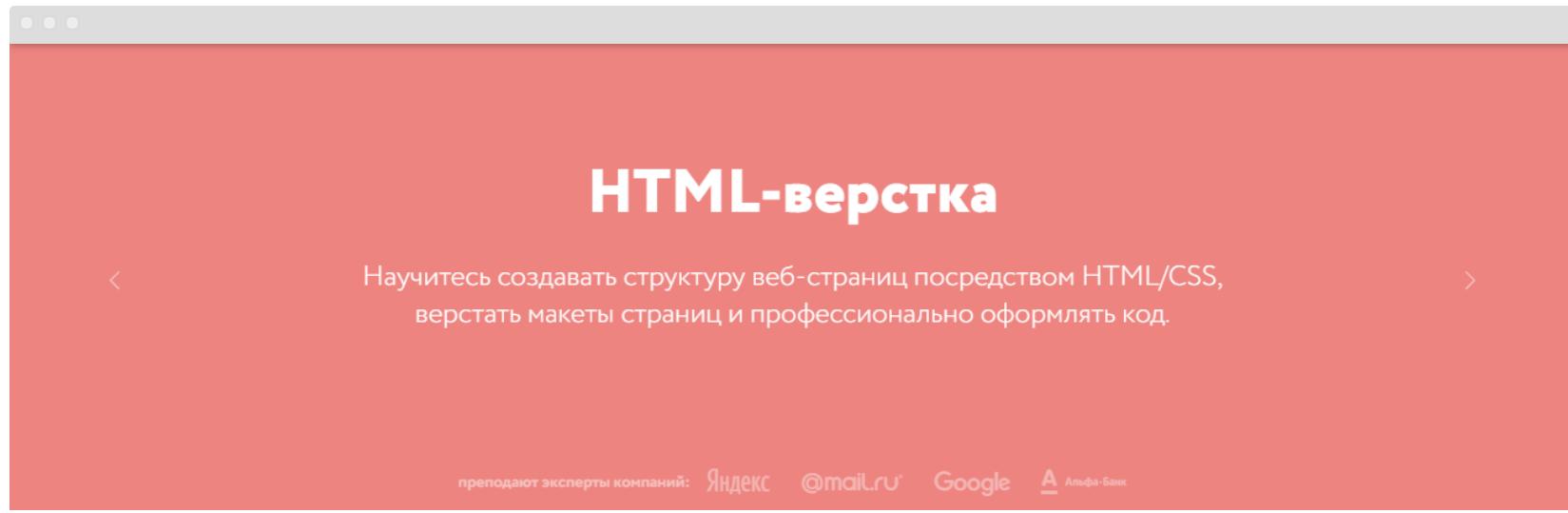
ФОНОВЫЙ ЦВЕТ

Чтобы избежать потерю читаемости всего информационного блока в случае ошибки при загрузке изображения, для фоновых изображений-подложек рекомендуется указывать фоновый цвет с помощью свойства `background-color`.

```
1 | .slider {  
2 |   background-color: #ed8480;  
3 | }
```

СМЫСЛ СОХРАНЯЕТСЯ

Как видно, информация воспринимается все так же хорошо, просто без декоративных элементов.



СКРЫВАЕМ БЛОК С ФОНОМ

В верстке бывают случаи, когда нужно скрыть блок с фоновым изображением, например, он будет появляться по клику или по наведению мыши.

Казалось бы, достаточно задать этому блоку свойство `display: none`, однако не все так просто.

МАСШТАБНЫЙ ЭКСПЕРИМЕНТ

Рассмотрим задачу с точки зрения оптимизации загружаемых браузером ресурсов.

Проверять загрузку изображения мы будем во вкладке **Network** в инструментах разработчика.

Протестируем загрузку изображений:

- заданных с помощью `background-image` на элемент;
- `background-image` на псевдоэлемент `:before`;
- заданного с помощью тега `img`.

РАЗМЕТКА ДЛЯ ТЕСТА

В разметке мы разместим изображение при помощи тега ``:

```
1 <div class="block-bg block-hidden"></div>
2 <div class="block-before block-hidden"></div>
3 
```

СТИЛИ ДЛЯ ТЕСТА

В стилях задаем фоновые картинки для элемента с классом `.block-bg` и для псевдоэлемента `::before` для элемента с классом `.block-before`:

```
1 .block-bg {  
2     background-image: url("background_image.jpg");  
3     background-repeat: no-repeat;  
4 }  
5  
6 .block-before::before {  
7     content: "";  
8     display: block;  
9     background-image: url("before_image.jpg");  
10    background-repeat: no-repeat;  
11    width: 100%;  
12    height: 100%;  
13 }
```

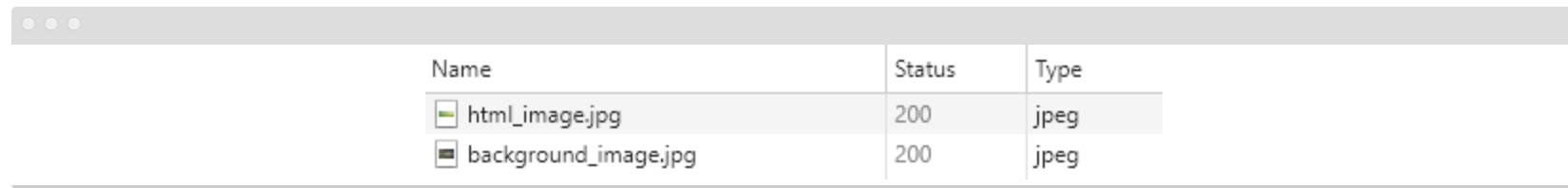
ПРОБУЕМ `display: none`

В первую очередь рассмотрим влияние на загрузку свойства `display: none`:

```
1 .block-hidden {  
2   display: none;  
3 }
```

display: none В GOOGLE CHROME

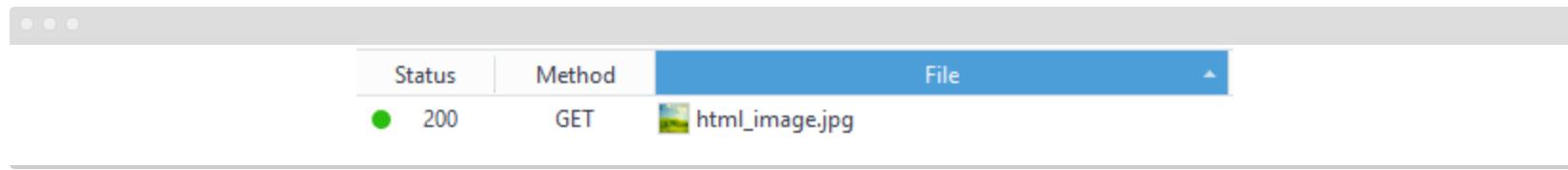
Загрузилось только изображение, заданное тегом `img` и изображение, заданное свойством `background-image`, установленном на элемент.



Name	Status	Type
html_image.jpg	200	jpeg
background_image.jpg	200	jpeg

display: none В MOZILLA FIREFOX

В браузере Firefox загрузилось только одно из трех изображение – заданное с помощью тега `img`.



display: none В IE 11

В Internet Explorer, как и в Chrome, загрузилось только два изображения.



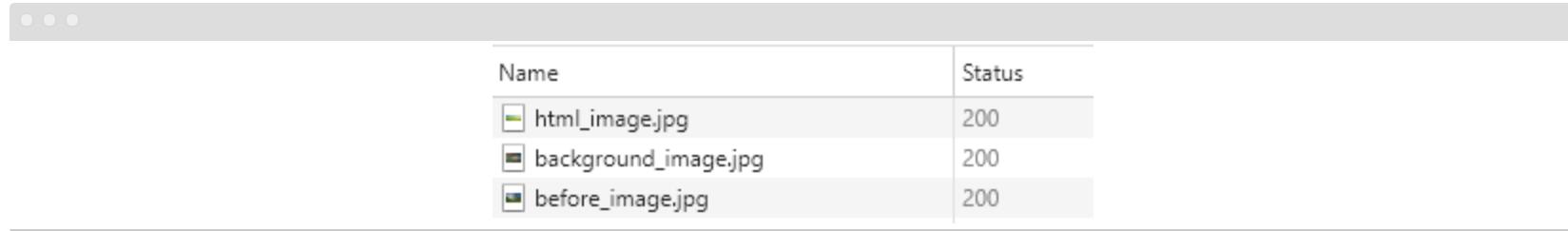
The screenshot shows the Network tab of the F12 developer tools in Internet Explorer. It displays a table with four columns: Name / Path, Protocol, Method, and Result / Description. There are two rows in the table.

