



# ПОЗИЦИОНИРОВАНИЕ FLEX-ЭЛЕМЕНТОВ



СЕМЕН БОЙКО



# СЕМЕН БОЙКО

Front-end разработчик



[simonderus@gmail.com](mailto:simonderus@gmail.com)



[fb.me/simonboycko](https://fb.me/simonboycko)

---

# ПЛАН ЗАНЯТИЯ

## 1. Позиционирование элементов

- Свойство `align-items`
- Свойство `justify-content`
- Свойство `position: fixed`
- Свойство `display:none`

## 2. Специфичность CSS-селекторов

## 3. Работа в Adobe Photoshop

- Свойства шрифта
- Цвет
- Изображения

# НАША ЗАДАЧА

Сегодня нам поступила довольно интересна задача. Нам нужно сверстать всплывающее окно, которое будет напоминать пользователю о курсах, которые скоро начнутся.

Ки  
веб  
О т  
до  
на  
Но  
гад  
изд  
Но  
«гу  
обла  
Над  
моз  
В ка  
вход  
буд

Х

**Осталась одна неделя!**

Успей записаться на следующие курсы:

**Профессия fullstack-дизайнер**

Fullstack-дизайнер способен самостоятельно разработать концепцию проекта и выполнить всю работу по её воплощению: создать прототипы, разработать дизайн, нарисовать все компоненты и сверстать их, используя HTML и CSS, а также курировать работу разработчиков по интеграции интерфейсов в приложение.

**ЗАПИСАТЬСЯ**

**Digital-старт: первый шаг к востребованной профессии**

Обзор профессий мира digital. С чего начать свой путь в digital, как выбрать ту специализацию, которая пройдет именно вам. Как представлять себя и свои умения заказчику, грамотно вести деловые переговоры и переписку, эффективно организовывать работу над проектом. Особенности работы на фрилансе и удаленно. Возможные направления развития в digital.

**ЗАПИСАТЬСЯ**

Об этих (и некоторых других) технологиях и инструментах мы сегодня и поговорим. Все они являются базовыми составляющими

Демо

# НАША ЗАДАЧА

Работу над задачей начал другой разработчик и передал ее нам в следующем виде.

**Осталась одна неделя!**

Успей записаться на следующие курсы:

**Профессия fullstack-дизайнер**

Fullstack-дизайнер способен самостоятельно разработать концепцию проекта и выполнить всю работу по её воплощению: создать прототипы, разработать дизайн, нарисовать все компоненты и сверстать их, используя HTML и CSS, а также курировать работу разработчиков по интеграции интерфейсов в приложение.

**Digital-старт: первый шаг к востребованной профессии**

Обзор профессий мира digital. С чего начать свой путь в digital, как выбрать ту специализацию, которая пройдет именно вам. Как представлять себя и свои умения заказчику, грамотно вести деловые переговоры и переписку, эффективно организовывать работу над проектом. Особенности работы на фрилансе и удаленно. Возможные направления развития в digital.

**ЗАПИСАТЬСЯ**

**ЗАПИСАТЬСЯ**

*Читать еще: «Справляемся с техническим долгом»*

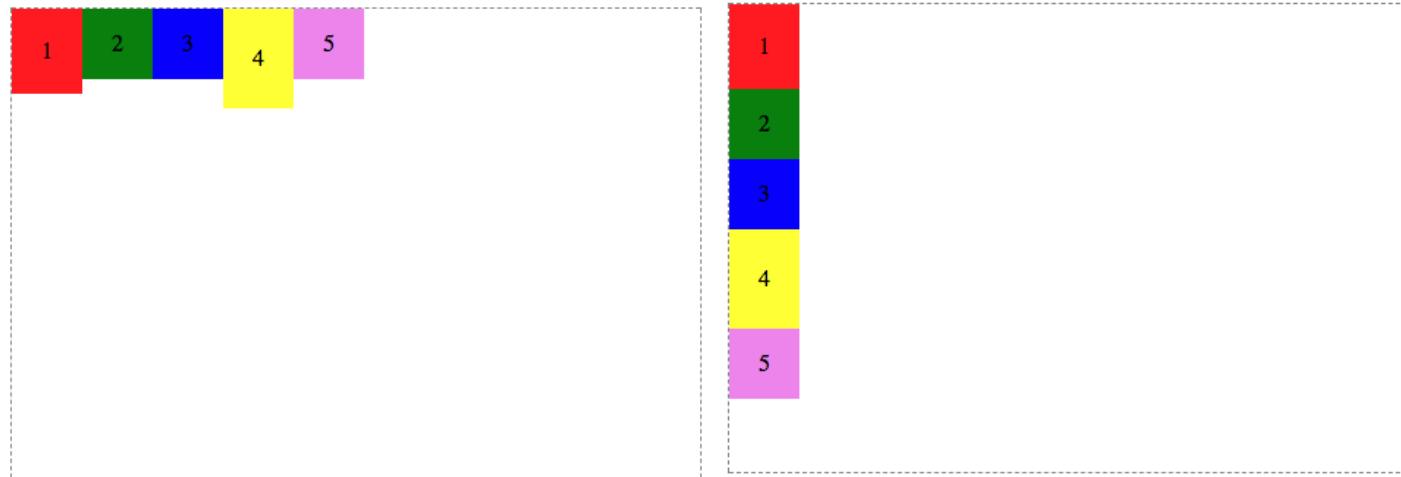
# СВОЙСТВО align-items

Свойство `align-items` отвечает за позиционирование элемента по дополнительной оси. У элемента есть следующие значения:

- `flex-start`;
- `flex-end`;
- `center`;
- `baseline`;
- `stretch`.

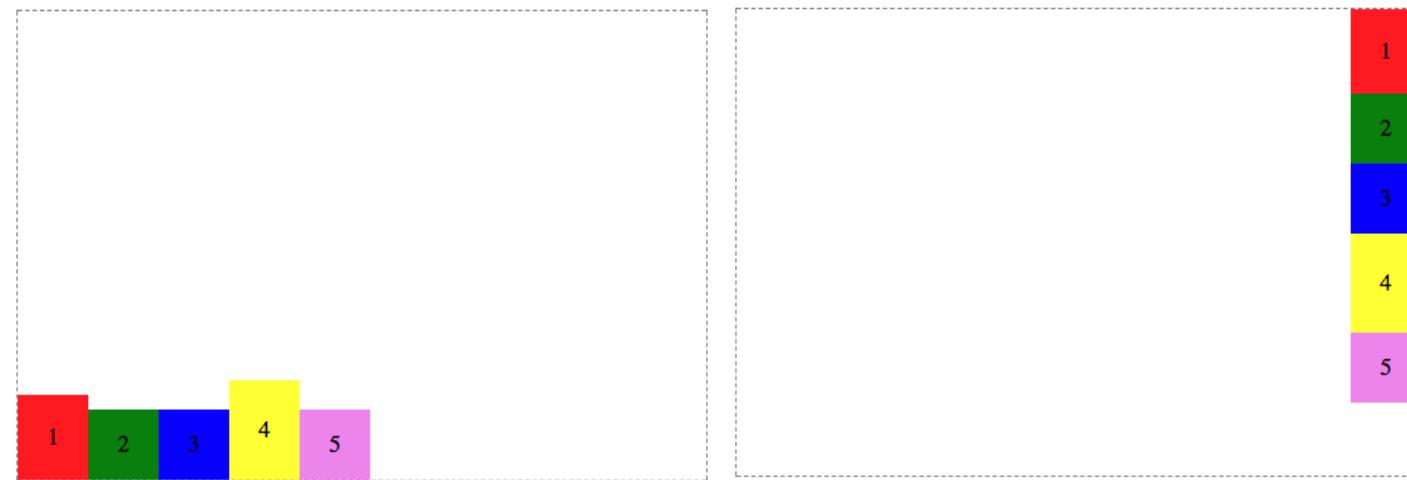
# flex-start

При `align-items: flex-start` браузер переместит элементы в начало flex-контейнера:



# flex-end

Значение `flex-end` располагает flex-элементы в конце flex-контейнера:



# center

Значение `center` отцентрирует элементы по дополнительной оси:



# baseline

Значение `baseline` выравнивает элементы по базовой линии:

...



# stretch

При этом значении flex-элементы растягиваются по дополнительной оси:



`stretch` является значением по умолчанию и его можно не указывать.  
Именно поэтому наша кнопка и оказалась растянутой.

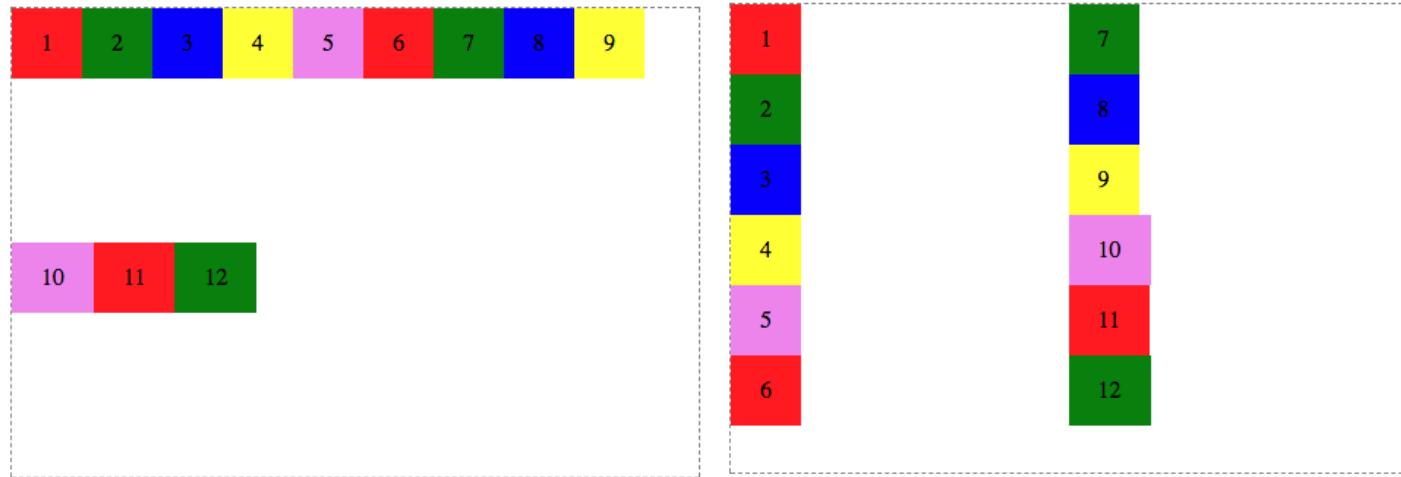
# СВОЙСТВО justify-content

Свойство `justify-content` отвечает за позиционирование элемента по основной оси. У элемента есть следующие значения:

- `flex-start`;
- `flex-end`;
- `space-around`;
- `space-between`;
- `space-evenly`.

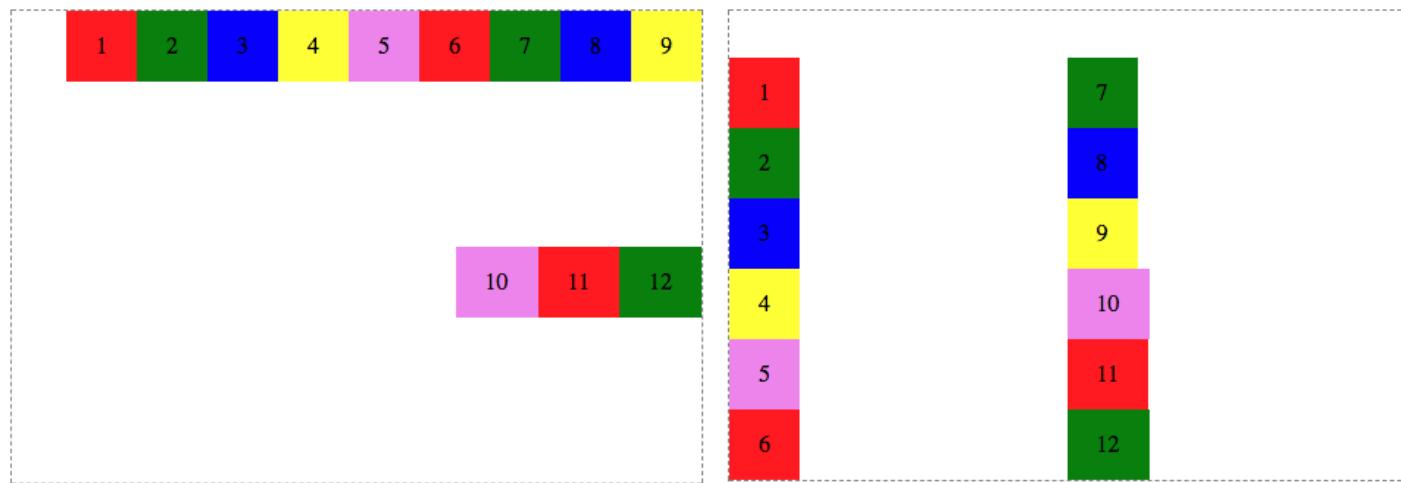
# flex-start

Значение по умолчанию. При этом flex-элементы выстраиваются в начале flex-контейнера:



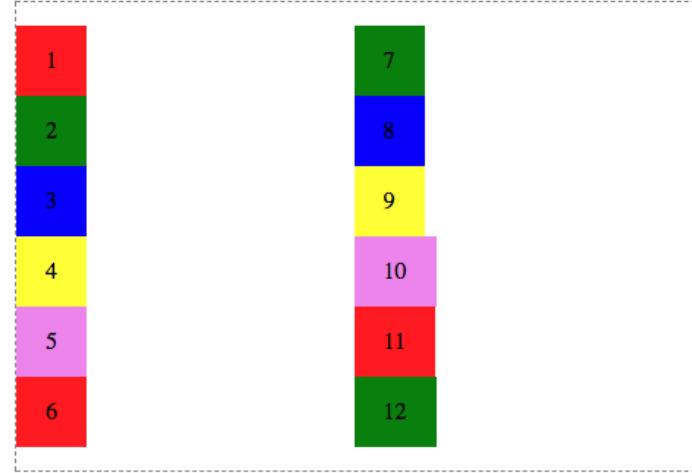
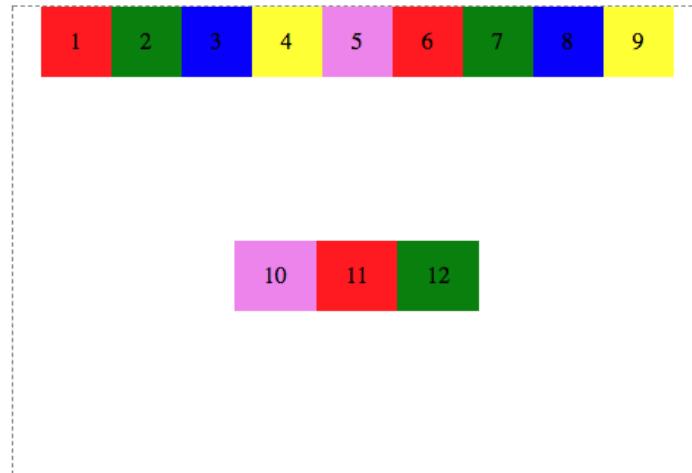
# flex-end

При этом значении flex-элементы выстраиваются в конце flex-контейнера:



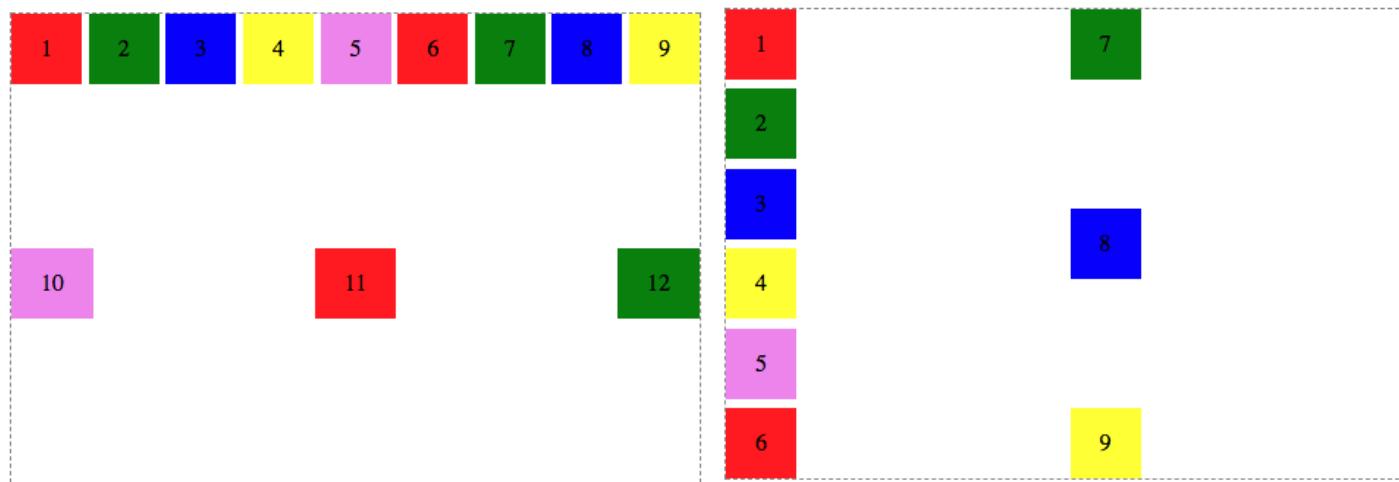
# center

Значение выравнивает элементы по центру контейнера:



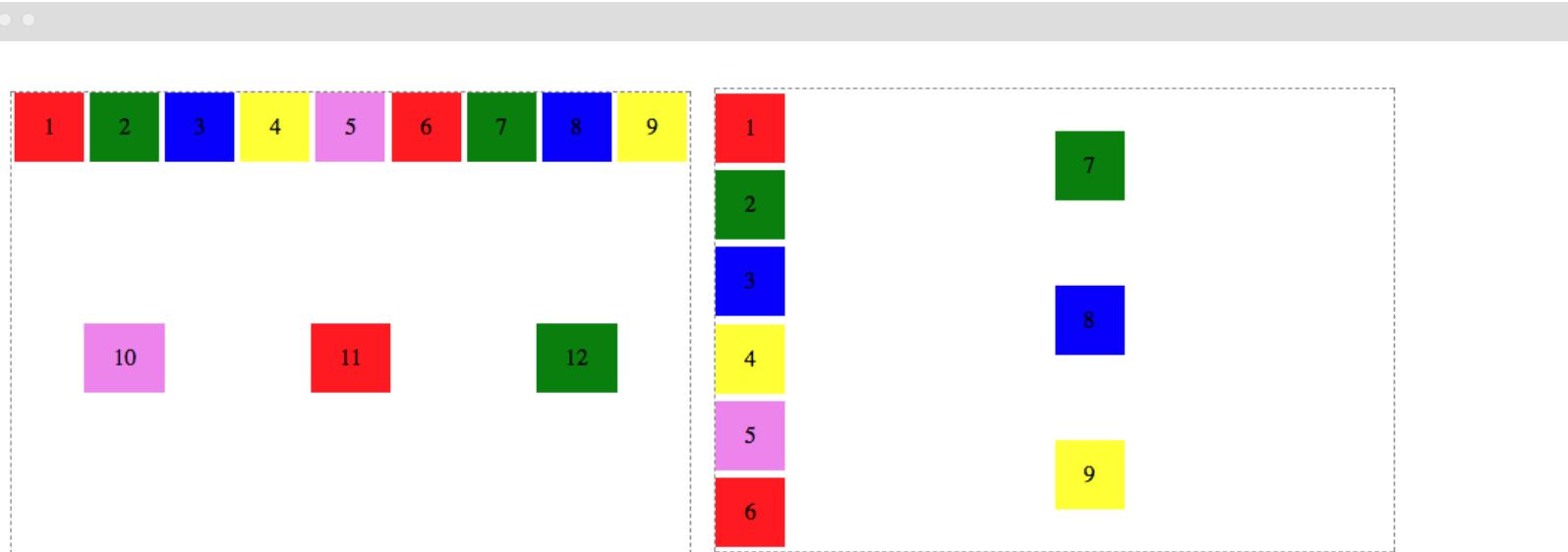
# space-between

Располагает элементы равномерно так, что первый и последний находятся вплотную к краям.



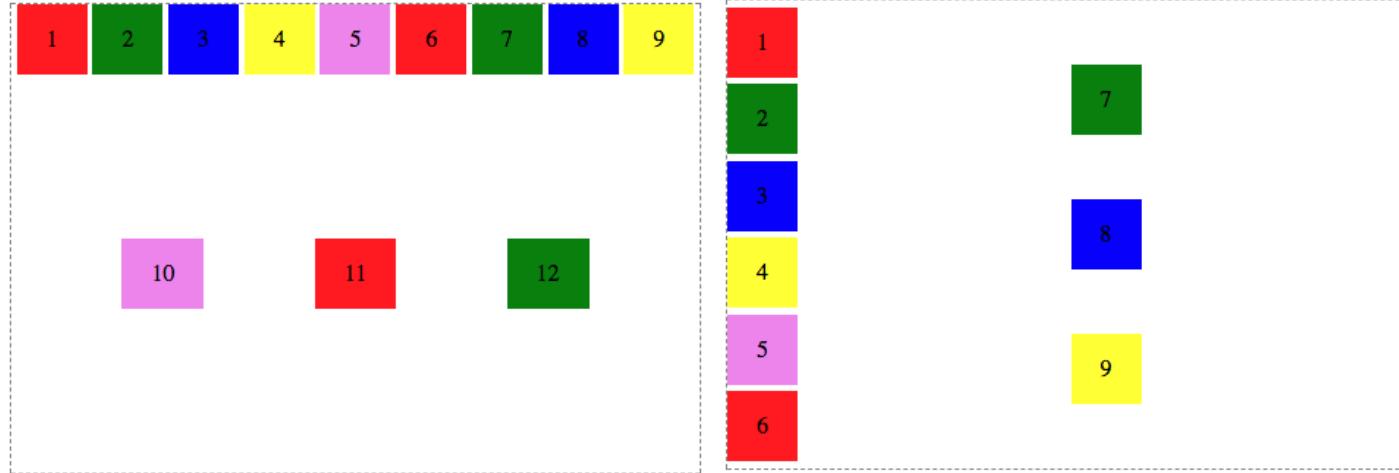
# space-around

Располагает элементы равномерно, при этом расстояние между ними одинаково, а расстояние от первого/последнего элемента до краев равняется половине интервала между элементами.



# space-evenly

Располагает элементы равномерно, при этом расстояния между ними, а также от первого/последнего элемента до краев одинаковы.



# ДОБАВИМ СВОЙСТВА

Сделаем элемент с классом `popup` flex-контейнером и добавим свойства `align-items` и `justify-content`:

```
1 .popup {  
2   display: flex;  
3   justify-content: center;  
4   align-items: center;  
5 }
```

Осталась одна неделя!

Успей записаться на следующие курсы:

**Профессия fullstack-дизайнер**

Fullstack-дизайнер способен самостоятельно разработать концепцию проекта и выполнить всю работу по её воплощению: создать прототипы, разработать дизайн, нарисовать все компоненты и сверстать их, используя HTML и CSS, а также курировать работу разработчиков по интеграции интерфейсов в приложение.

**ZАПИСАТЬСЯ**

**Digital-старт: первый шаг к востребованной профессии**

Обзор профессий мира digital. С чего начать свой путь в digital, как выбрать ту специализацию, которая пройдет именно вам. Как представлять себя и свои умения заказчику, грамотно вести деловые переговоры и переписку, эффективно организовывать работу над проектом. Особенности работы на фрилансе и удаленно. Возможные направления развития в digital.

**ЗАПИСАТЬСЯ**

читать еще. «Справляемся с техническим долгом»

# ПРОКРУТИМ СТРАНИЦУ

ПРОФЕССИЯ

нарисовать все компоненты и сверстать их, используя HTML и CSS, а также курировать работу разработчиков по интеграции интерфейсов в приложение.



## Digital-старт: первый шаг к востребованной профессии

Обзор профессий мира digital. С чего начать свой путь в digital, как выбрать ту специализацию, которая пройдет именно вам. Как представлять себя и свои умения заказчику, грамотно вести деловые переговоры и переписку, эффективно организовывать работу над проектом. Особенности работы на фрилансе и удаленно. Возможные направления развития в digital.

ЗАПИСАТЬСЯ

несколько основных технологий, которые входят в арсенал каждого программиста. Тогда сразу для ряда своих задач, о которых мы поговорим дальше в статье, ему больше не будут нужны программисты, и так слишком увлеченные разработкой новых фич.

Об этих (и некоторых других) технологиях и инструментах мы сегодня и поговорим. Все они являются базовыми составляющими современной веб-разработки (и не только веб-разработки), и умение их использовать может послужить как во благо не связанной напрямую с программированием деятельности, так и в качестве фундамента будущей карьеры программиста.

Окно сдвигается вместе со страницей.

# position: fixed

Когда у элемента есть значение `fixed` для свойства `position`, то он начинает обладать следующими свойствами:

- Элемент фиксируется при прокрутке страницы;
- Свойства `width` и `height` рассчитываются по содержимому;
- Для свойств `width` и `height` можно установить значения;
- Элемент не видим для родителя и соседних элементов.

# ЭЛЕМЕНТ ФИКСИРУЕТСЯ ПРИ ПРОКРУТКЕ СТРАНИЦЫ



НЕТОЛОГИЯ Маркетинг Бизнес и управление Дизайн и UX Программирование Data Science 8 (800) 301-39-69 📧

## О Нетологии

«Нетология» — это университет по подготовке и дополнительному обучению специалистов в области интернет-маркетинга, управления проектами, дизайна, проектирования интерфейсов и веб-разработки.

преподают эксперты компаний:



НЕТОЛОГИЯ Маркетинг Бизнес и управление Дизайн и UX Программирование Data Science 8 (800) 301-39-69 📧



## Реальные проекты

Отрабатывайте полученные знания на

# СВОЙСТВА `width` И `height` РАССЧИТЫВАЮТСЯ ПО СОДЕРЖИМОМУ

элемент `span` с `position: fixed`

элемент `span` с `position: fixed` и длинным текстом

Когда браузер встречает элемент, у которого есть `position: fixed`, то свойства `width` и `height` рассчитываются по содержимому элемента.

# для свойств width и height можно установить значения

```
1 span {  
2   position: fixed;  
3   width: 200px;  
4   height: 150px;  
5 }
```

элемент span с  
position: fixed

# ЭЛЕМЕНТ С `position: fixed` НЕ ВИДИМ ДЛЯ РОДИТЕЛЯ

Родительский элемент `div` (с голубой рамкой) содержит в себе дочерний элемент `div`.

Родительский контейнер

элемент `div` с `position: fixed`

Когда у дочернего элемента есть `position: fixed`, то он становится не видимым для своего родителя.

Родительский контейнер

элемент `div` с `position: fixed`

# ЭЛЕМЕНТ С position: fixed НЕ ВИДИМ ДЛЯ СОСЕДНИХ ЭЛЕМЕНТОВ

По умолчанию два элемента `div` располагаются друг под другом.

Родительский контейнер

элемент `div` с `position: fixed`

соседний элемент

Когда элементу добавлено `position: fixed`, то он становится невидим для своих соседнего элемента.

Родительский контейнер

элемент `div` с `position: fixed`

# ПОЗИЦИОНИРОВАНИЕ ЭЛЕМЕНТА С position: fixed

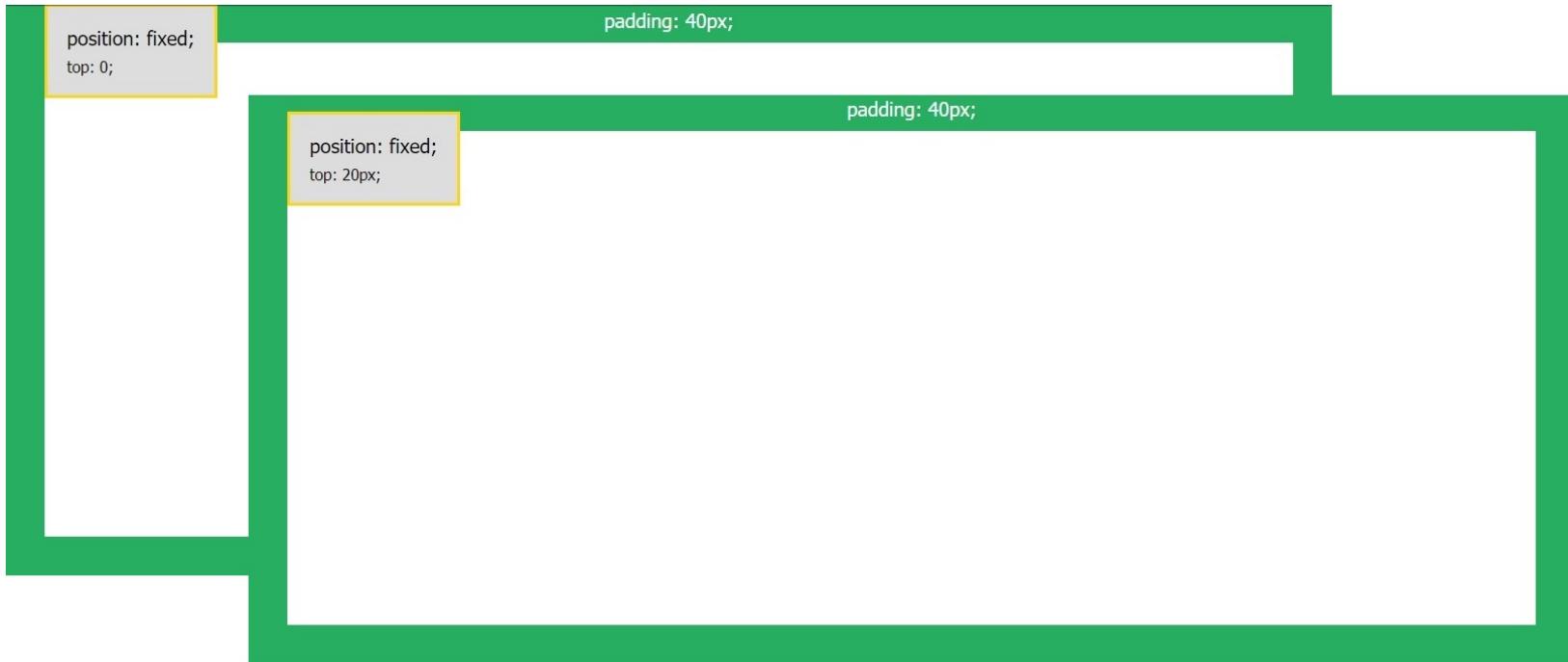
```
1 <body>
2   <div class="box">
3     position: fixed;
4   </div>
5 </body>
```

```
1 body {
2   margin: 0;
3   padding: 40px;
4 }
5
6 .box {
7   position: fixed;
8 }
```