Name / Path	Protocol	Method	Result / Description
html_image.jpg https://image.ibb.co/hEyZGa/	HTTP/2	GET	200
background_image.jpg https://image.ibb.co/e6hGUv/	HTTP/2	GET	200

ПРОБУЕМ opacity: 0

```
1 | .block-hidden {  
2 |   opacity: 0;  
3 | }
```

opacity: 0 GOOGLE CHROME



A screenshot of a browser developer tools Network tab. The table shows three entries:

Name	Status
html_image.jpg	200
background_image.jpg	200
before_image.jpg	200

opacity: 0 MOZILLA FIREFOX

Status	Method	File
● 200	GET	background_image.jpg
● 200	GET	before_image.jpg
● 200	GET	html_image.jpg

opacity: 0 IE 11



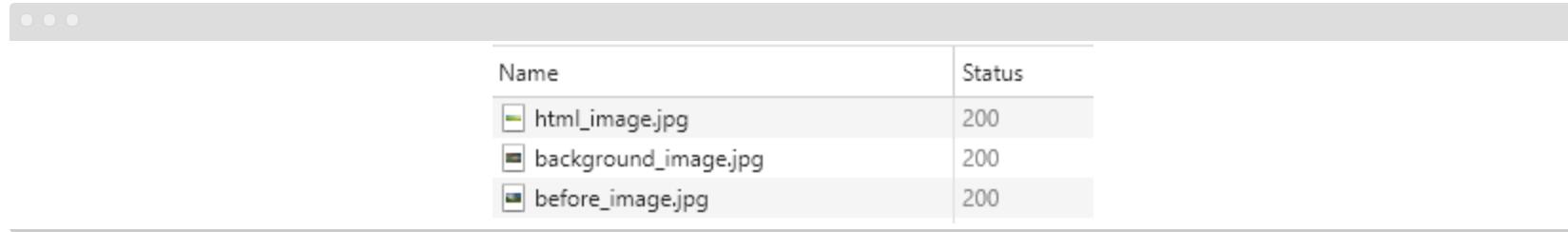
Request	Method	Status
html_image.jpg https://image.ibb.co/hEyZGa/	HTTP/2 GET	200
before_image.jpg https://image.ibb.co/gR8C2F/	HTTP/2 GET	200
background_image.jpg https://image.ibb.co/e6hGUv/	HTTP/2 GET	200

Во всех трех браузерах загружается все три изображения.

ТЕСТИРУЕМ `visibility: hidden`

```
1 | .block-hidden {  
2 |   visibility: hidden;  
3 | }
```

visibility: hidden CHROME



Name	Status
html_image.jpg	200
background_image.jpg	200
before_image.jpg	200

visibility: hidden FIREFOX

Status	Method	File
● 200	GET	background_image.jpg
● 200	GET	before_image.jpg
● 200	GET	html_image.jpg

visibility: hidden IE 11



Request	Method	Status
html_image.jpg https://image.ibb.co/hEyZGa/	HTTP/2 GET	200
before_image.jpg https://image.ibb.co/gR8C2F/	HTTP/2 GET	200
background_image.jpg https://image.ibb.co/e6hGUv/	HTTP/2 GET	200

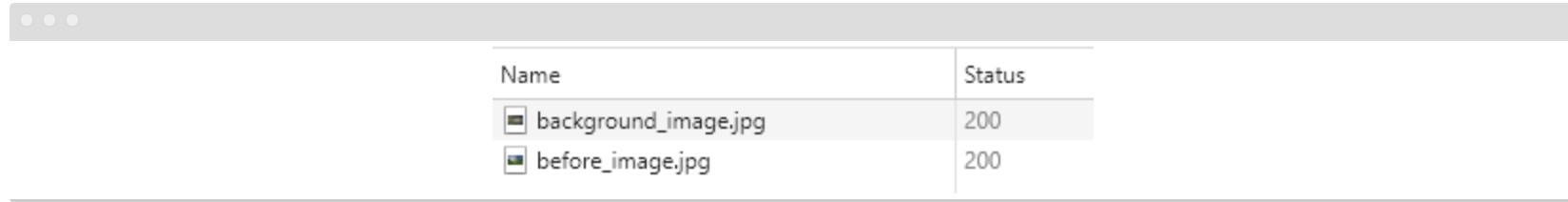
Та же ситуация, загрузились все три изображения.

background-size: 0 0

Попробуем применить это свойство к фоновым изображениям.

```
1 .block-hidden {  
2     background-size: 0 0;  
3 }  
4  
5 .block-hidden::before {  
6     background-size: 0 0;  
7 }
```

background-size: 0 0 CHROME



Name	Status
background_image.jpg	200
before_image.jpg	200

background-size: 0 0 FIREFOX

Status	Method	File
200	GET	 background_image.jpg
200	GET	 before_image.jpg

background-size: 0 0 IE 11

Name / Path	Protocol	Method	Result / Description
before_image.jpg https://image.ibb.co/gR8C2F/	HTTPS	GET	200
background_image.jpg https://image.ibb.co/e6hGUv/	HTTPS	GET	200

Свойство `background-size: 0 0` не предотвращает загрузку фоновых изображений во всех браузерах.

background: none

Есть универсальный и работающий во всех браузерах способ не загружать фоновое изображение – свойство `background: none`:

```
1 .block-hidden {  
2     background: none;  
3 }  
4  
5 .block-hidden::before {  
6     background: none;  
7 }
```

Рекомендуется скрывать фоновые изображения именно так, чтобы не загружать лишних ресурсов.

С КОНТЕНТНЫМ ИЗОБРАЖЕНИЕМ НЕ РАБОТАЕТ

В случае с контентными изображениями браузеры единогласно продолжают загружать их, несмотря на свойства `display: none` и `visibility: hidden`.

РЕЗИНОВЫЕ ИЗОБРАЖЕНИЯ



Резиновыми называют изображения, размеры которых зависят от размеров родительского блока.



ДЕЛАЕМ КАРТИНКУ РЕЗИНОВОЙ

Сделать резиновое изображение достаточно просто, нужно всего лишь задать тегу `img` свойство `max-width: 100%`.

```
1 .parent {  
2     width: 90%;  
3     height: 100px;  
4 }  
5  
6 img.child {  
7     max-width: 100%;  
8 }
```

РАБОТАЕТ И С ВЫСОТОЙ

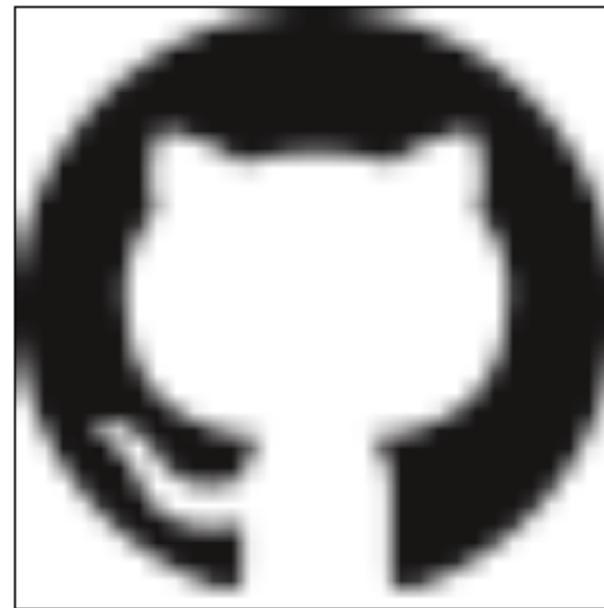
Либо `max-height: 100%`, если нужно адаптировать высоту изображения.

```
1 .parent {  
2     width: 150px;  
3     height: 100%;  
4 }  
5  
6 img.child {  
7     max-height: 100%;  
8 }
```

Чаще все же изображение адаптируют по ширине.

УХУДШЕНИЕ КАЧЕСТВА

Не рекомендуется задать изображению `width: 100%`, так как в этом случае, если размер изображения меньше родительского блока, то изображение займет всю ширину родительского блока, тем самым потеряв качество при увеличении.



БОЛЬШАЯ КАРТИНКА

У нас есть резиновый одноколоночный макет с фиксированной шириной в 960 пикселей. Под шапкой находится изображение, оригинальные размеры которого превышают ширину контейнера.

РАЗМЕТКА МАКЕТА

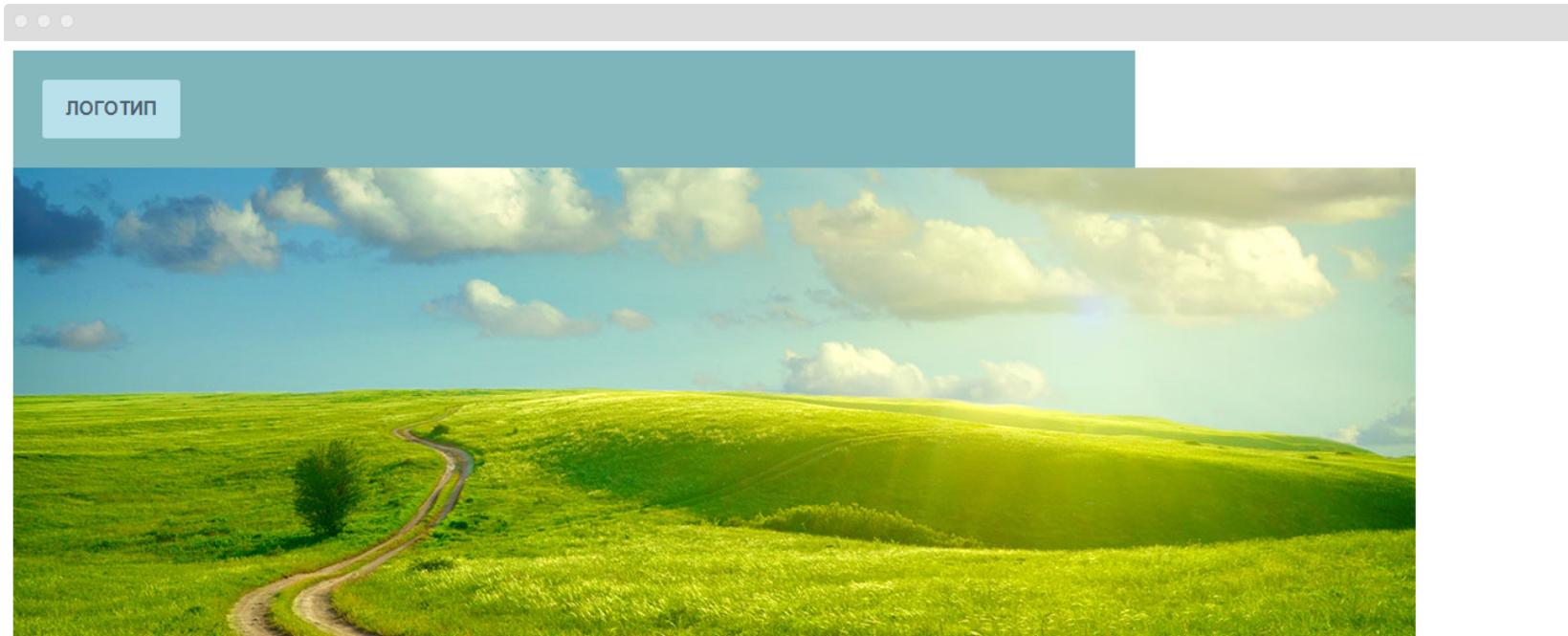
```
1 <div class="wrapper">
2   <div class="header">
3     <div class="logo">Логотип</div>
4   </div>
5   <div class="hero">
6     
7   </div>
8 </div>
```

СТИЛИ ПРИМЕРА

```
1 .wrapper {  
2     max-width: 960px;  
3     margin: 0 auto;  
4 }  
5  
6 .hero {  
7     text-align: center;  
8 }
```

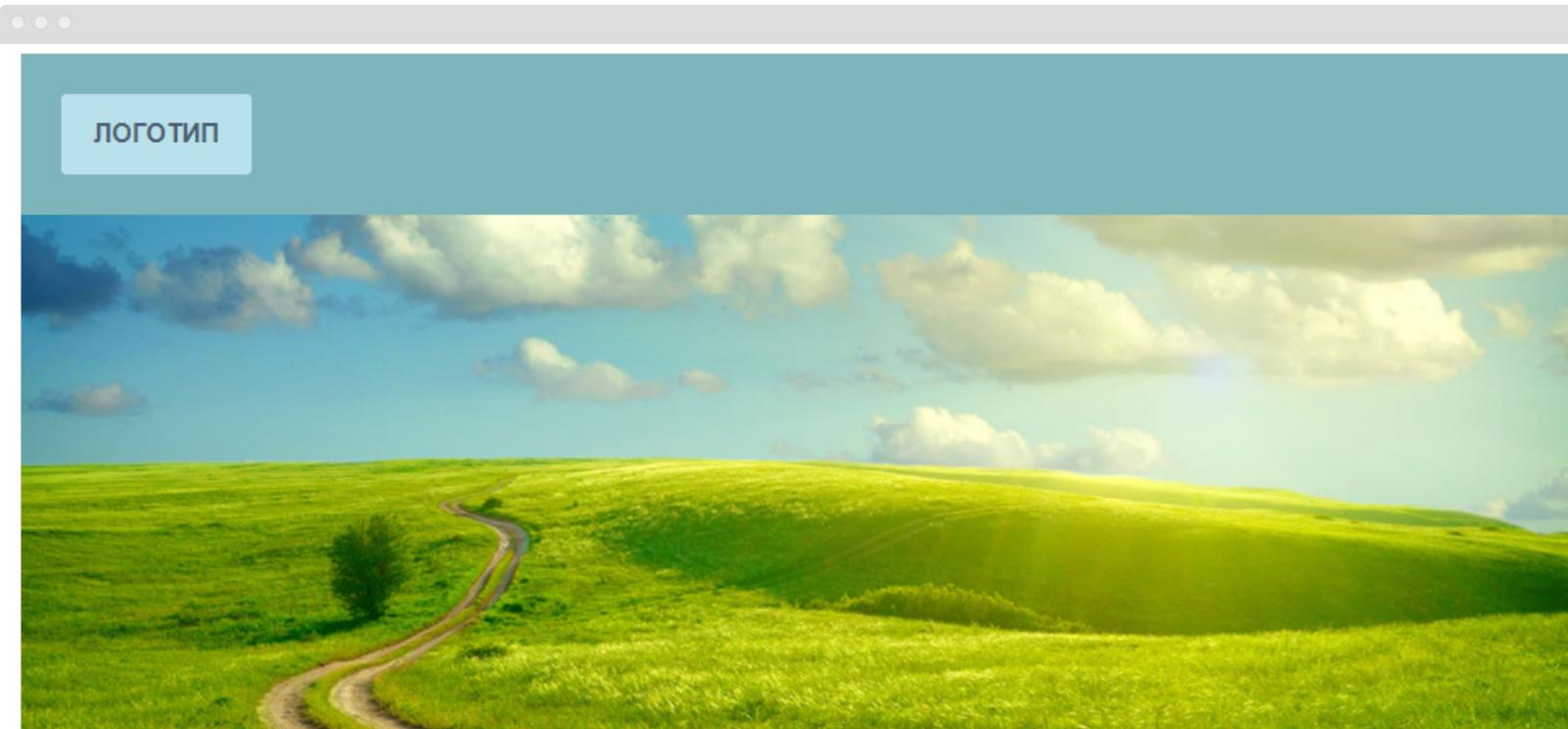
СЛИШКОМ БОЛЬШАЯ КАРТИНКА

Изображение ушло за пределы контейнера.



РЕШЕНИЕ С width

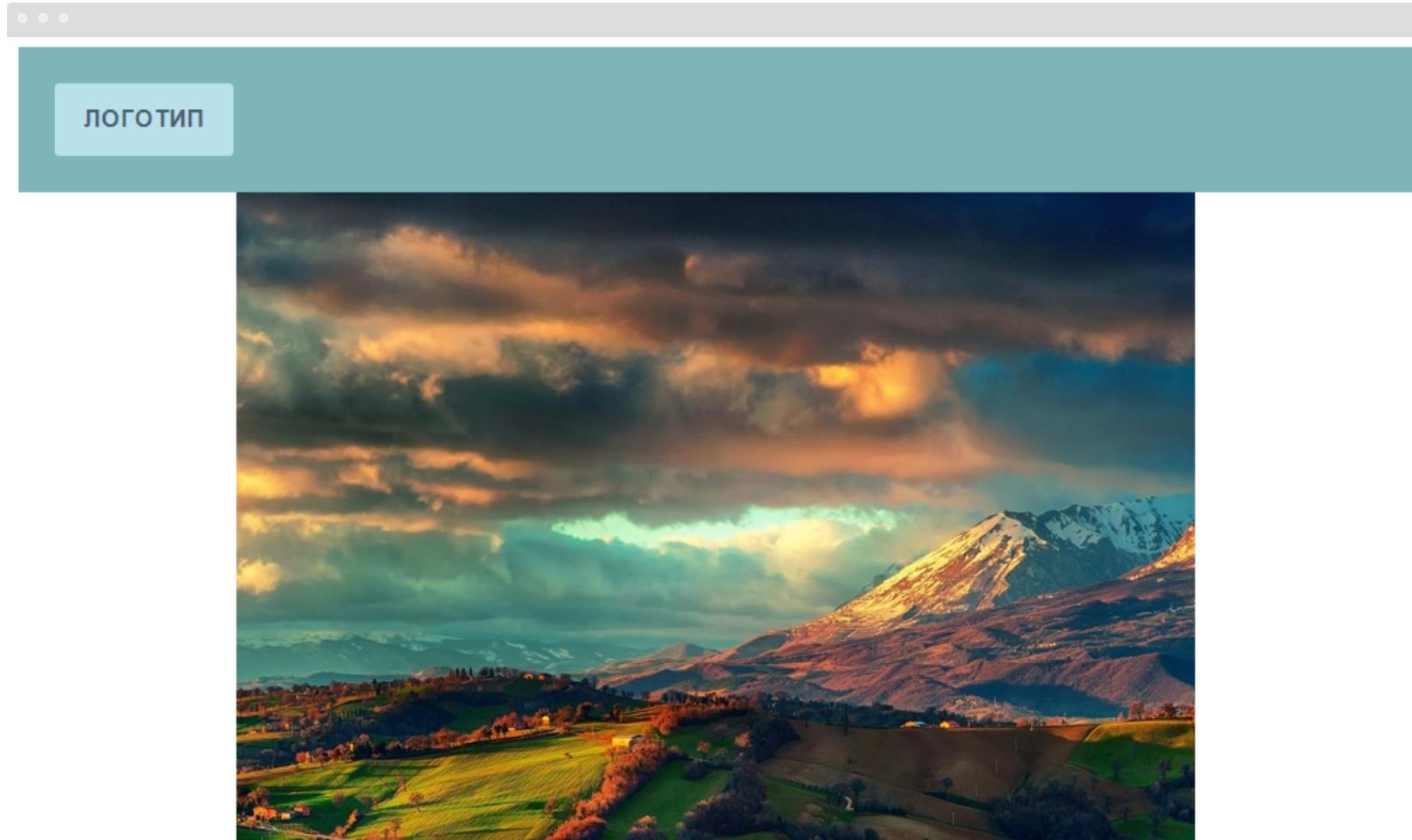
```
1 | .image {  
2 |   width: 100%;  
3 | }
```



НЕТ ГАРАНТИИ РЕЗУЛЬТАТА

Изображение адаптировалось под ширину родительского контейнера и сохранило пропорции, однако в случае с динамически генерируемыми страницами нет никакой гарантии, что размеры этого изображения будут больше размеров родительского контейнера.