# ПОЗИЦИОНИРОВАНИЕ ЭЛЕМЕНТА С position: fixed

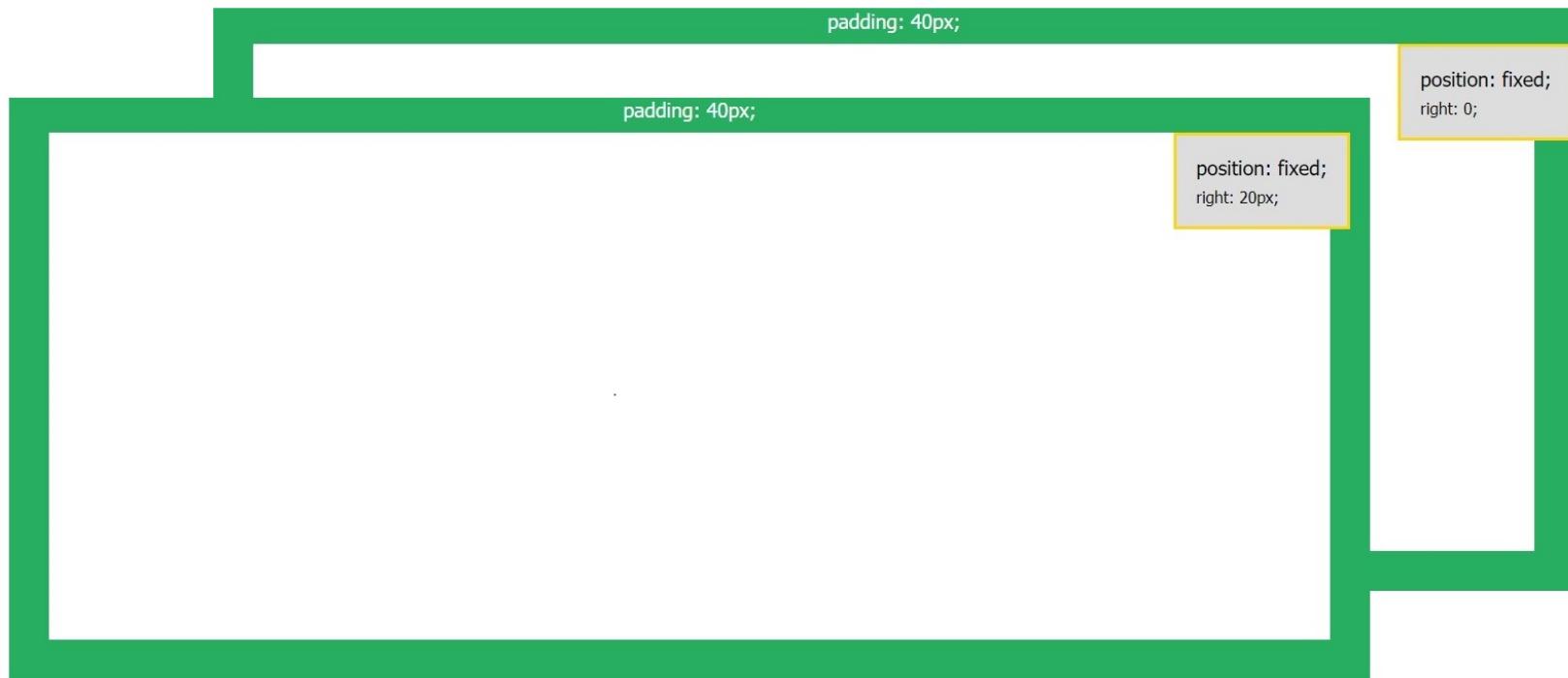


## top: 0 И top: 20px



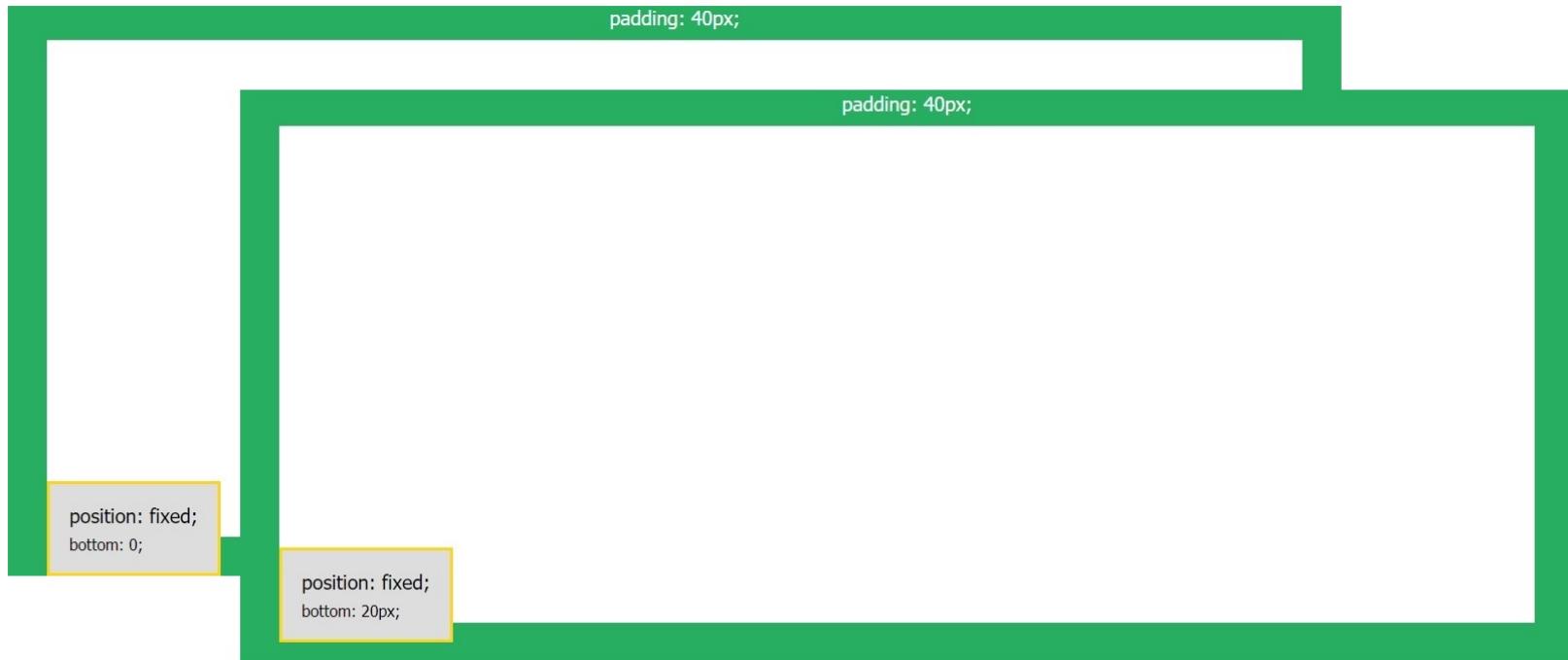
По оси X элемент сохранил прежнюю позицию, а по оси Y прижался к верхней границе элемента `body`, когда значение `0`, и отодвинулся на `20px`, когда значение `20px`.

# right: 0 и right: 20px



По оси Y элемент сохранил прежнюю позицию, а по оси X прижался к правой границе элемента `body`, когда значение `0`, и отодвинулся на `20px`, когда значение `20px`.

## bottom: 0 и bottom: 20px



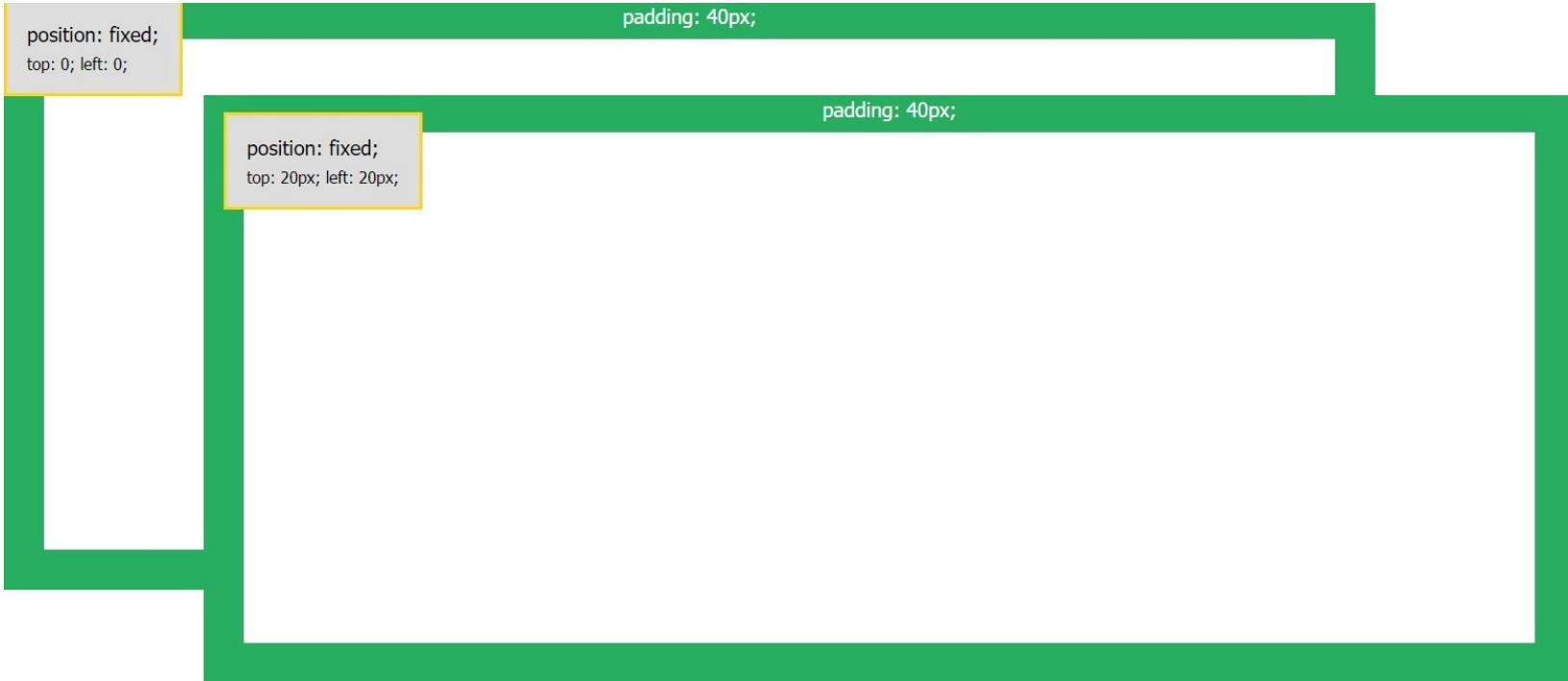
По оси X элемент сохранил прежнюю позицию, а по оси Y прижался к нижней границе элемента body, `body`, когда значение `0`, и отодвинулся на `20px`, когда значение `20px`.

## left: 0 и left: 20px



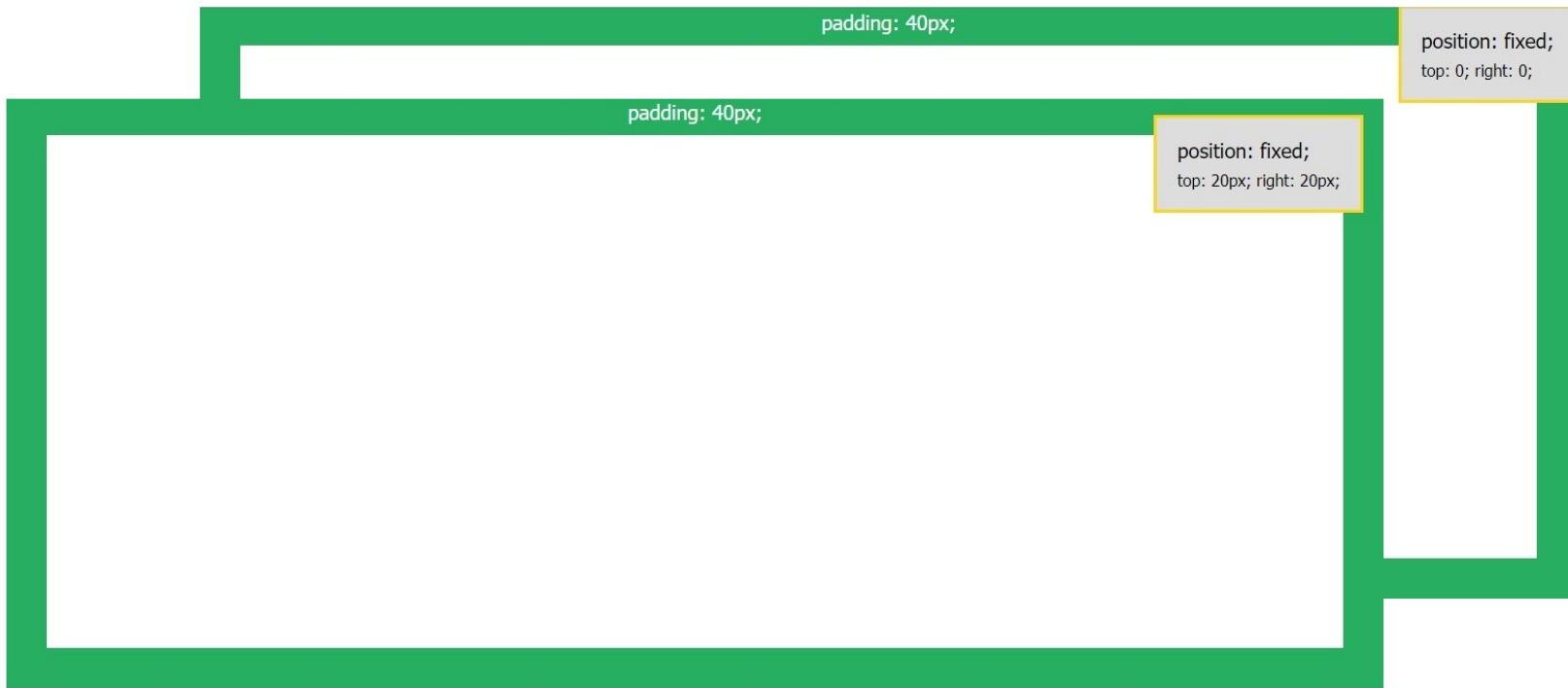
По оси Y элемент сохранил прежнюю позицию, а по оси X прижался к левой границе элемента `body`, когда значение `0`, и отодвинулся на `20px`, когда значение `20px`.

## 0 и 20px для top и left



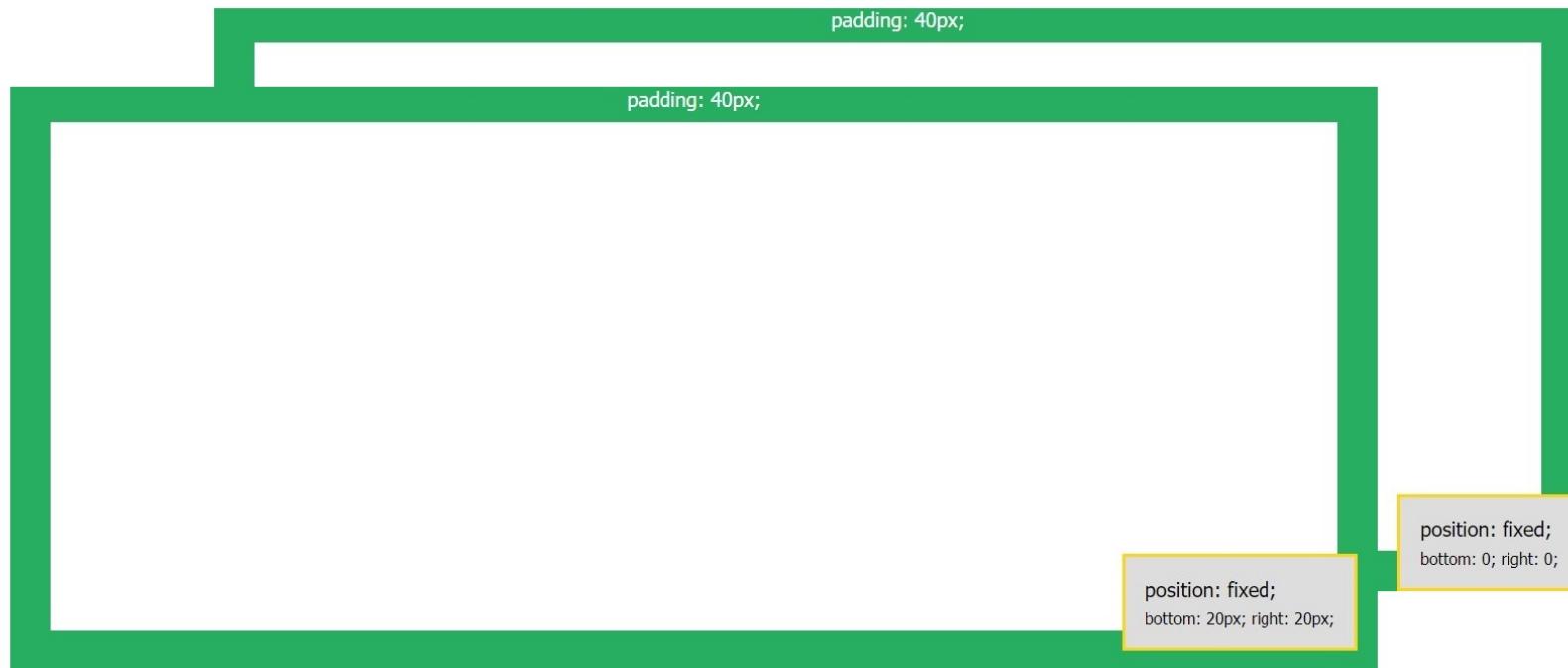
Когда к элементу применяются свойства `top` и `left` со значением `0`, то он отображается в левом верхнем углу. При значении `20px`, браузер добавляет отступ в `20px` по оси X и Y.