КАРТИНКА МЕНЬШЕ РОДИТЕЛЯ



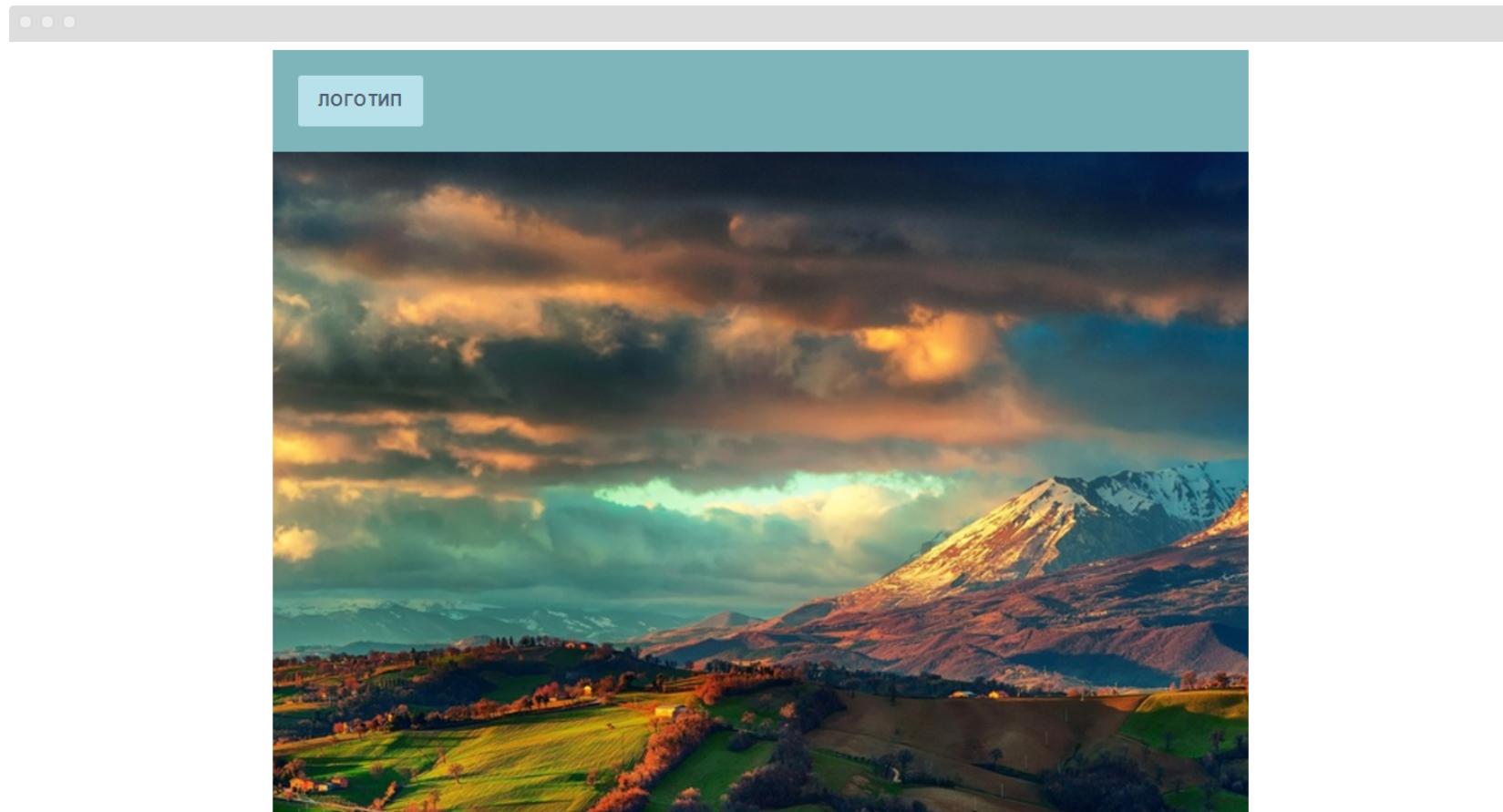
СНОВА ПРОБУЕМ `width`

Нам необходимо адаптировать ширину изображения в случае уменьшения экрана, зададим изображению ширину 100%:

```
1 | .image {  
2 |   width: 100%;  
3 | }
```

ПОТЕРЯ КАЧЕСТВА

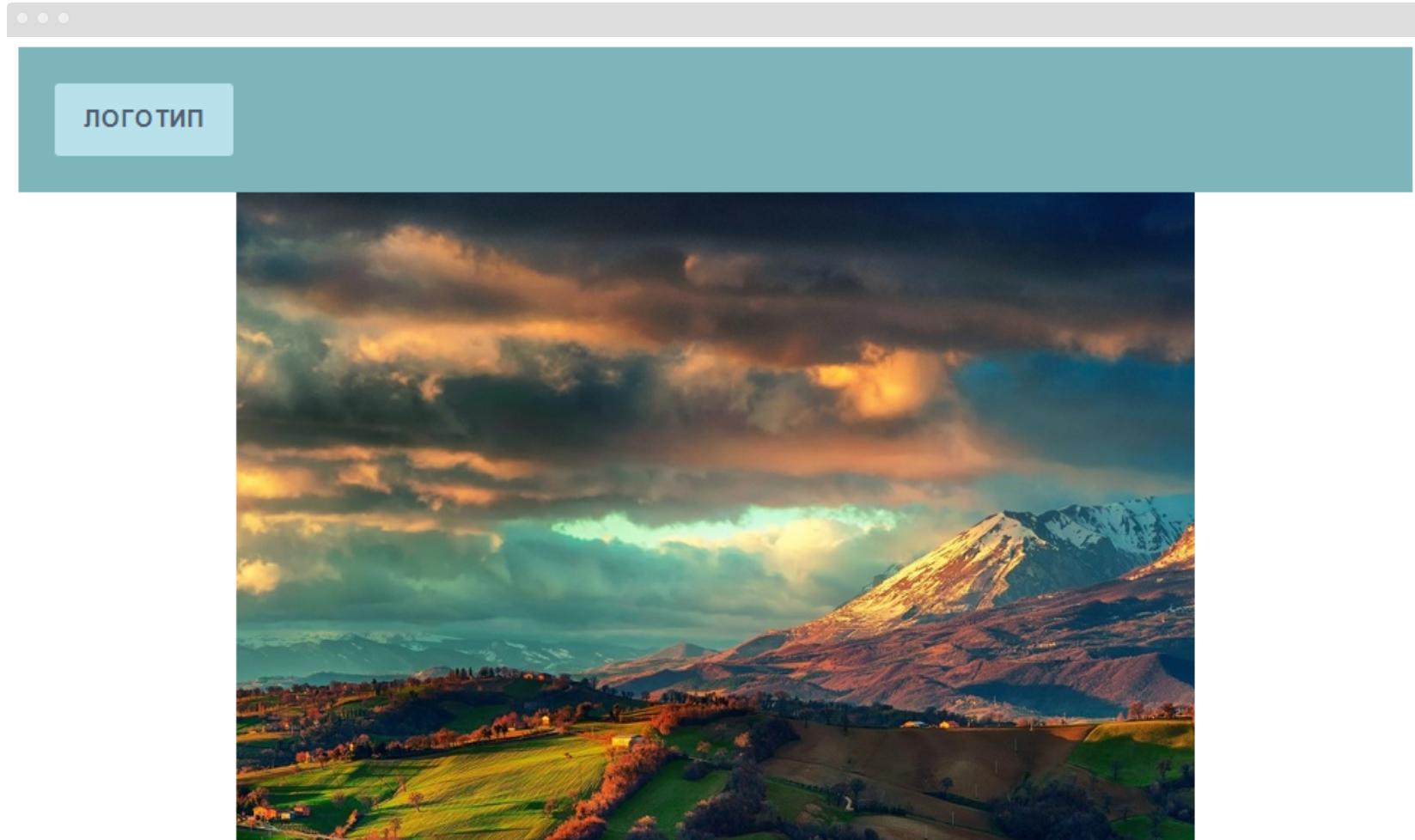
Изображение заняло всю ширину родительского контейнера и пропорционально увеличилось по высоте, заметно потеряв в качестве.



РЕШЕНИЕ С max-width

```
1 | .image {  
2 |   max-width: 100%;  
3 | }
```

КАЖЕТСЯ, НИЧЕГО НЕ ИЗМЕНИЛОСЬ



СОХРАНЕНИЕ ПРОПОРЦИЙ

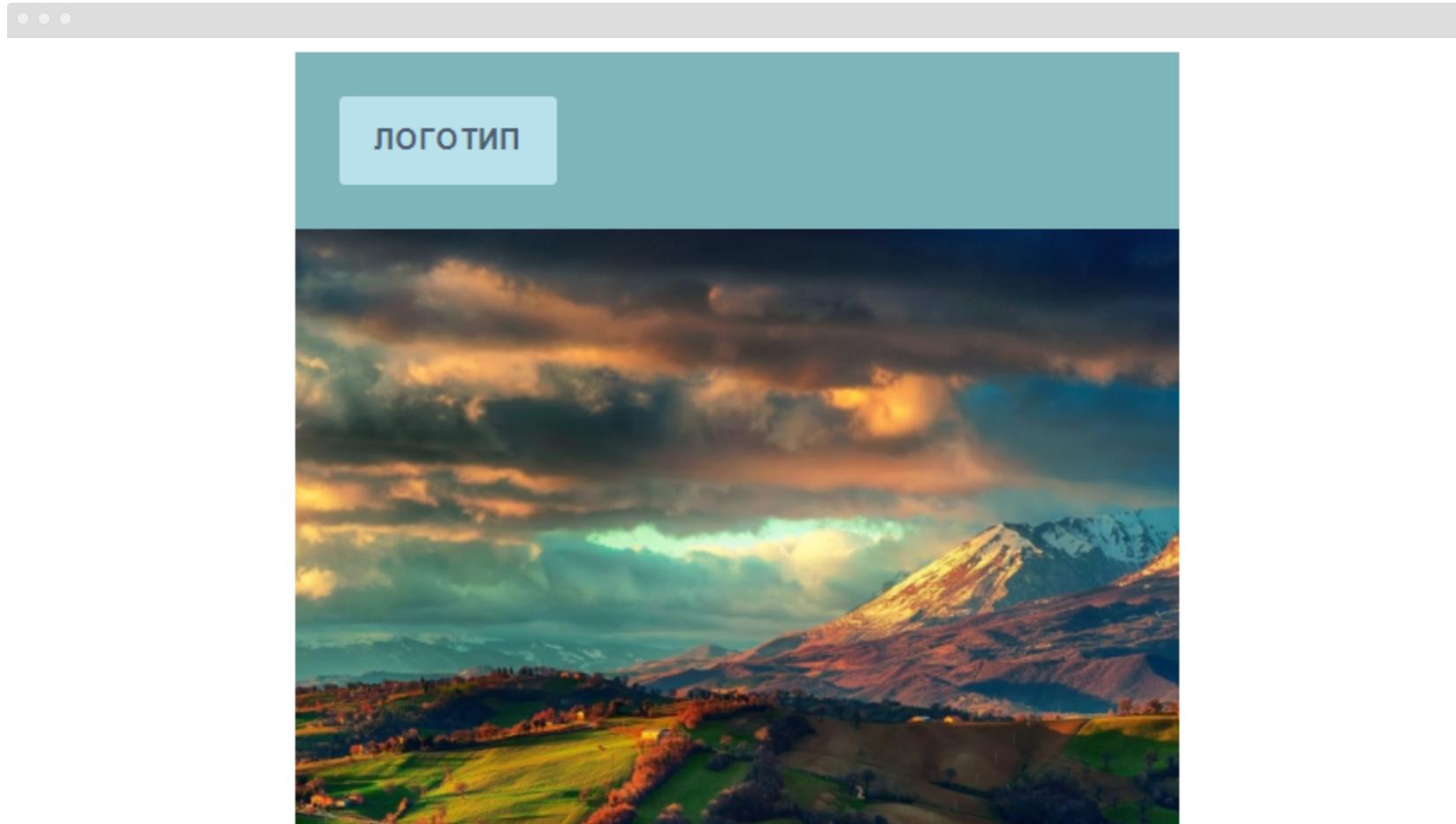
Изображение выглядит точно так же, как и на первой иллюстрации, потому что CSS-правило `max-width: 100%` говорит браузеру сохранять оригинальный размер изображения до тех пор, пока оно не превышает размеров родительского контейнера.

УМЕНЬШИМ РОДИТЕЛЯ

Если уменьшить размер родительского контейнера до 500 пикселей, то изображение также уменьшится.

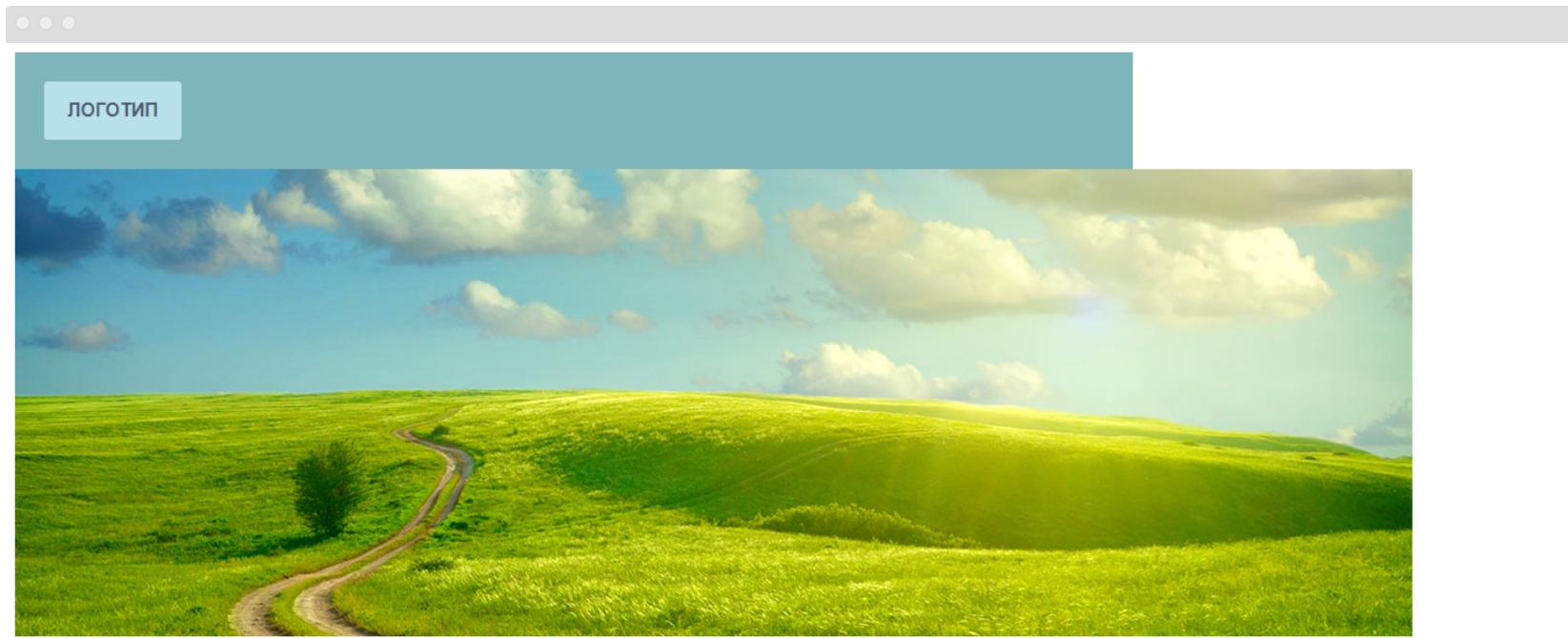
```
1 .wrapper {  
2     max-width: 500px;  
3 }  
4  
5 .image {  
6     max-width: 100%;  
7 }
```