## 0 и 20px для top и right



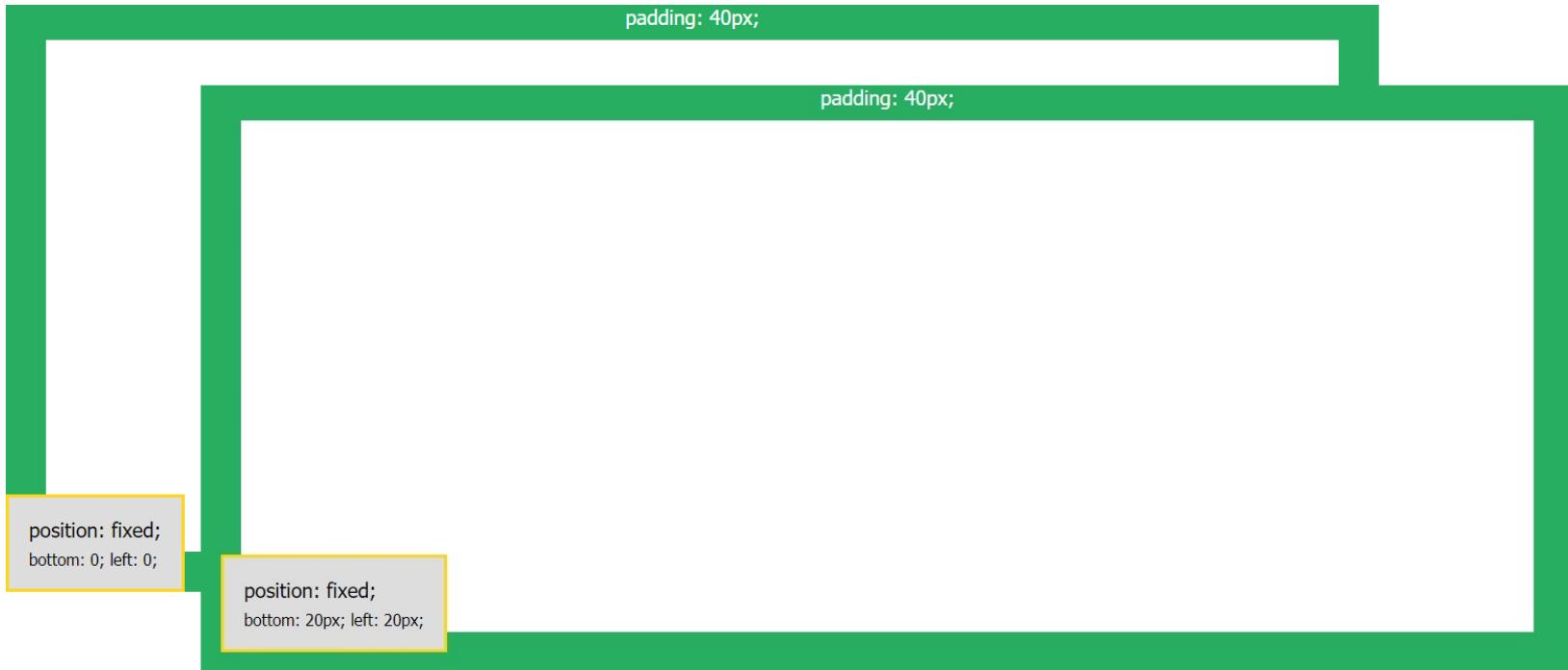
Когда к элементу применяются свойства `top` и `right` со значением `0`, то он отображается в левом правом углу. При значении `20px`, браузер добавляет отступ в `20px` по осям X и Y.

## 0 и 20px для bottom и right



Когда к элементу применяются свойства `top` и `right` со значением `0`, то он отображается в нижнем верхнем углу. При значении `20px`, браузер добавляет отступ в `20px` по осям X и Y.

# 0 и 20px для bottom и left



Когда к элементу применяются свойства `bottom` и `left` со значением `0`, то он отображается в нижнем левом углу. При значении `20px`, браузер добавляет отступ в `20px` по осям X и Y.

# 0 для top , right , bottom И left

```
position: fixed;  
top: 0; right: 0; bottom: 0; left: 0;
```

Когда мы задаем значения сразу со всех сторон, то элемент начинается растягиваться сразу со всех краев.

# 40px для top , right , bottom и left

```
padding: 40px;
```

```
position: fixed;  
top: 40px; right: 40px; bottom: 40px; left: 40px;
```

При ином значении (например, 40px), браузер сделает отступы со всех границ элемента body и на оставшееся пространство растянет элемент.

# РОДИТЕЛЬСКИЙ ЭЛЕМЕНТ position: relative И ДОЧЕРНИЙ ЭЛЕМЕНТ position: fixed

```
1 <div class="parent">
2   <div class="child">position: fixed;</div>
3 </div>
```

```
1 body {
2   margin: 0;
3   padding: 40px;
4 }
5
6 .box {
7   position: fixed;
8 }
```

# РОДИТЕЛЬСКИЙ ЭЛЕМЕНТ position: relative И ДОЧЕРНИЙ ЭЛЕМЕНТ position: fixed

Родительский контейнер с position: relative;



# top: 0 И top: 20px

```
position: fixed;  
top: 0;
```

Родительский контейнер с position: relative;

```
padding: 40px;
```



```
position: fixed;  
top: 20px;
```

Родительский контейнер с position: relative;

```
padding: 40px;
```



# right: 0 И right: 20px

Родительский контейнер с position: relative;



Родительский контейнер с position: relative;



# bottom: 0 И bottom: 20px

Родительский контейнер с position: relative;

padding: 40px;



position: fixed;  
bottom: 0;

Родительский контейнер с position: relative;

padding: 40px;



position: fixed;  
bottom: 20px;

# left: 0 и left: 20px

Родительский контейнер с position: relative;



Родительский контейнер с position: relative;



# z-index И position: fixed

Посмотрим как свойство `z-index` работает и с `position: fixed`.

Пример без `z-index`

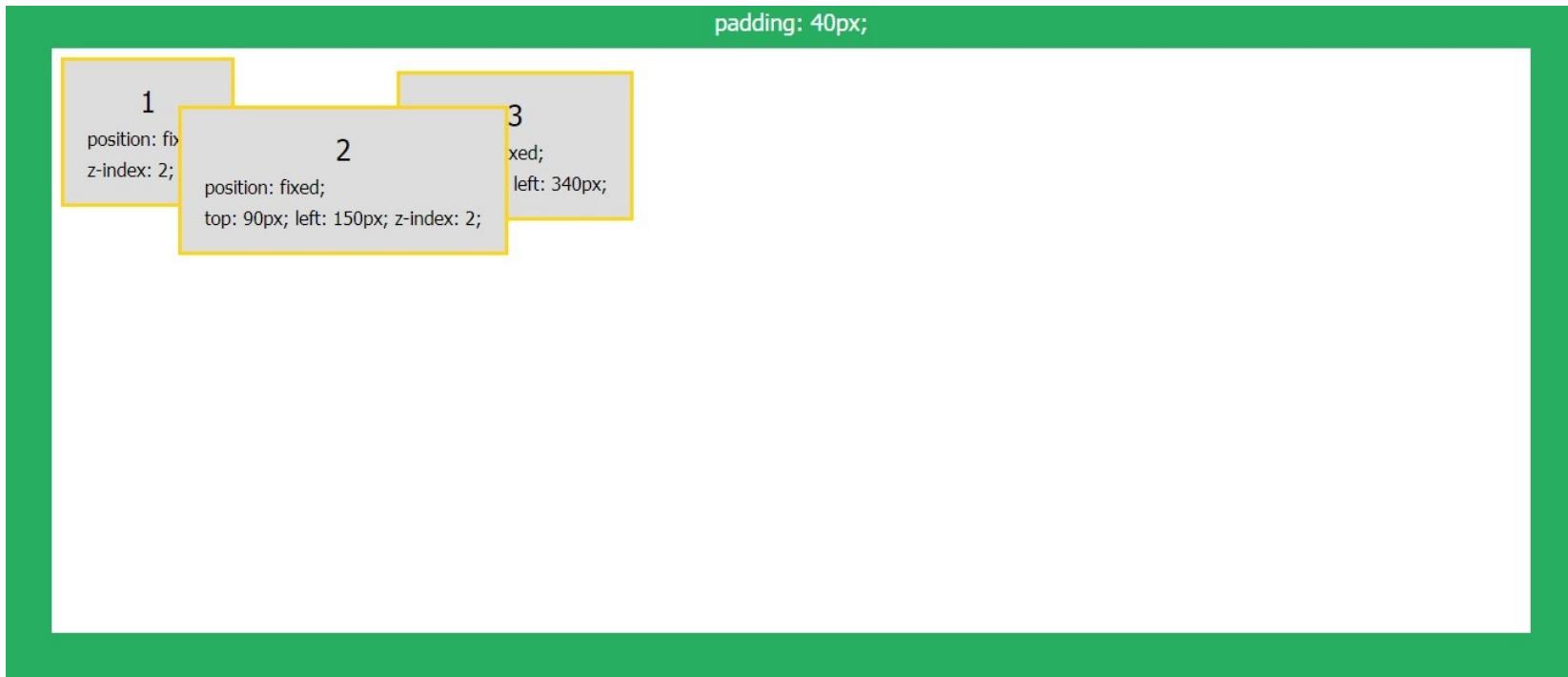


# z-index УСТАНОВЛЕНО



Первый элемент имеет свойство `z-index` со значением 2 и будет поверх и второго элемента (показано на изображении) и третьего, если они пересекутся.

# z-index для нескольких элементов с одинаковым значением



Первый и второй элемент имеют свойство `z-index` со значением 2, и поэтому они будут перекрывать третий элемент. Второй элемент идет после первого в HTML, и он будет перекрывать первый.

# ЗАМЕНИМ ЗНАЧЕНИЕ ABSOLUTE НА FIXED

```
1 .popup {  
2     position: fixed;  
3     top: 0;  
4     right: 0;  
5     bottom: 0;  
6     left: 0;  
7     z-index: 9999;  
8  
9     display: flex;  
10    justify-content: center;  
11    align-items: center;  
12}
```

# display: none

- Размеры элемента становятся `0` на `0` по ширине и высоте, и это нельзя изменить;
- Элемент перестает отображаться;
- Если элемент с `display: none` имеет дочерние интерактивные элементы, они будут недоступны с клавиатуры.

# СКРОЕМ ЭЛЕМЕНТ

```
1 .popup {  
2     position: fixed;  
3     top: 0;  
4     right: 0;  
5     bottom: 0;  
6     left: 0;  
7     z-index: 9999;  
8  
9     display: flex;  
10    justify-content: center;  
11    align-items: center;  
12 }
```

У элемента уже установлено свойство `display`.

[Демо](#)

# СКРОЕМ ЭЛЕМЕНТ

```
1 .popup {  
2     position: fixed;  
3     top: 0;  
4     right: 0;  
5     bottom: 0;  
6     left: 0;  
7     z-index: 9999;  
8  
9     display: none;  
10 }  
11  
12 .popup-active {  
13     display: flex;  
14     justify-content: center;  
15     align-items: center;  
16 }
```

---

# СПЕЦИФИЧНОСТЬ CSS-СЕЛЕКТОРОВ

# НАША ЗАДАЧА

**ВОЙТИ ЧЕРЕЗ ЭЛЕКТРОННУЮ ПОЧТУ**

АДРЕС ЭЛЕКТРОННОЙ ПОЧТЫ

ПАРОЛЬ

войти

**ВОЙТИ ЧЕРЕЗ...**

 FACEBOOK

 TWITTER

 ВКОНТАКТЕ

Демо

# НАША ЗАДАЧА

**ВОЙТИ ЧЕРЕЗ ЭЛЕКТРОННУЮ ПОЧТУ**

АДРЕС ЭЛЕКТРОННОЙ ПОЧТЫ

ПАРОЛЬ

войти

**ВОЙТИ ЧЕРЕЗ...**

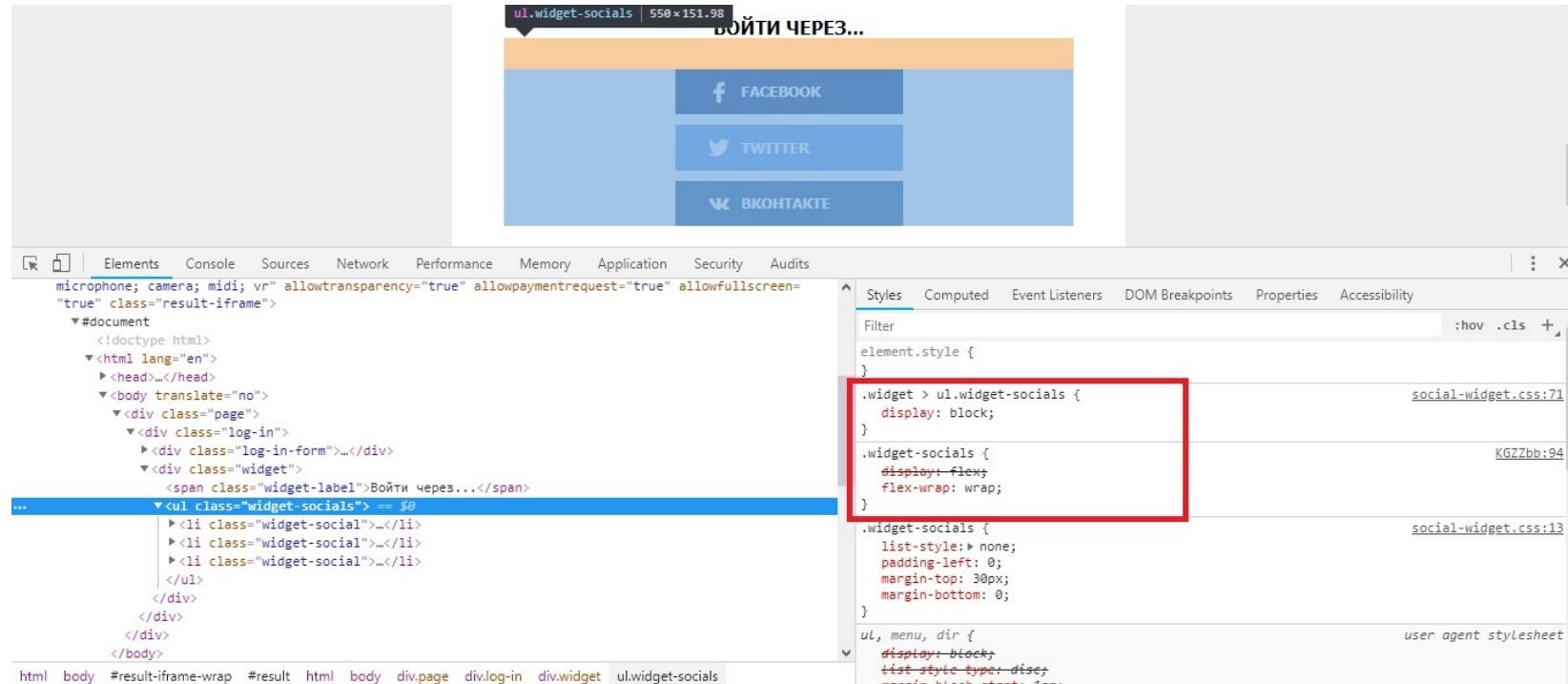


## ДОБАВИМ `display: flex`

```
1 .widget-socials {  
2   display: flex;  
3   flex-wrap: wrap;  
4 }
```

[Демо](#)

# ИНСПЕКТИРУЕМ



CSS-правило, которое мы написали, почему-то располагается под правилом из файла виджета.

# ПОСМОТРИМ ПРИМЕР

```
<span id="slogan" class="slogan" style="font-size: 20px;">Верстка это просто!</span>
```

```
1  span {
2      font-size: 10px;
3  }
4
5  .slogan {
6      font-size: 40px;
7  }
8
9  #slogan {
10     font-size: 30px;
11 }
```

# СПЕЦИФИЧНОСТЬ CSS-СЕЛЕКТОРА

Специфичность CSS-селектора - это коэффициент, по которому браузер понимает, какой селектор важнее в случае, когда к элементу срабатывают несколько селекторов.

0

**style**

0

**id**

0

**class, псевдокласс**

0

**элементы**

# СПЕЦИФИЧНОСТЬ CSS-СЕЛЕКТОРА

```
span { font-size: 10px; }
```

0

style

0

id

0

class, псевдокласс

1

элементы

Для селектора `span` специфичность будет `0001`, потому что в этом селекторе находится 1 элемент `span`.

# СПЕЦИФИЧНОСТЬ CSS-СЕЛЕКТОРА

.slogan { font-size: 40px; }

0

style

0

id

1

class, псевдокласс

0

элементы

Для селектора `.slogan` специфичность будет `0010`, потому что в этом селекторе находится 1 элемент с атрибутом `class` и значением `slogan`.

# СПЕЦИФИЧНОСТЬ CSS-СЕЛЕКТОРА

```
#slogan { font-size: 30px; }
```

0

style

1

id

0

class, псевдокласс

0

элементы

Для селектора `#slogan` специфичность будет `0100`, потому что в этом селекторе находится 1 элемент с атрибутом `id` и значением `slogan`.

# СПЕЦИФИЧНОСТЬ CSS-СЕЛЕКТОРА

<span style="font-size: 20px;">

**1**

**style**

**0**

**id**

**0**

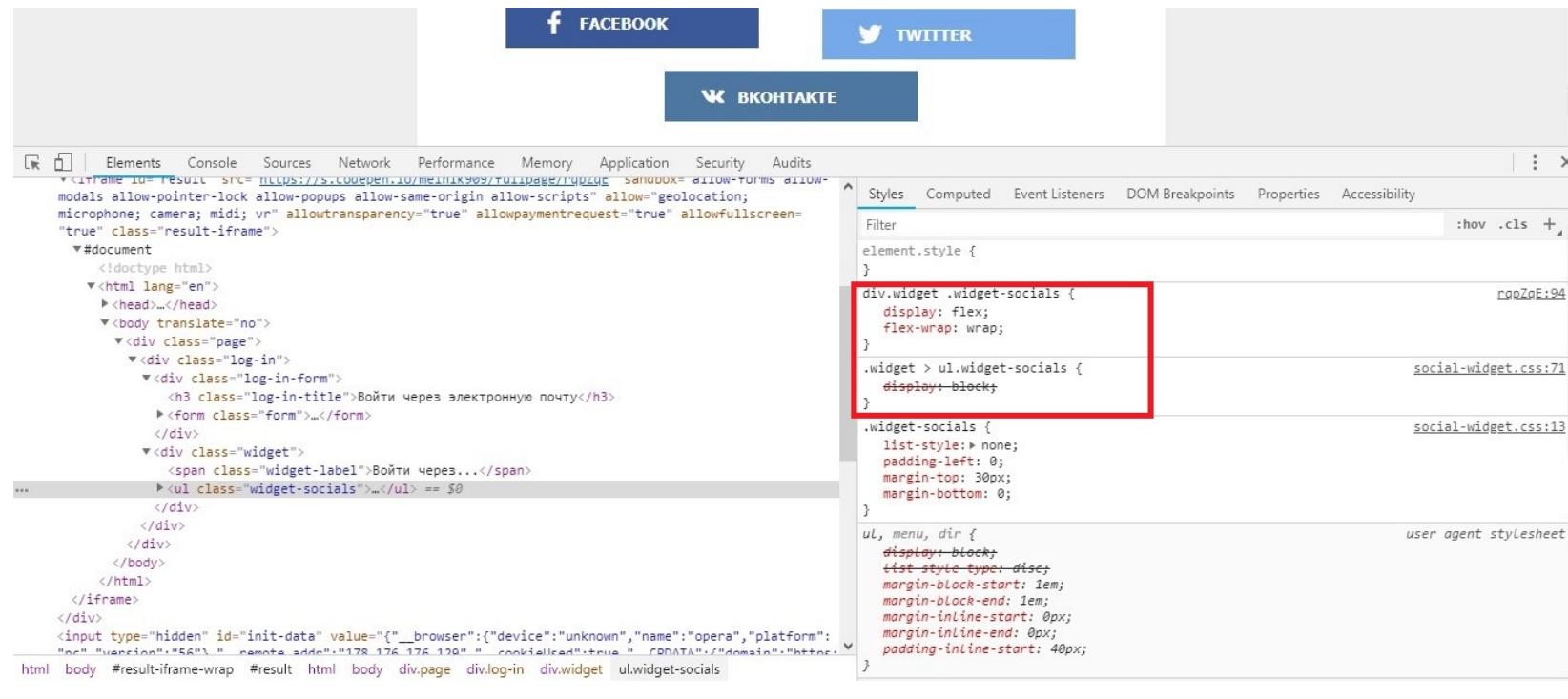
**class, псевдокласс**

**0**

**элементы**

Для атрибута `style` специфичность будет **1000**, потому что у элемента есть атрибут `style`.

# ПОСМОТРИМ НА СЕЛЕКТОРЫ



The screenshot shows a browser window with three social media links at the top: Facebook, Twitter, and VKontakte. Below the links is a developer tools interface. The left pane shows the DOM tree with various elements like <html>, <body>, and <div> with classes like "widget-socials". The right pane shows the "Styles" tab of the developer tools, displaying CSS rules. A red box highlights a section of rules from "social-widget.css":

```

div.widget .widget-socials {
    display: flex;
    flex-wrap: wrap;
}

.widget > ul.widget-socials {
    display: block;
}

.widget-socials {
    list-style-type: none;
    padding-left: 0;
    margin-top: 30px;
    margin-bottom: 0;
}

```

The bottom part of the screenshot shows the corresponding CSS code in green:

```

.widget > ul.widget-socials { }

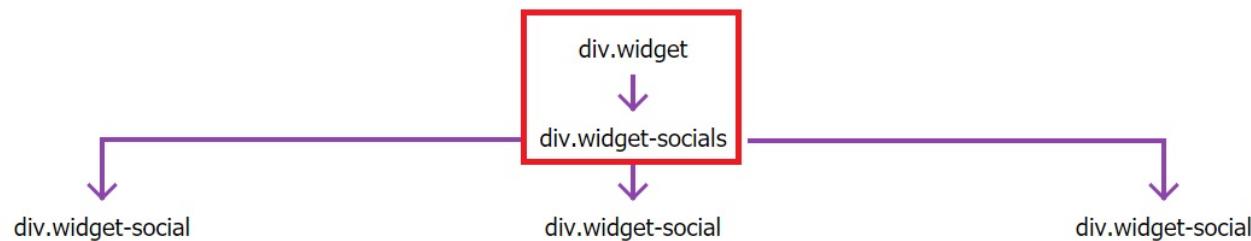
.widget-socials { }

```

[Демо](#)

# ДОЧЕРНИЙ СЕЛЕКТОР

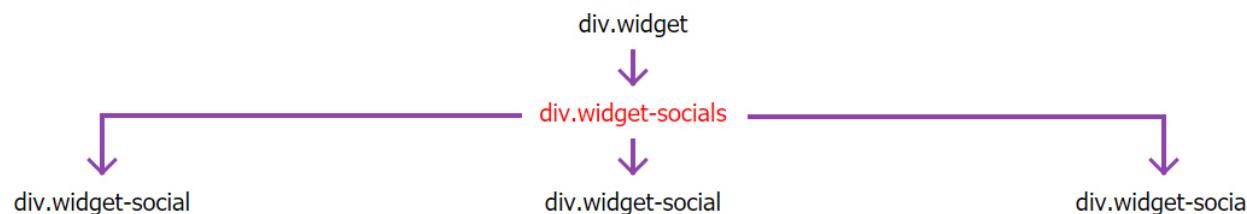
Дочерним селектором называют группу из двух элементов, когда один из них является родителем, а другой ребенком. Например, в нашем коде есть пара элементов `div.widget` и `div.widget-socials`.



# СТИЛИЗУЕМ

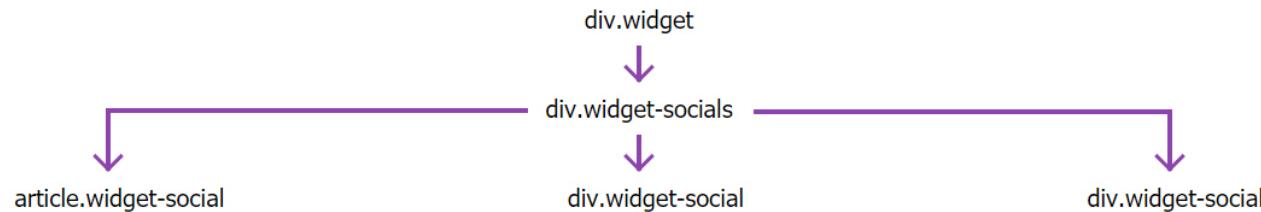
Элемент `div.widget-socials` является ребенком `div.widget`, мы можем воспользоваться дочерним селектором. Для этого между двумя элементами надо поставить знак `>`.

```
1 | div.widget > div.widget-socials {  
2 |   color: #ff0000;  
3 | }
```



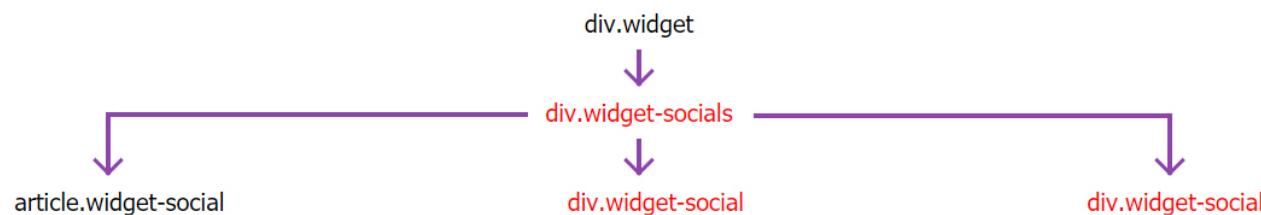
# ВЛОЖЕННЫЙ СЕЛЕКТОР

Вложенным селектором называют группу из нескольких элементов, когда один из них является родителем, а другие потомками. В нашем коде есть `div` с классом `widget`, в котором есть 3 элемента `div` и один `article`.



# СТИЛИЗУЕМ ЭЛЕМЕНТЫ `div`

```
1 | .widget div {  
2 |   color: #ff0000;  
3 | }
```

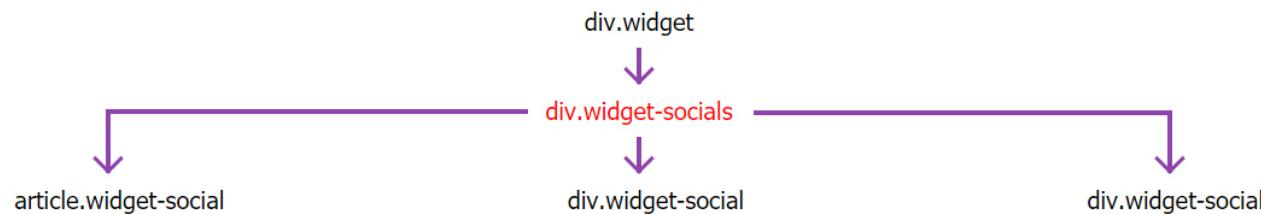


Браузер найдет все элементы `div`, которые являются потомками элемента с классом `widget`.

# КЛЮЧЕВАЯ РАЗНИЦА

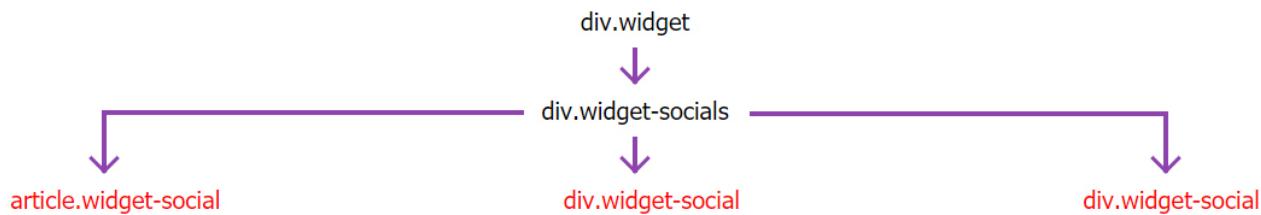
Вложенным селектором браузер выбирает все дочерние элементы родителя, а дочерним селектором только те, которые находится сразу внутри родителя.

```
1 | .widget > div {  
2 |   color: #ff0000;  
3 | }
```



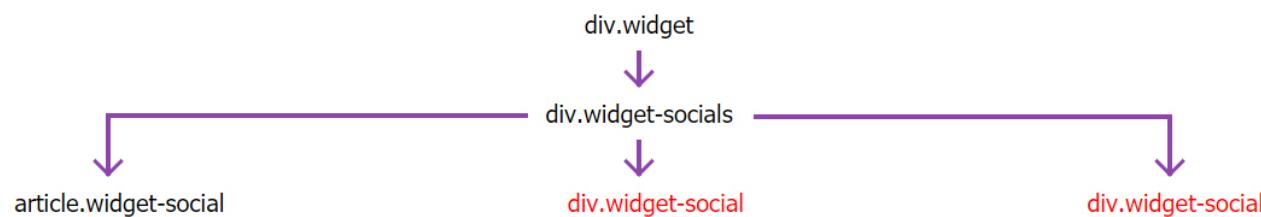
# КОМБИНАЦИЯ СЕЛЕКТОРА ПО КЛАССУ И СЕЛЕКТОРА ПО ТЕГУ

```
1 .widget-social {  
2   color: #ff0000;  
3 }
```



# КОМБИНАЦИЯ СЕЛЕКТОРОВ

```
1 | div.widget-social {  
2 |   color: #ff0000;  
3 | }
```

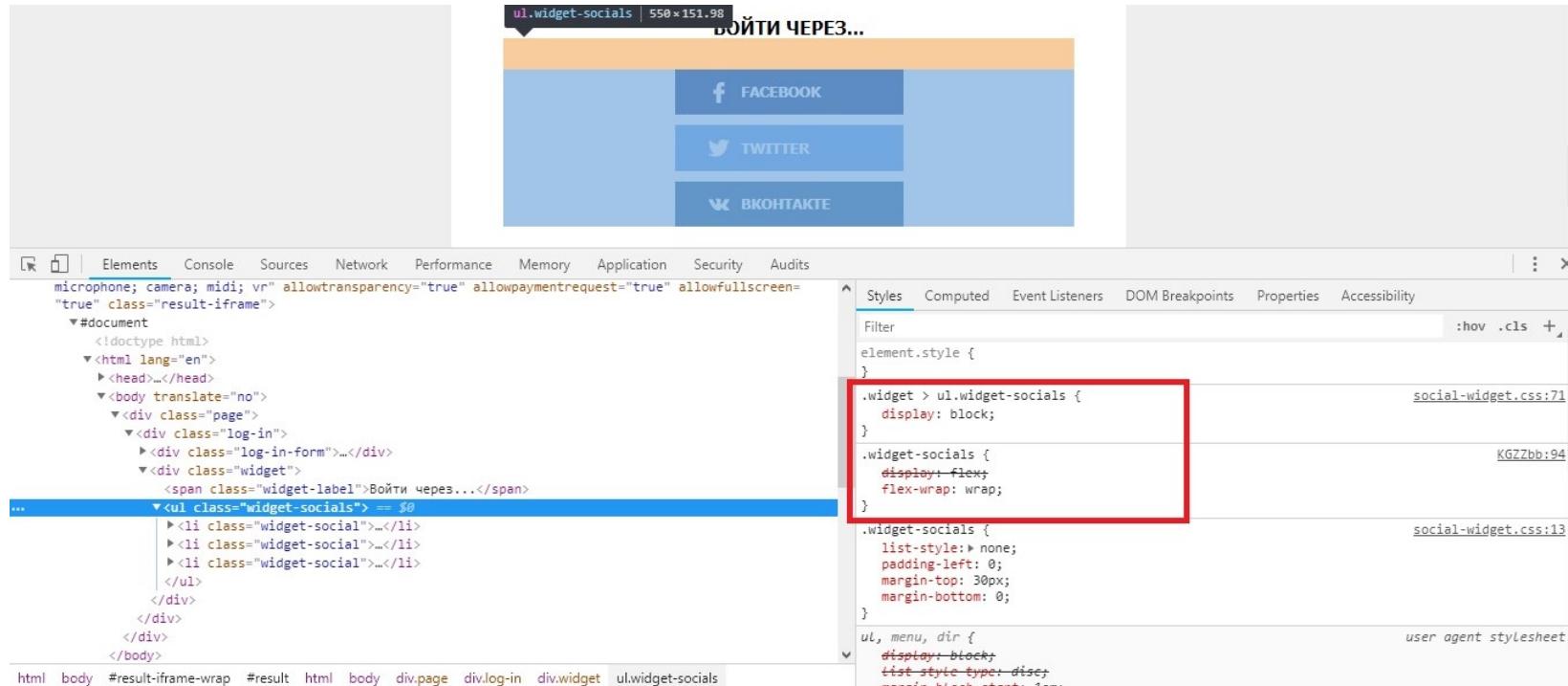


# ПОСЧИТАЕМ СПЕЦИФИЧНОСТЬ СЕЛЕКТОРОВ

```
1 | .widget > ul.widget-socials { } /* 0021 */
2 | .widget-socials { } /* 0010 */
```

В первом селекторе 2 класса ( `widget` и `widget-socials` ) и один элемент `ul`. У второго селектора 1 класс.