КАРТИНКА АДАПТИРОВАЛАСЬ



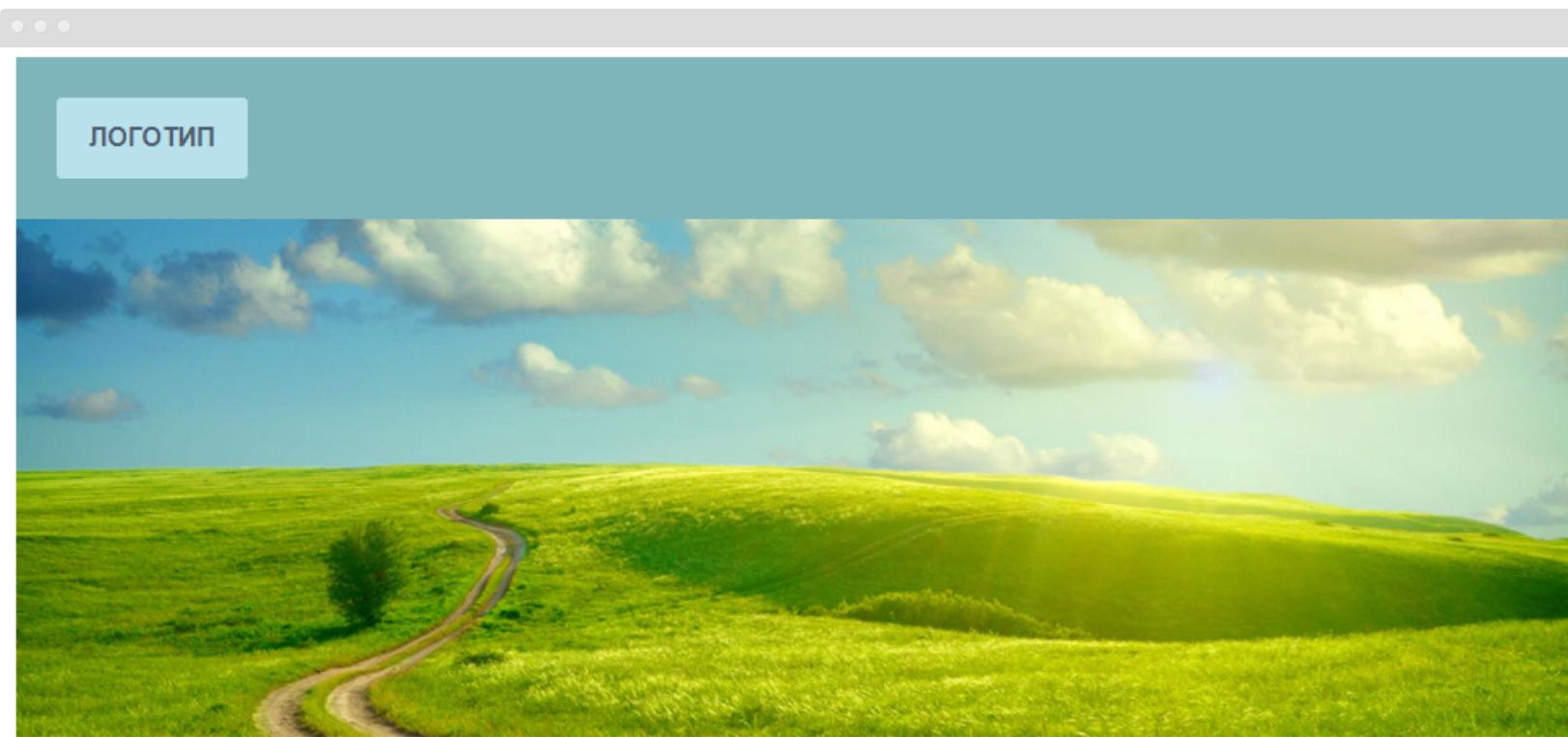
ПРОБУЕМ `max-width` С БОЛЬШОЙ КАРТИНКОЙ

Вернемся к первому примеру и проверим, верно ли для него то же самое.



ВПИСЫВАЕМ В РОДИТЕЛЯ

```
1 | .image {  
2 |   max-width: 100%;  
3 | }
```



УМЕНЬШАЕМ РОДИТЕЛЯ

```
1 | .wrapper {  
2 |     max-width: 500px;  
3 | }
```



РАЗМЕР КАРТИНКИ ЗАВИСИТ ОТ РОДИТЕЛЯ

Как и задумывалось, ширина изображения теперь равна ширине родительского контейнера – 500 пикселей.

ЗАДАЕМ ФОН

Допустим, у нас есть блок шириной `400px` и высотой `150px`. Нам нужно задать ему фоновое изображение. Например, вот это:



ПИШЕМ СТИЛИ

```
1 .image {  
2     width: 600px;  
3     height: 300px;  
4     background-image: url("1.jpeg");  
5     background-repeat: no-repeat;  
6     border: 1px solid #000000;  
7     box-sizing: border-box;  
8 }
```

НЕ ВЕСЬ ФОН ВИДЕН

Что-то пошло не так. Изображение явно больше нашего блока, и мы видим только часть:



background-size

Необходимо каким-то образом, наподобие контентных изображений, задать нашему фоновому изображению размер таким образом, чтобы он подстраивался под размер родительского блока.

В этом нам поможет свойство `background-size`.

ВОЗМОЖНЫЕ ЗНАЧЕНИЯ

Свойство `background-size` отвечает за размер фонового изображения и принимает значения в пикселях и процентах.

Значение в пикселях:

```
1 | .image {  
2 |   background-size: 150px;  
3 | }
```

Значение в процентах:

```
1 | .image {  
2 |   background-size: 25%;  
3 | }
```

РЕШАЕМ ПРОБЛЕМУ БОЛЬШОГО ФОНА

Зададим размер ширины фонового изображения равным ширине родительского блока:

```
1 .image {  
2     width: 600px;  
3     height: 300px;  
4     background-image: url("1.jpeg");  
5     background-repeat: no-repeat;  
6     border: 1px solid #000000;  
7     box-sizing: border-box;  
8     background-size: 600px;  
9 }
```

НЕ СОВПАДАЕТ СООТНОШЕНИЕ СТОРОН

Изображение заняло всю ширину родительского контейнера, однако по высоте все равно обрезано. Это произошло из-за того, что соотношение сторон родительского блока и изображения не совпадают.



АВТОМАТИЧЕСКАЯ ВЫСОТА

Свойство `background-size` принимает два значения: ширины и высоты, но по умолчанию, если не указывать высоту, то высота будет в значении `auto`, то есть, пропорциональной соотношениям сторон изображения.

ВЫСОТЫ РОДИТЕЛЯ И ФОНА СОВПАДАЮТ

Если задать высоту равной высоте родительского блока, то нет гарантии, что пропорции изображения и блока совпадут, в этом случае можно получить следующий результат:

```
1 .image {  
2     width: 600px;  
3     height: 300px;  
4     background-image: url("1.jpeg");  
5     background-repeat: no-repeat;  
6     border: 1px solid #000000;  
7     box-sizing: border-box;  
8     background-size: 600px 300px;  
9 }
```

ИСКАЖЕНИЕ ПРОПОРЦИЙ



АДАПТИРУЕМ ФОН ПО ВЫСОТЕ

Можно попробовать адаптировать изображение по высоте родительского блока, а не по ширине:

```
1 .image {  
2     width: 600px;  
3     height: 300px;  
4     background-image: url("1.jpeg");  
5     background-repeat: no-repeat;  
6     border: 1px solid #000000;  
7     box-sizing: border-box;  
8     background-size: auto 300px;  
9 }
```

ФОН ПОМЕЩАЕТСЯ ПОЛНОСТЬЮ

Выглядит уже лучше, по крайней мере, изображение полностью помещается в родительский блок.



ЦЕНТРИРУЕМ ФОН

Можно сделать этот блок более эстетичным и отцентрировать изображение с помощью свойства `background-position`:

```
1 .image {  
2     width: 600px;  
3     height: 300px;  
4     background-image: url("1.jpeg");  
5     background-repeat: no-repeat;  
6     border: 1px solid #000000;  
7     box-sizing: border-box;  
8     background-size: auto 300px;  
9     background-position: center;  
10 }
```