# СОРТИРОВКА ПРАВИЛ В ИНСПЕКТОРЕ



[Демо](#)

# ПОВЫСИМ СПЕЦИФИЧНОСТЬ

```
1 | .widget > ul.widget-socials { } /* 0021 */
2 | div.widget > .widget-socials { } /* 0021 */
```

ВОЙТИ ЧЕРЕЗ...

f FACEBOOK

TWITTER

vk ВКОНТАКТЕ

[Демо](#)

# ПРОВЕРИМ В ИНСПЕКТОРЕ



У нас же теперь два селектора имеют одинаковую специфичность. Почему выиграл наш?

# ПРИОРИТЕТНОСТЬ ПРИ РАВНОЙ СПЕЦИФИЧНОСТИ

```
1 | <span class="slogan">Верстка это просто!</span>
2 | <span class="slogan text">А специфичность еще проще!</span>
```

```
1 | /* 0010 */
2 | .slogan {
3 |   font-size: 40px;
4 |
5 |
6 | /* 0010 */
7 | .text {
8 |   font-size: 20px;
9 | }
```

# РЕЗУЛЬТАТ

Верстка это просто! А специфичность еще проще!

The screenshot shows the developer tools of a web browser with the 'Styles' tab selected. Two CSS rules are highlighted with a red box:

```
.text {  
    font-size: 20px;  
}  
.slogan {  
    font-size: 40px;  
}
```

The 'Computed' tab shows the final values for the selected element, with the 'padding' property highlighted with a green box:

```
margin -  
border -  
padding -  
auto x auto -  
- - - - -
```

The bottom of the interface shows the element tree with the selected node highlighted in blue:

```
html body.fullpage.logged-in div#result-iframe-wrap iframe#result.result-iframe html body span.slogan.text
```

# ПОМЕНЯЕМ ПОРЯДОК

```
1  /* 0010 */  
2  .text {  
3      font-size: 20px;  
4  }  
5  
6  /* 0010 */  
7  .slogan {  
8      font-size: 40px;  
9  }
```

# РЕЗУЛЬТАТ

Верстка это просто! А специфичность еще проще!

The screenshot shows the Chrome DevTools interface. The left pane displays the DOM tree with various HTML elements and their classes. The right pane shows the Styles tab of the DevTools, which lists CSS rules for the selected element. A red box highlights the rule `.slogan { font-size: 40px; }`. Below this, another rule `.text { font-size: 20px; }` has a checkbox next to it, indicating it is being overridden. To the right of the styles, there is a visual representation of the element's bounding box with nested boxes for margin, border, padding, and content, labeled with their respective properties.

```
<!DOCTYPE html>
<!--[if lt IE 9]>
<html lang="en" class="oldie">
<![endif]-->
<!--[if gt IE 9]><!--
<html lang="en">
<!--<![endif]-->
<head></head>
<body class="fullpage logged-in">
  <svg xmlns="http://www.w3.org/2000/svg" width="0" height="0" display="none">...</svg>
  <header class="main-header" id="main-header">...</header>
  <div class="oldie-header">...</div>
  <div id="result-iframe-wrap" role="main">
    <iframe id="result" src="https://s.codepen.io/melnik909/fullpage/QZaYMR" sandbox="allow-forms allow-modals allow-pointer-lock allow-popups allow-same-origin allow-scripts" allow="geolocation; microphone; camera; midi; vr" allowtransparency="true" allowpaymentrequest="true" allowfullscreen="true" class="result-iframe">
      <#document
        <!DOCTYPE html>
        <html lang="en">
          <head>...</head>
          <body translate="no">
            <span class="slogan">Верстка это просто!</span>
            <span class="slogan text">А специфичность еще проще!</span>
          </body>
        </html>
      </#document>
    </iframe>
  </div>
</body>
</html>
```

Styles tab content:

- element.style { } QZaYMR:26
- .slogan { font-size: 40px; } QZaYMR:26
- .text { font-size: 20px; } QZaYMR:21

Element bounding box diagram:

- margin -
- border -
- padding -
- auto x auto

# СНОВА ПОСМОТРИМ В ИНСПЕКТОР

ВОЙТИ ЧЕРЕЗ...



FACEBOOK



TWITTER



ВКОНТАКТЕ

Из-за того что на сайте Codepen стили из вкладки CSS подключаются последними, то наш селектор и сработал.

# ПРОИНСПЕКТИРУЕМ КНОПКУ TWITTER

Посмотрим на его родительский элемент с классом `widget-social`

The screenshot shows a browser window with developer tools open. At the top, there's a toolbar with various tabs: Elements, Console, Sources, Network, Performance, Memory, Application, Security, and Audits. The Elements tab is selected. Below the toolbar, the main content area displays a login interface with three buttons: Facebook, Twitter, and VKontakte. The Twitter button is highlighted with a blue selection bar. Above the buttons, the text "ВОЙТИ ЧЕРЕЗ..." is visible. The developer tools sidebar on the right shows the DOM tree and the Styles panel. A red box highlights a specific CSS rule in the Styles panel:

```
.widget > ul.widget-socials .widget-social:nth-child(n+2) {  
    margin-top: 10px;  
}  
.widget > ul.widget-socials .widget-social {  
    width: 220px;  
    margin-left: auto;  
    margin-right: auto;  
}  
li {  
    display: list-item;  
    text-align: -webkit-match-parent;  
}  
Inherited from ul.widget-socials  
.widget-socials {  
    list-style-type: none;  
    padding-left: 0;  
    margin-top: 30px;  
    margin-bottom: 0;  
}
```

The CSS rule `.widget > ul.widget-socials .widget-social:nth-child(n+2)` is highlighted with a red box. This rule applies a `margin-top: 10px;` to the second child element of the `.widget-socials` list item. The `ul.widget-socials` selector is also highlighted with a red box. The `list-style-type: none;` rule is present in the inherited styles.

# ПОСЧИТАЕМ СПЕЦИФИЧНОСТЬ СЕЛЕКТОРА

```
1  /* 0031 */
2  .widget > ul.widget-socials .widget-social {
3    width: 220px;
4    margin-left: auto;
5    margin-right: auto;
6  }
7
8  /* 0041 */
9  .widget > ul.widget-socials .widget-social:nth-child(n+2) {
10   margin-top: 10px;
11 }
```

# НУЖНО ПЕРЕОПРЕДЕЛИТЬ СЕЛЕКТОРЫ

Для `.widget > ul.widget-socials .widget-social`

```
1 /* 0031 */
2 div.widget .widget-socials .widget-social {
3   width: 170px;
4   margin-left: 5px;
5   margin-right: 5px;
6 }
```

Для `.widget > ul.widget-socials .widget-social:nth-child(n+2)`

```
1 /* 0041 */
2 div.widget .widget-socials .widget-social:nth-child(n+2) {
3   margin-top: 0;
4 }
```

# РЕЗУЛЬТАТ

The screenshot shows a browser's developer tools with the "Elements" tab selected. Below it, the "Styles" tab is active, displaying a list of CSS rules. A red box highlights a specific rule for the second child of a ".widget-socials" element:

```
div.widget .widget-socials .widget-social:nth-child(n+2) {  
    margin-top: 0;  
}  
.widget > ul.widget-socials .widget-social:nth-child(n+2) {  
    margin-top: 10px;  
}  
div.widget .widget-socials .widget-social {  
    width: 170px;  
    margin-left: 5px;  
    margin-right: 5px;  
}  
.widget > ul.widget-socials .widget-social {  
    width: 220px;  
    margin-left: auto;  
    margin-right: auto;  
}  
li {  
    display: list-item;  
    text-align: -webkit-match-parent;  
},
```

The "Network" tab is also visible at the top of the developer tools.

# !important

Вернемся к примеру

```
<span id="slogan" class="slogan" style="font-size: 20px;">Верстка это просто!</span>
```

```
1  span {
2      font-size: 10px !important;
3  }
4
5  .slogan {
6      font-size: 40px;
7  }
8
9  #slogan {
10     font-size: 30px;
11 }
```

# РЕЗУЛЬТАТ

Верстка это просто!

The screenshot shows the Chrome DevTools interface. The left panel, 'Elements', displays the HTML structure of the page. The right panel, 'Styles', shows the CSS rules applied to the elements. A specific rule for a 'span' element is highlighted with a red box:

```
span {
  font-size: 10px !important;
}
```

Below the styles panel, a visual representation of the element's box model is shown with colored outlines: orange for margin, yellow for border, green for padding, and blue for the content area.

Elements panel (HTML structure):

```
<!DOCTYPE html>
<!--[if lte IE 9]>
<html lang="en" class="oldie">
<![endif]-->
<!--[if gt IE 9]><!-->
<html lang="en">
<!--<[endif]-->
<head></head>
<body class="fullpage logged-in">
  <svg xmlns="http://www.w3.org/2000/svg" width="0" height="0" display="none">...</svg>
  <header class="main-header" id="main-header">...</header>
  <div class="oldie-header">...</div>
  <div id="result-iframe-wrap" role="main">
    <iframe id="result" src="https://s.codepen.io/melnik909/fullpage/QZaYMR" sandbox="allow-forms allow-modals allow-pointer-lock allow-popups allow-same-origin allow-scripts" allow="geolocation; microphone; camera; midi; vr" allowtransparency="true" allowpaymentrequest="true" allowfullscreen="true" class="result-iframe">
      <#document
        <!doctype html>
        <html lang="en">
          <head></head>
          <body translate="no">
            <span id="slogan" class="slogan" style="font-size: 20px;">Верстка это просто!</span> == $0
          </body>
        </html>
      </#document>
    </iframe>
  </div>
</body>
```

Styles panel (CSS rules):

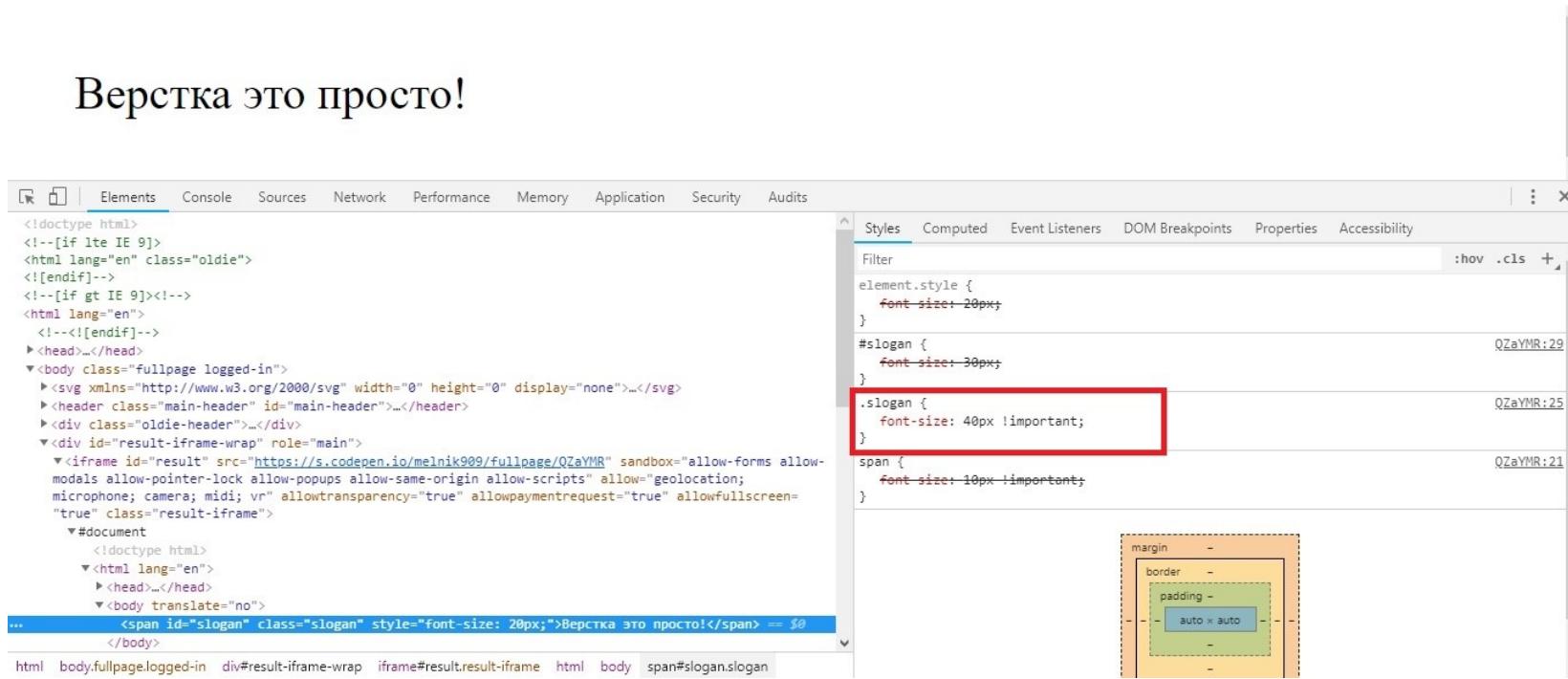
```
element.style {
  font-size: 20px;
}
#slogan {
  font-size: 30px;
}
.slogan {
  font-size: 40px;
}
span {
  font-size: 10px !important;
}
```

# МОЖНО ПЕРЕКРЫТЬ !important

```
1  span {  
2      font-size: 10px !important;  
3  }  
4  
5  .slogan {  
6      font-size: 40px !important;  
7  }  
8  
9  #slogan {  
10     font-size: 30px;  
11 }
```

# РЕЗУЛЬТАТ

Верстка это просто!



The screenshot shows the Chrome DevTools interface. The left pane displays the DOM tree with various CSS rules applied to elements. The right pane shows the 'Styles' tab where a specific style rule for '.slogan' is highlighted with a red box. This rule sets the font size to 40px !important. Below the styles, there is a visual representation of the element's bounding box with its margin, border, padding, and content areas.

```

<!DOCTYPE html>
<!--[if lt IE 9]>
<html lang="en" class="oldie">
<![endif]-->
<!--[if gt IE 9]><!-->
<html lang="en">
<!--<![endif]-->
<head>...</head>
<body class="fullpage logged-in">
  <svg xmlns="http://www.w3.org/2000/svg" width="0" height="0" display="none">...</svg>
  <header class="main-header" id="main-header">...</header>
  <div class="oldie-header">...</div>
  <div id="result-iframe-wrap" role="main">
    <iframe id="result" src="https://s.codepen.io/melnik909/fullpage/QZaYMR" sandbox="allow-forms allow-modals allow-pointer-lock allow-popups allow-same-origin allow-scripts" allow="geolocation; microphone; camera; midi; vr" allowtransparency="true" allowpaymentrequest="true" allowfullscreen="true" class="result-iframe">
      <#document
        <!DOCTYPE html>
        <html lang="en">
          <head>...</head>
          <body translate="no">
            <span id="slogan" class="slogan" style="font-size: 20px;">Верстка это просто!
          </body>
        </html>
      </#document>
    </iframe>
  </div>
</body>

```

Styles tab content (highlighted by a red box):

```

.slogan {
  font-size: 40px !important;
}

```

Такое определение приоритета значения свойства является неправильным!

---

# СЛЕДУЙТЕ ПРАВИЛАМ

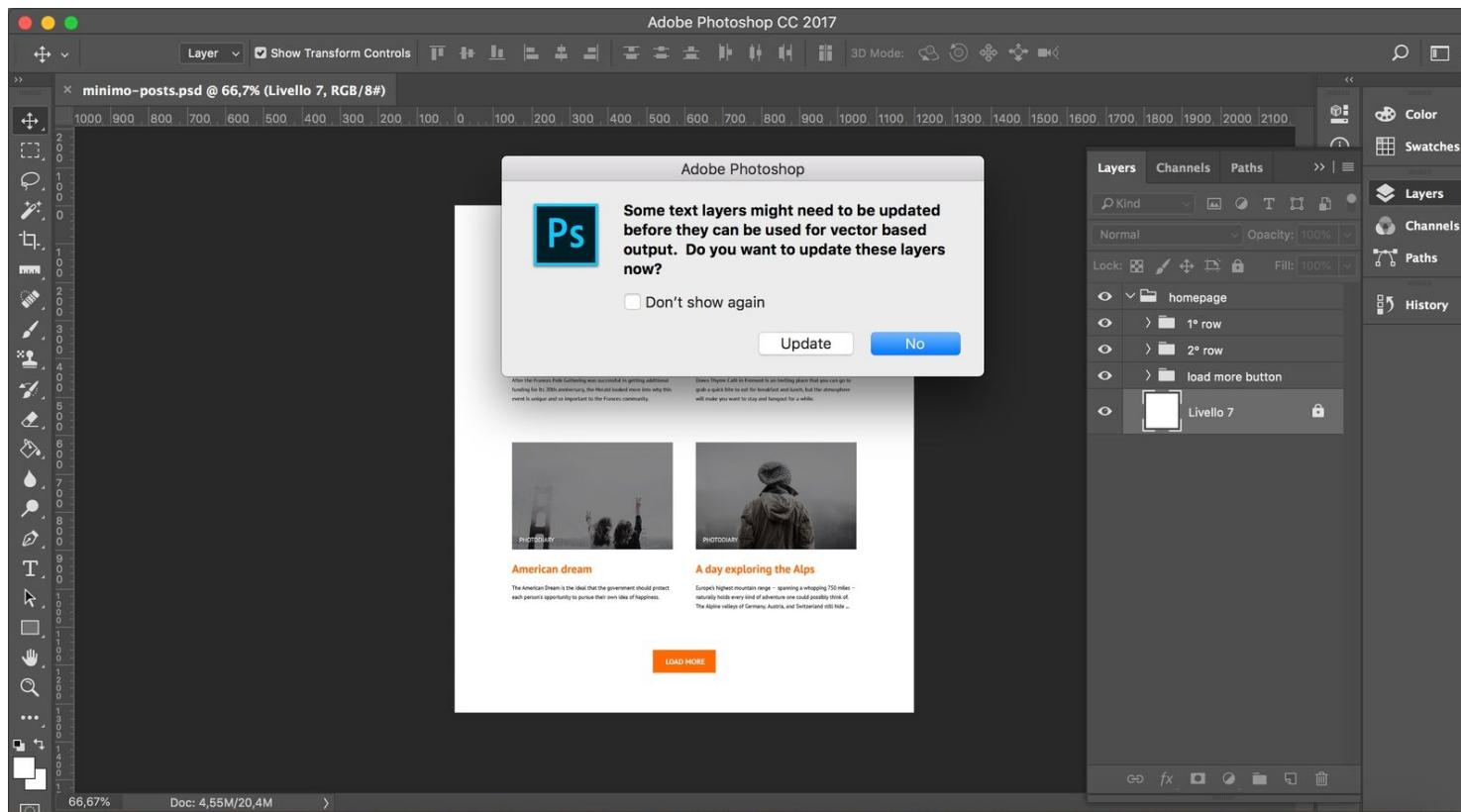
- Чтобы не делать ошибку, не используйте `!important` в своем коде;
- Не используйте селектор `#`;
- Вместо селектора `#` используйте классы.

---

# СВОЙСТВА ШРИФТА В ADOBE PHOTOSHOP

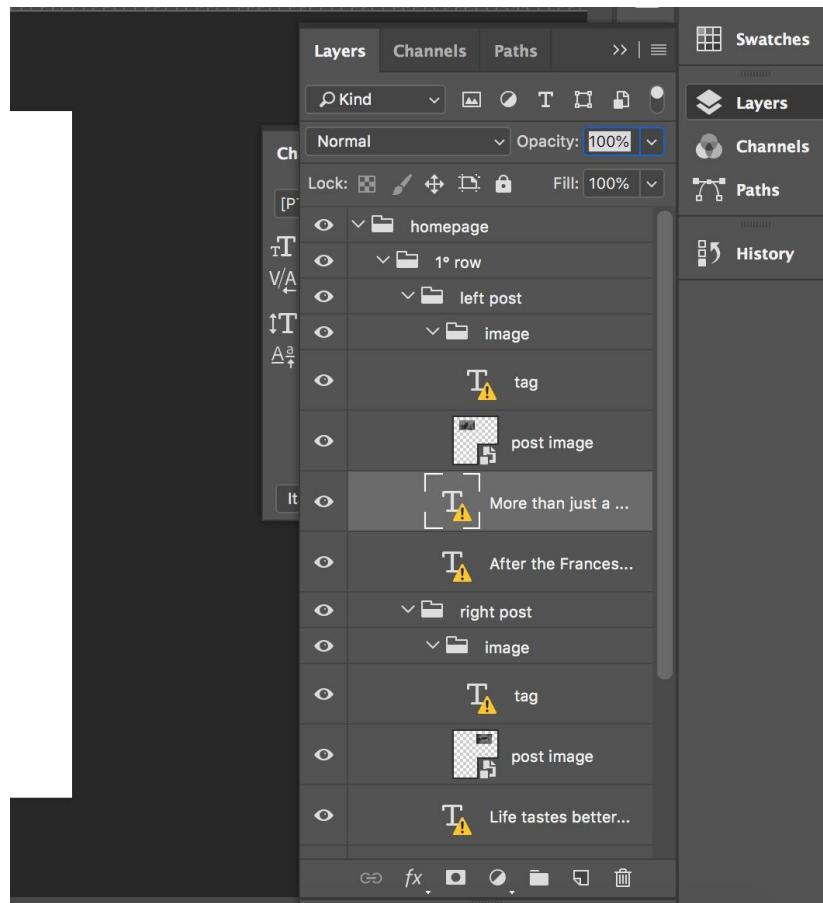
# ОКНО С ПРЕДУПРЕЖДЕНИЕМ

Когда мы открывали наш макет, у нас появлялось окно с предупреждением:



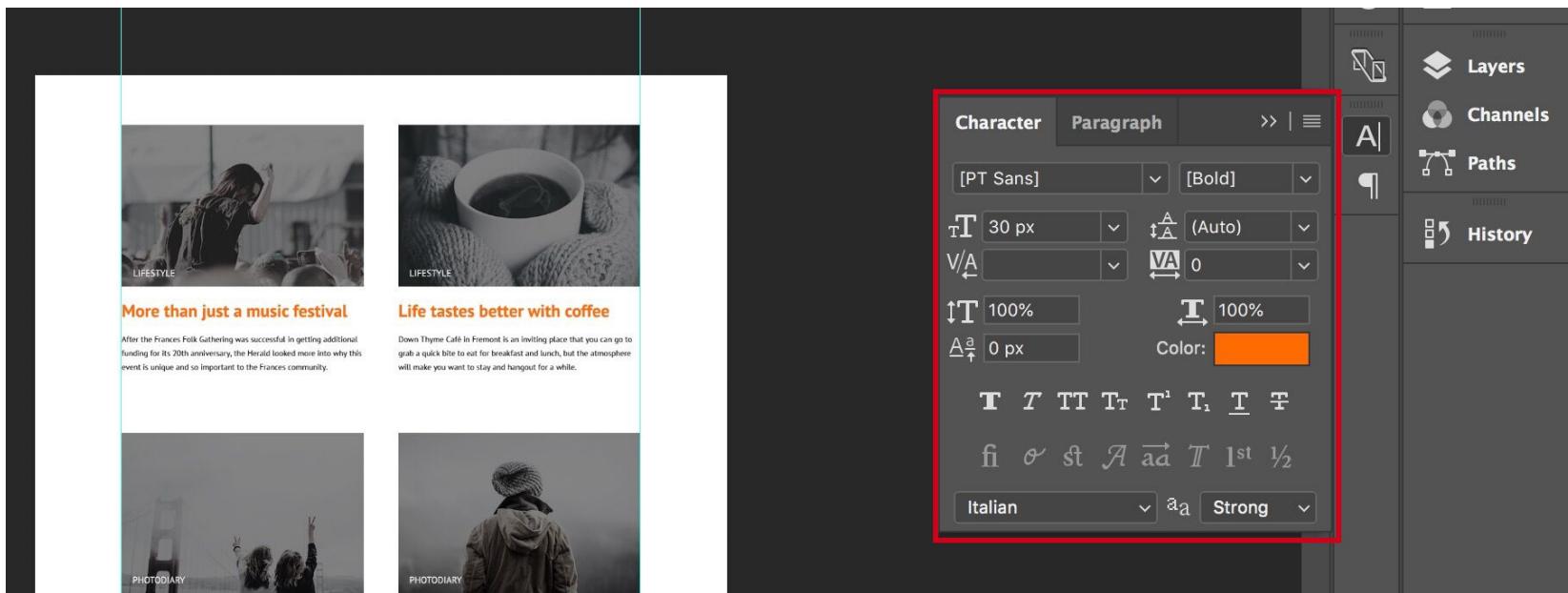
# ПОСМОТРИМ В ПАНЕЛЬ СЛОЕВ

Можно заметить, что все текстовые слои помечены значком ! в желтом треугольнике:

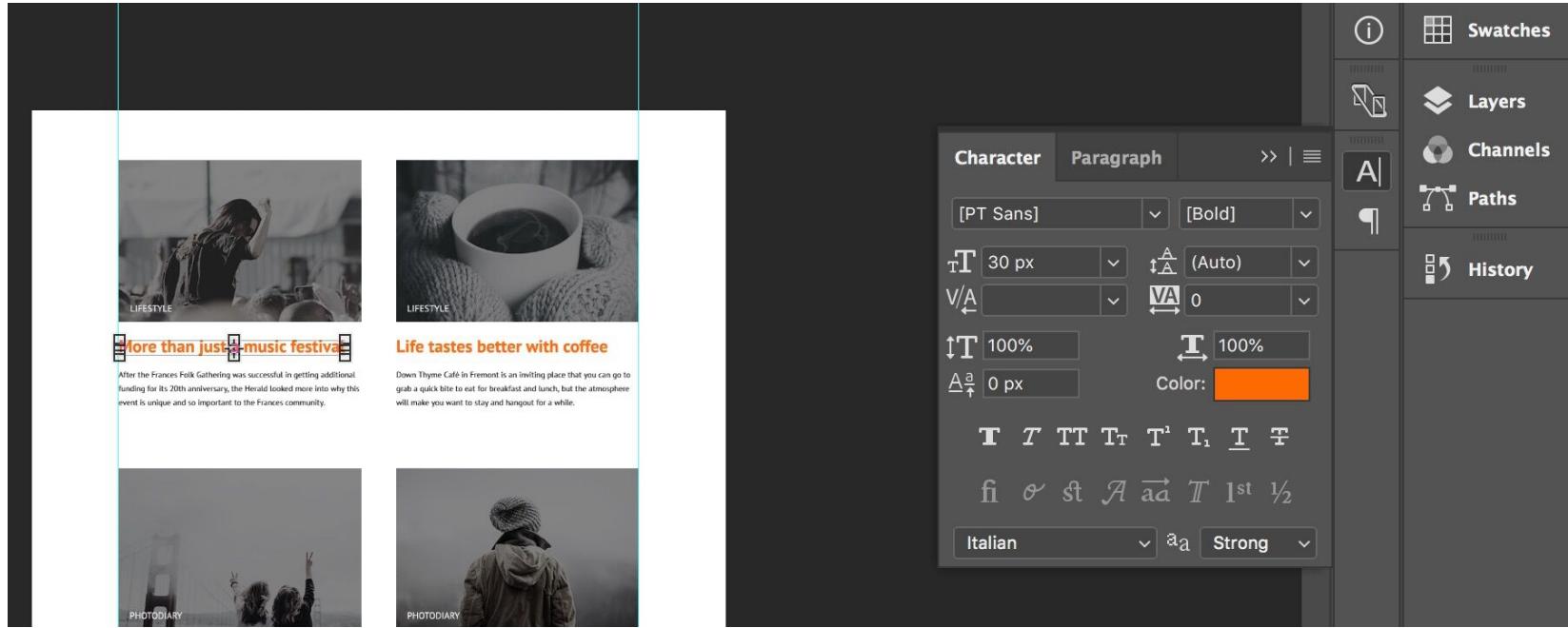


# ПАРАМЕТРЫ ШРИФТА

Чтобы посмотреть параметры шрифта надписи в макете, мы будем использовать панель **Character (Символ)**. Откроем панель с помощью меню **Window (Окно) - Character (Символ)**.

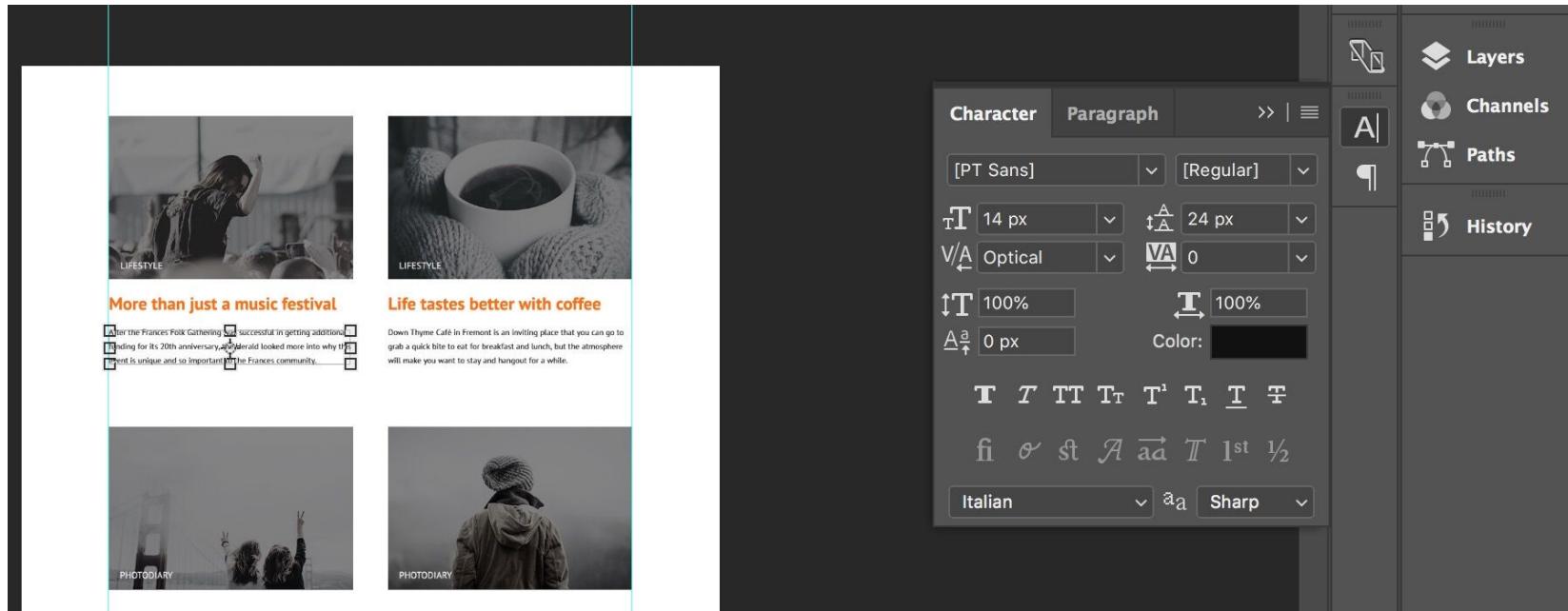


# ВЫБЕРЕМ СЛОЙ С ЗАГОЛОВКОМ



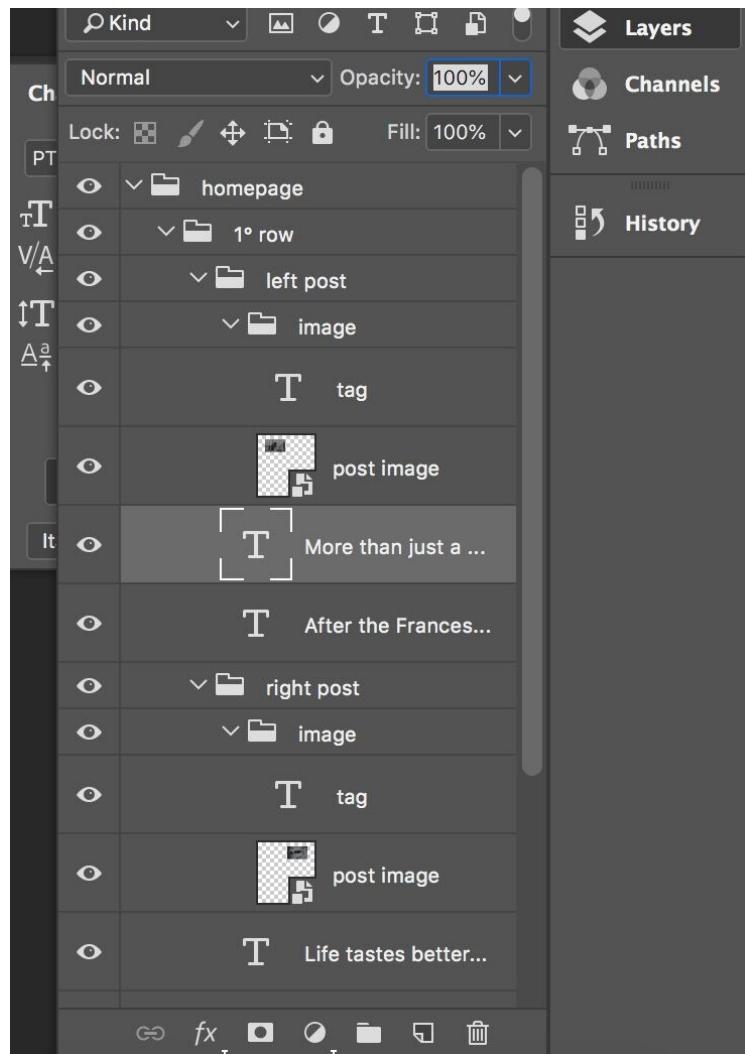
В левой верхней части мы видим надпись **[Pt Sans]**. Это название семейства шрифта заголовка. Чуть правее мы видим надпись **[Bold]** - это жирность шрифта.