ВСЯ КАРТИНКА ПО ЦЕНТРУ

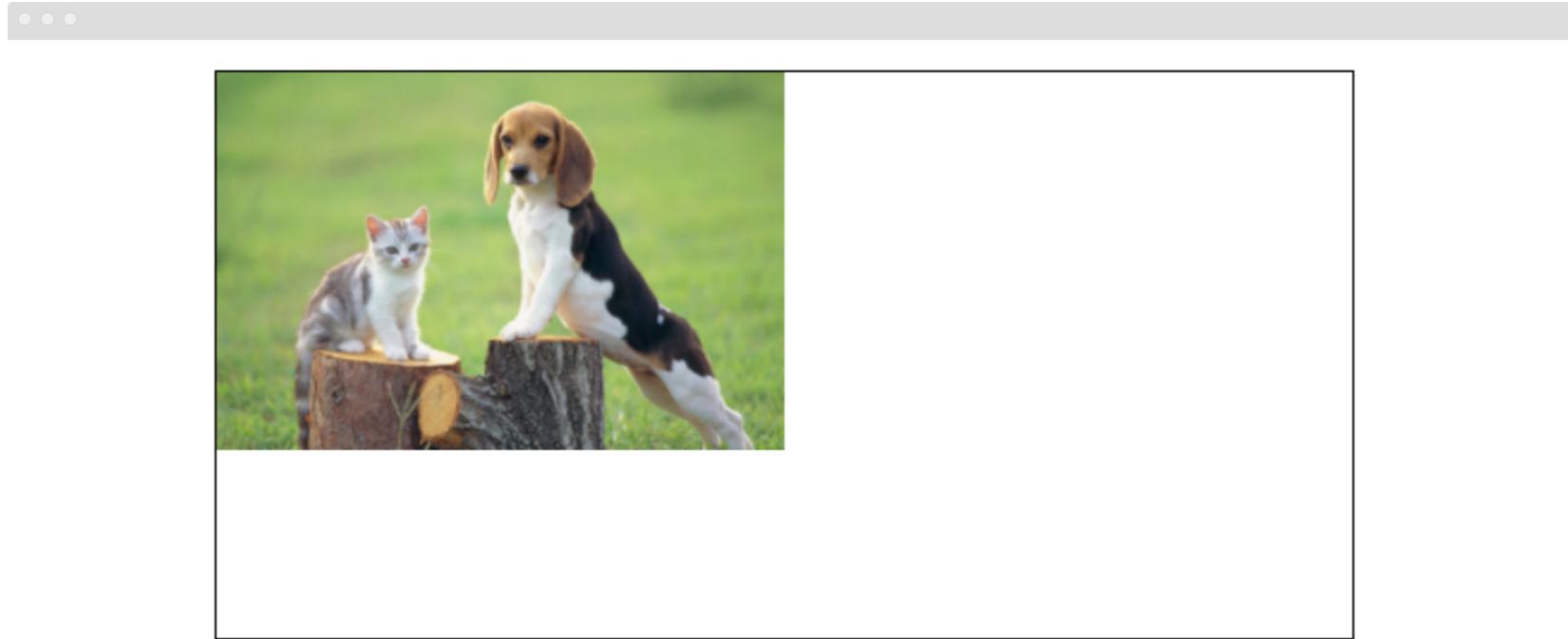


ПРОЦЕНТЫ В КАЧЕСТВЕ ЗНАЧЕНИЯ

То же самое справедливо и для процентных значений в свойстве `background-size`, только вместо фиксированных значений ширины или высоты задается значение в процентах от размеров родительского блока. Первым значением выставляется ширина, а вторым – высота. Если задано только одно значение, то второе автоматически становится `auto`.

```
1 .image {  
2     width: 600px;  
3     height: 300px;  
4     background-image: url("1.jpeg");  
5     background-repeat: no-repeat;  
6     border: 1px solid #000000;  
7     box-sizing: border-box;  
8     background-size: 50%;  
9 }
```

ФОН НА ПОЛОВИНУ ШИРИНЫ БЛОКА



Если задать `background-position: 100%`, то это будет равносильно `background-position: 600px`.

РЕЗИНОВЫЙ ФОН

Таким образом можно делать фоновые изображение резиновыми, так же, как и контентные.

ОСОБЕННОСТИ HTML-ЭЛЕМЕНТОВ

ПРОТОКОЛЫ В `href`

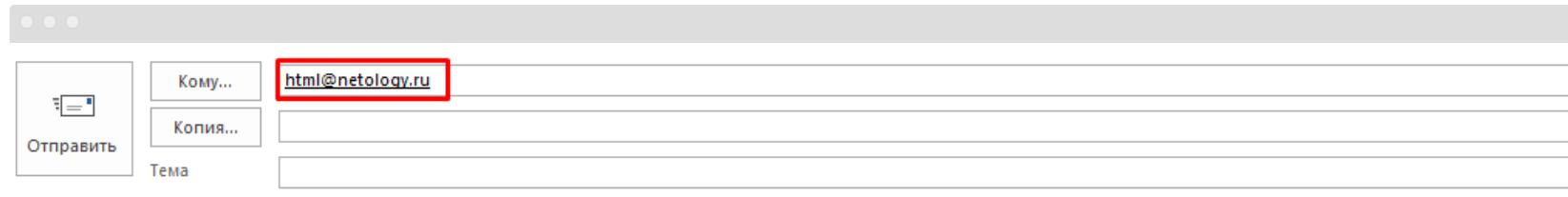
В атрибуте `href` HTML-тега `<a>` можно указывать не только ссылку на документ, но и протоколы, меняющие стандартное поведение тега `<a>`.

ПРОТОКОЛ `mailto:`

В HTML есть возможность по клику на ссылку `<a>` открыть установленный почтовый клиент с простоявшим полем адресата. Добиться этого можно, если написать в атрибуте `href` ключевое слово `mailto:` перед почтовым адресом.

ОТКРЫВАЕМ ПОЧТОВУЮ ПРОГРАММУ ПО КЛИКУ

```
<a href="mailto:html@netology.ru">Пишите!</a>
```

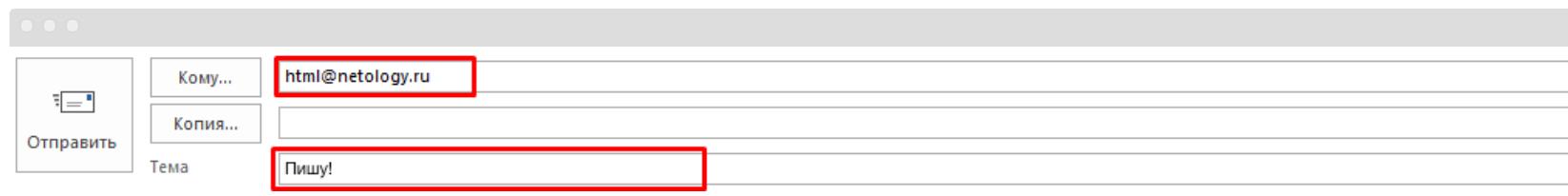


АВТОМАТИЧЕСКАЯ ТЕМА ПИСЬМА

Также протокол `mailto:` дает возможность таким же образом предзаполнить тему письма, для этого нужно через символ вопроса `?` написать ключевое слово `subject=тема письма`.

ДОБАВИМ ТЕМУ ПРЯМО В ССЫЛКУ

```
<a href="mailto:html@netology.ru?subject=Пишу!">  
    Пишите!  
</a>
```



АКТУАЛЬНО ДЛЯ МОБИЛЬНЫХ

Протокол `mailto:` особенно удобен на мобильных устройствах, где важно выполнять многие действия как можно быстрее, без необходимости заполнять данные вручную.

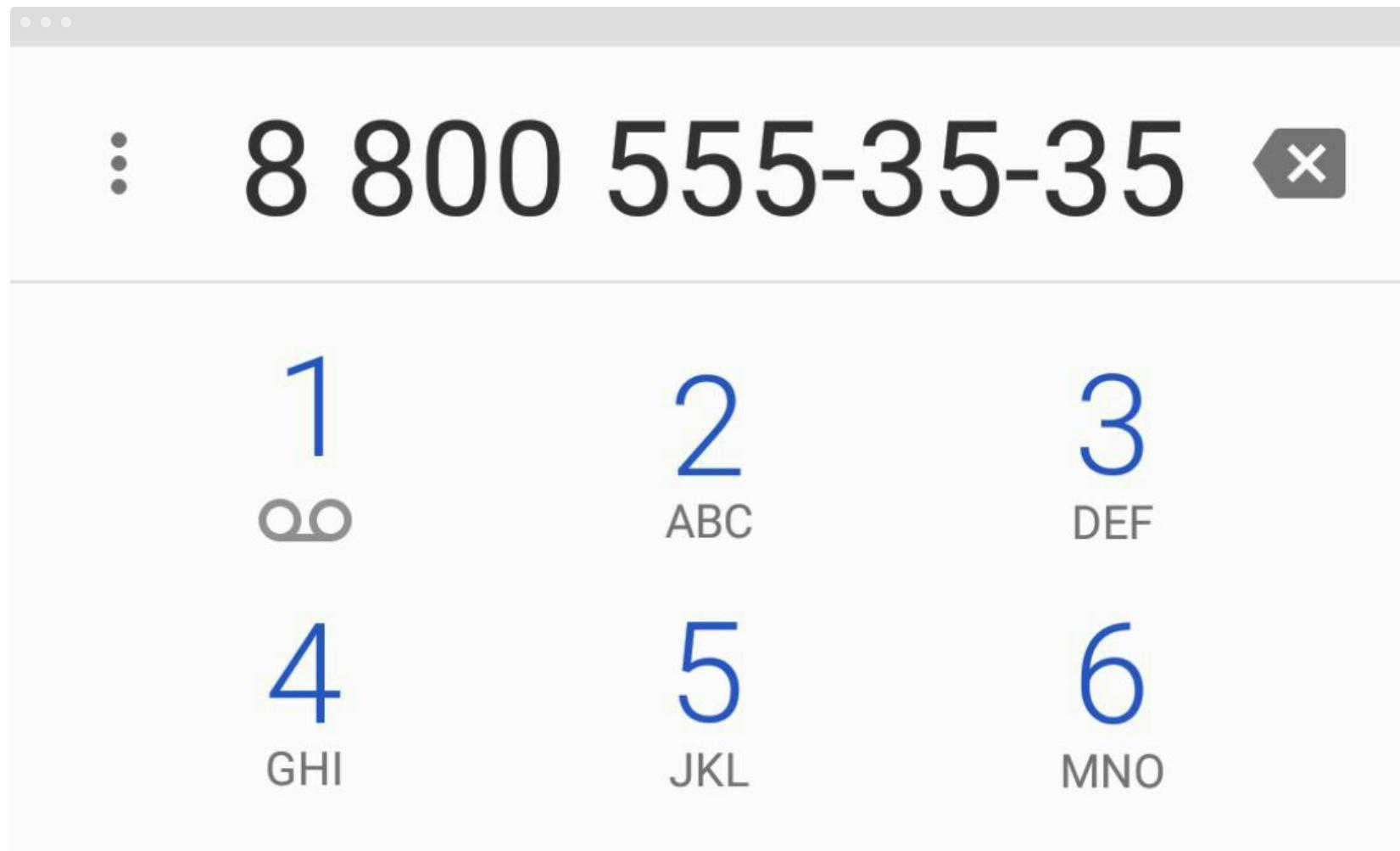
ПРОТОКОЛ tel:

Ключевое слово `tel:` по аналогии с `mailto:` позволяет переопределить обычную ссылку `<a>` и по клику на нее открыть программу, через которую осуществляются звонки, и сразу заполнить поле с номером телефона.

```
<a href="tel:88005553535">Звонок бесплатный!</a>
```

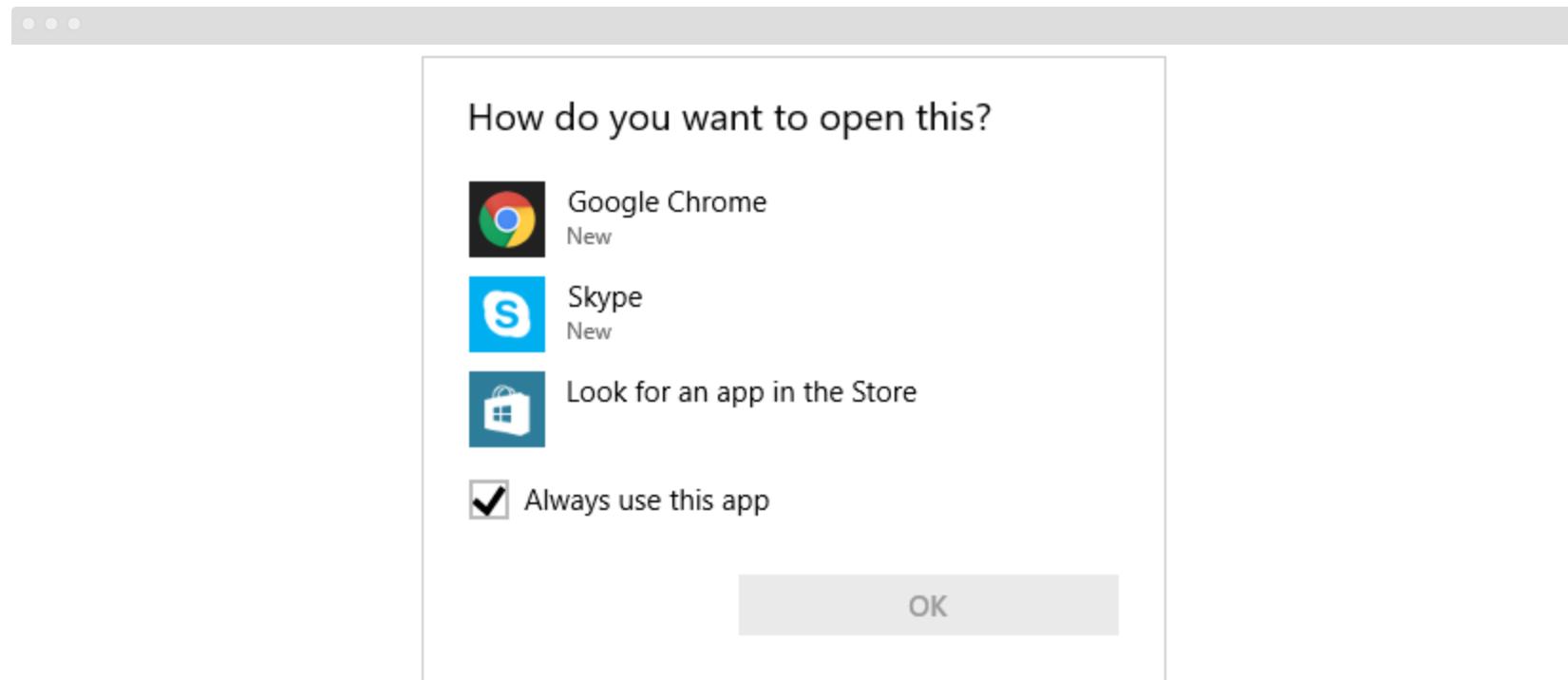
НАБОР НОМЕРА НА МОБИЛЬНОМ

Если пользователь переходит по ссылке из мобильного браузера, то с большой вероятностью откроется приложение набора номера:



НАБОР НОМЕРА НА КОМПЬЮТЕРЕ

В настольных браузерах операционная система может предложить выбор из нескольких приложений, либо сразу открыть установленное по умолчанию:



ИТОГИ

ПРИНЦИП РАСЧЕТА РАЗМЕРОВ ЭЛЕМЕНТА

- Блочная модель – алгоритм, по которому браузер рассчитывает размеры области, в которой находится элемент. В ней задействованы свойства: ширина контента `width`, высота контента `height`, внутренние отступы `padding`, рамка `border` и внешние отступы `margin`.
- Для расчета размеров элемента браузер суммирует свойства, задействованные в блочной модели. Например, для определения того, сколько места занимает элемент по горизонтали, суммирование производится по формуле: `width + padding-left + padding-right + border-left + border-right + margin-left + margin-right`.
- Важный момент: `width` и `height` устанавливают значение для размеров контента блока, а не для самого блока!

box-sizing

- Для более удобной работы с размерами с CSS3 появилось свойство `box-sizing`, позволяющее переключить алгоритм расчета размеров элементов. Оно имеет значения `content-box` (по умолчанию), `border-box`.
- При значении `content-box` браузер работает по стандартной блочной модели.
- А вот значение `border-box` изменяет порядок расчета: значения свойств `padding` и `border` уже входят в свойства `width` и `height`. А значит, теперь `width` и `height` задают размеры самого блока.

ФУНКЦИЯ calc()

- Функция `calc()` используется для указания вычисляемого значения свойств, которые в качестве значений используют размер, угол, время или число.
- В `calc()` можно делать следующие вычисления, используя выражения: сложение (`+`), вычитание (`-`), умножение (`*`), деление (`/`). На ноль делить запрещено!
- Если значение не может быть вычислено, оно игнорируется.

КОНТЕНТНЫЕ ИЗОБРАЖЕНИЯ

- Контентные изображения используются для донесения важной визуальной информации до пользователя. Для добавления такого изображения на страницу используется HTML-тег ``.
- Атрибут `alt` тега `` задает альтернативный текст к изображению, который отображается, если изображение не смогло загрузиться. Этот атрибут важен для обеспечения доступности сайта людям с ограниченными зрительными возможностями.
- Еще один полезный, но не обязательный атрибут тега – `title`; он позволяет отобразить текстовое описание при наведении курсора на изображение.

ФОНОВЫЕ ИЗОБРАЖЕНИЯ

- Фоновые изображения используются в декоративных целях и не несут для пользователя полезной информации. Если скрыть такое изображение, информация на сайте все равно будет доступна для восприятия. Для реализации лучше использовать `background-image`.
- Еще один важный момент: для фоновых изображений-подложек под текст рекомендуется указывать фоновый цвет с помощью свойства `background-color`, чтобы избежать потери читаемости информационного блока в случае ошибки загрузки изображения.
- Для скрытия блока с фоном рекомендуется использовать универсальный и работающий во всех браузерах способ – свойство `background: none`.

РЕЗИНОВЫЕ ИЗОБРАЖЕНИЯ, СВОЙСТВО `background-size`

- Размеры резиновых изображений зависят от размеров родительского блока. Чаще изображения адаптируют по ширине.
- Чтобы сделать резиновое адаптивное по ширине изображение, нужно задать тегу `` свойство `max-width: 100%`.
- Для задания фоновых изображений, подстраивающихся под размер родительского блока, используют свойство `background-size`, отвечающее за размер фонового изображения и принимающее значения в пикселях и процентах. Если не указать высоту, то она по умолчанию будет в значении `auto` – пропорциональной соотношениям сторон изображения.
- Центрировать фоновое изображение можно с помощью свойства `background-position`.

ОСОБЕННОСТИ HTML-ЭЛЕМЕНТОВ

- В атрибуте `href` HTML-тега `<a>` можно указывать не только ссылку на документ, но и протоколы, меняющие стандартное поведение тега.
- Ключевое слово `mailto:`, прописанное в атрибуте `href` перед почтовым адресом, позволяет по клику по ссылке открывать установленный почтовый клиент с проставленным полем адресата.
- Также протокол `mailto:` дает возможность предзаполнить тему письма. Для этого нужно через символ вопроса `?` написать ключевое слово `subject=тема письма`.
- Ключевое слово `tel:` по аналогии с `mailto:` позволяет переопределить обычную ссылку `<a>` и по клику на нее открыть программу для осуществления звонков и сразу заполнить поле с номером телефона.



Задавайте вопросы и напишите отзыв о лекции!

АЛЕКСАНДР БЕСПОЯСОВ



bespoyasov@me.com



[@bespoyasov](https://t.me/bespoyasov)