# ВЫБЕРЕМ СЛОЙ С ТЕКСТОМ



Мы видим, что для краткого текста также используется семейство шрифта **[Pt Sans]**, но жирность уже **[Regular]**.

# ШРИФТ УСТАНОВЛЕН



# ПОДКЛЮЧЕНИЕ ШРИФТА

Для того, чтобы подключить шрифт, в CSS существует специальная конструкция `@font-face`. Она имеет следующий вид:

```
1 @font-face {  
2     font-family: Gentium;  
3     src: url(http://example.com/fonts/Gentium.woff);  
4     font-weight: 400;  
5     font-style: normal;  
6 }
```

После подключения шрифта его можно использовать на странице, например, указать для `body`:

```
1 body{  
2     font-family: Gentium;  
3 }
```

# РЕЗУЛЬТАТ

## Example Domain

This domain is established to be used for illustrative examples in documents.  
You may use this domain in examples without prior coordination or asking for  
permission.

[More information...](#)

После имени подключаемого шрифта нужно указать похожий безопасный  
шрифт:

```
1 | body{  
2 |   font-family: Gentium, Arial, sans-serif;  
3 | }
```

# ПОДКЛЮЧАЕМ ПЕРВЫЙ ШРИФТ

У нас в макете два шрифта: **Pt Sans** и **Pt Sans Bold**, скопируем их в папку проекта:

▼	css		Today at 11:35	--	Folder
	styles.css		Today at 11:35	Zero bytes	CSS
▼	fonts		Today at 11:32	--	Folder
	PTSans-Bold.woff		7 Oct 2018 at 12:50	145 KB	Document
	PTSans-Regular.woff		7 Oct 2018 at 12:50	140 KB	Document
	index.html		Today at 11:35	135 bytes	HTML

```
1 @font-face {  
2   font-family: 'Pt Sans';  
3   src: url(..../fonts/PTSans-Regular.woff);  
4   font-weight: 400;  
5   font-style: normal;  
6 }
```

# font-weight и font-style

С помощью свойств можно сообщить браузеру, какой именно файл шрифта нужно брать для блока.

Мы записали для первого шрифта:

```
1 | font-weight: 400;  
2 | font-style: normal;
```

# ПОДКЛЮЧИМ ВТОРОЙ ШРИФТ

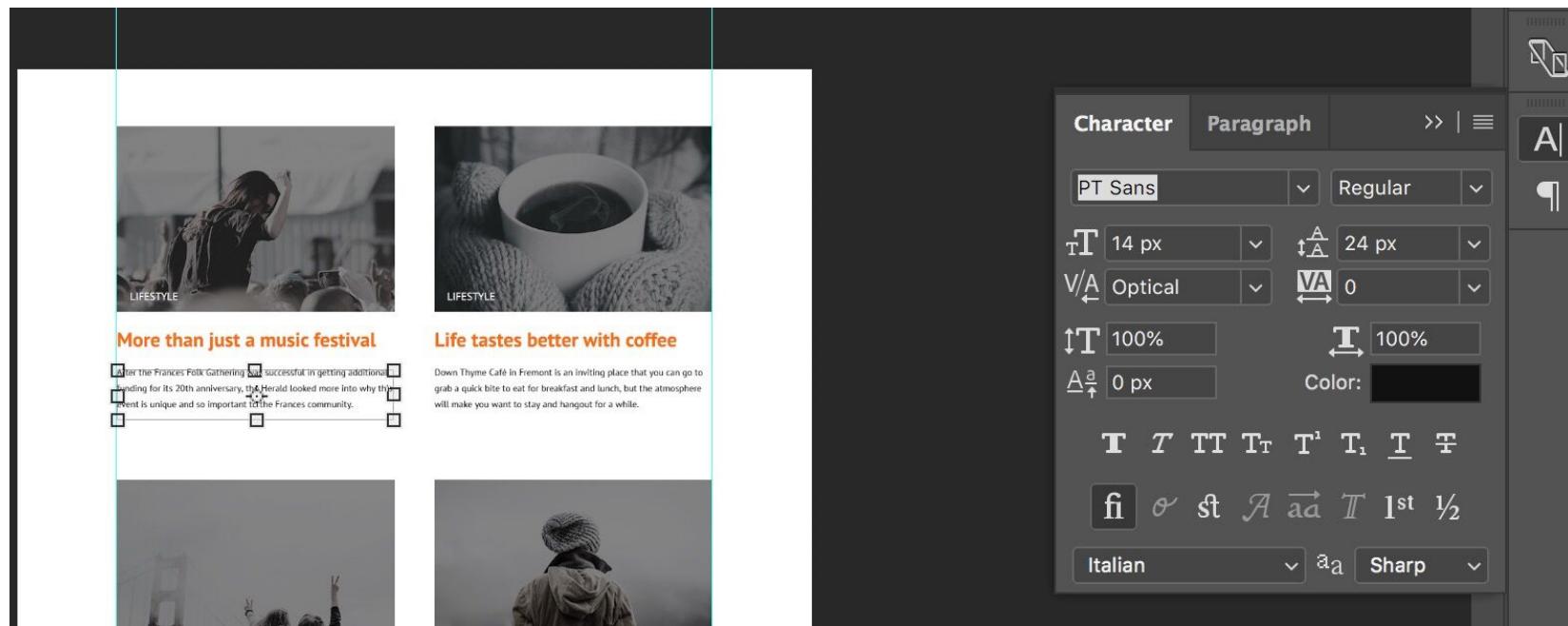
```
1 @font-face {  
2     font-family: 'Pt Sans';  
3     src: url(..../fonts/PTSans-Bold.woff);  
4     font-weight: 700;  
5     font-style: normal;  
6 }
```

Теперь мы можем использовать оба шрифта:

```
1 .last-posts{  
2     font-family: 'Pt Sans', Arial, sans-serif;  
3 }  
4 .last-post-title{  
5     font-weight: bold;  
6 }  
7 .load-more-posts{  
8     font-weight: bold;  
9 }
```

# РАЗМЕР ШРИФТА

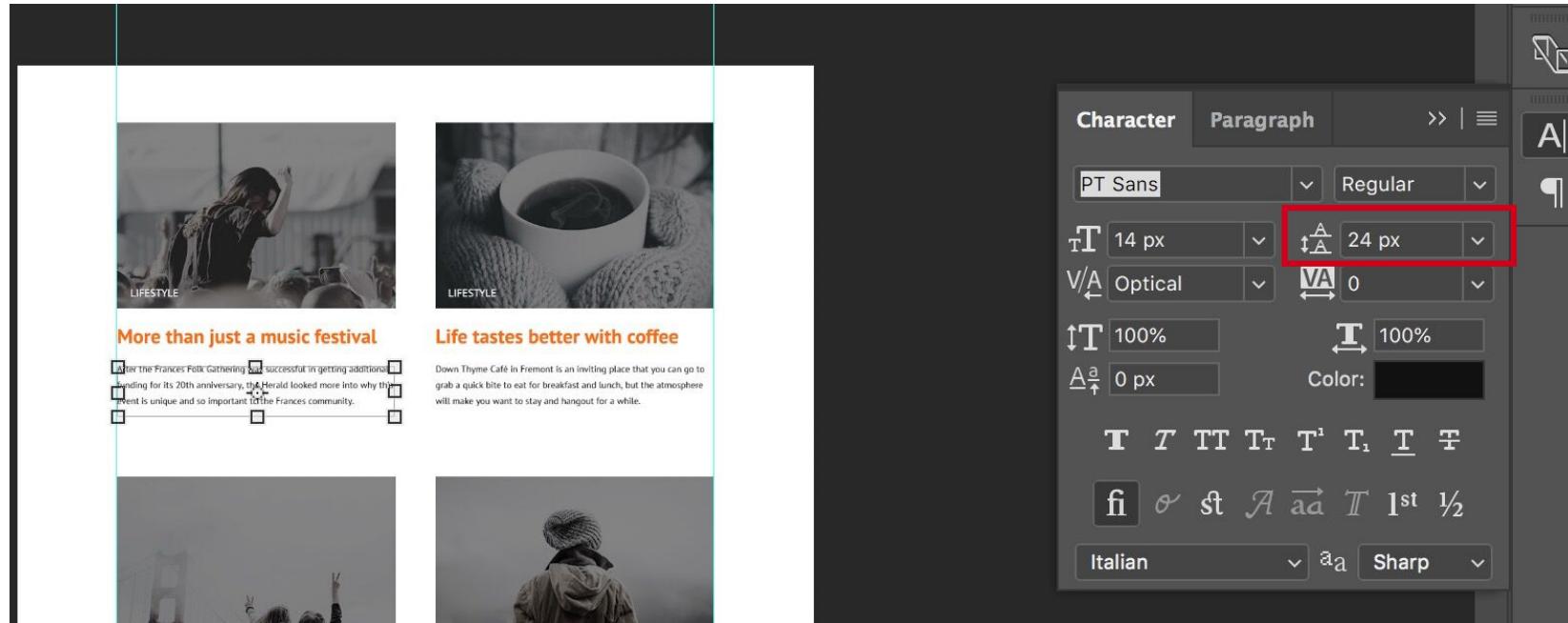
Выделим слой с кратким описанием поста и посмотрим в панель Character (Символ):



Мы видим символ **тТ**, и сразу после него указан размер шрифта данного слоя.

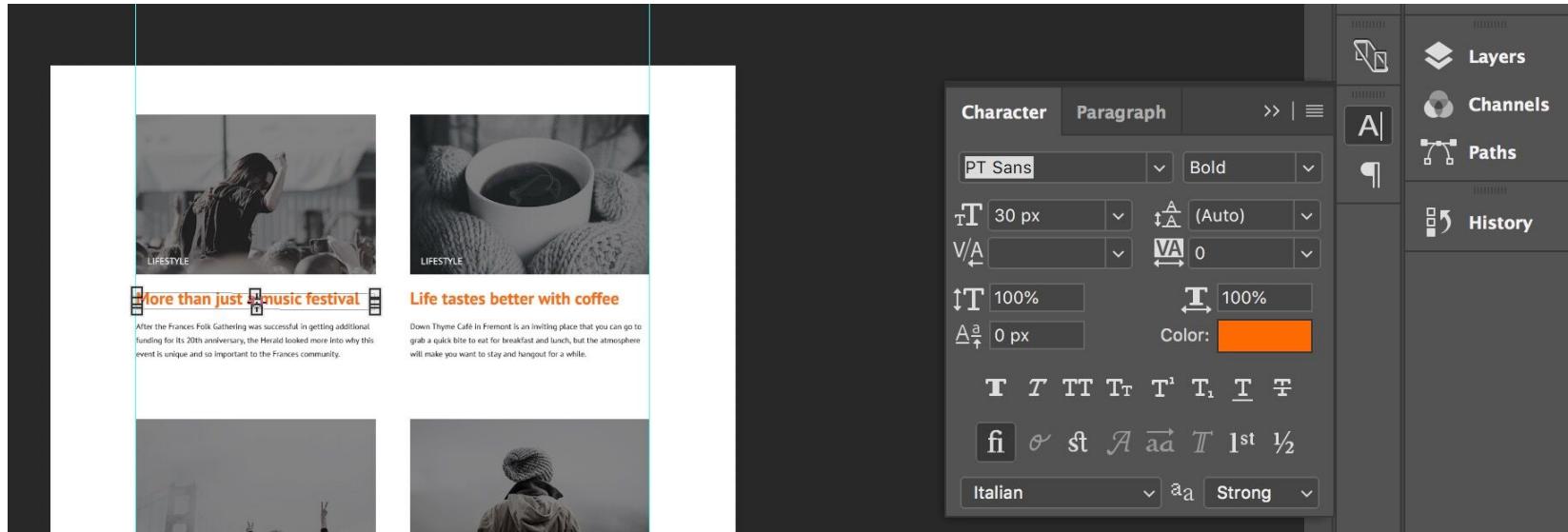
# ВЫСОТА СТРОКИ

В той же строке, что и размер шрифта, правее указана высота строки.



# ВЫСОТА СТРОКИ

А если мы посмотрим слой с заголовком, то увидим высоту строки `auto`:

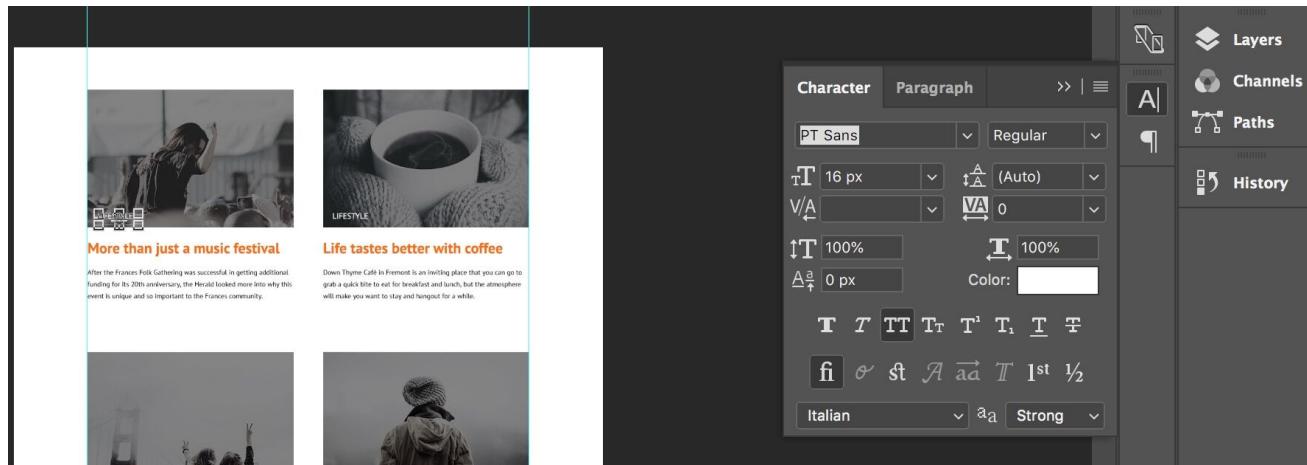


# СТИЛИЗУЕМ

```
1 .last-post-title {  
2   font-size: 30px;  
3   line-height: normal;  
4 }  
5 .last-post-description {  
6   font-size: 14px;  
7   line-height: 24px;  
8 }
```

# ТРАНСФОРМАЦИЯ ШРИФТА

Если мы выделим слой с тегом, то увидим «нажатой» кнопку в нижней части панели:



Таким образом дизайнер показал, что шрифт надписи в тегах должен выводиться заглавными буквами.

```

1 .last-post-tag {
2   font-size: 16px;
3   text-transform: uppercase;
4 }
```

---

# ЦВЕТ

---

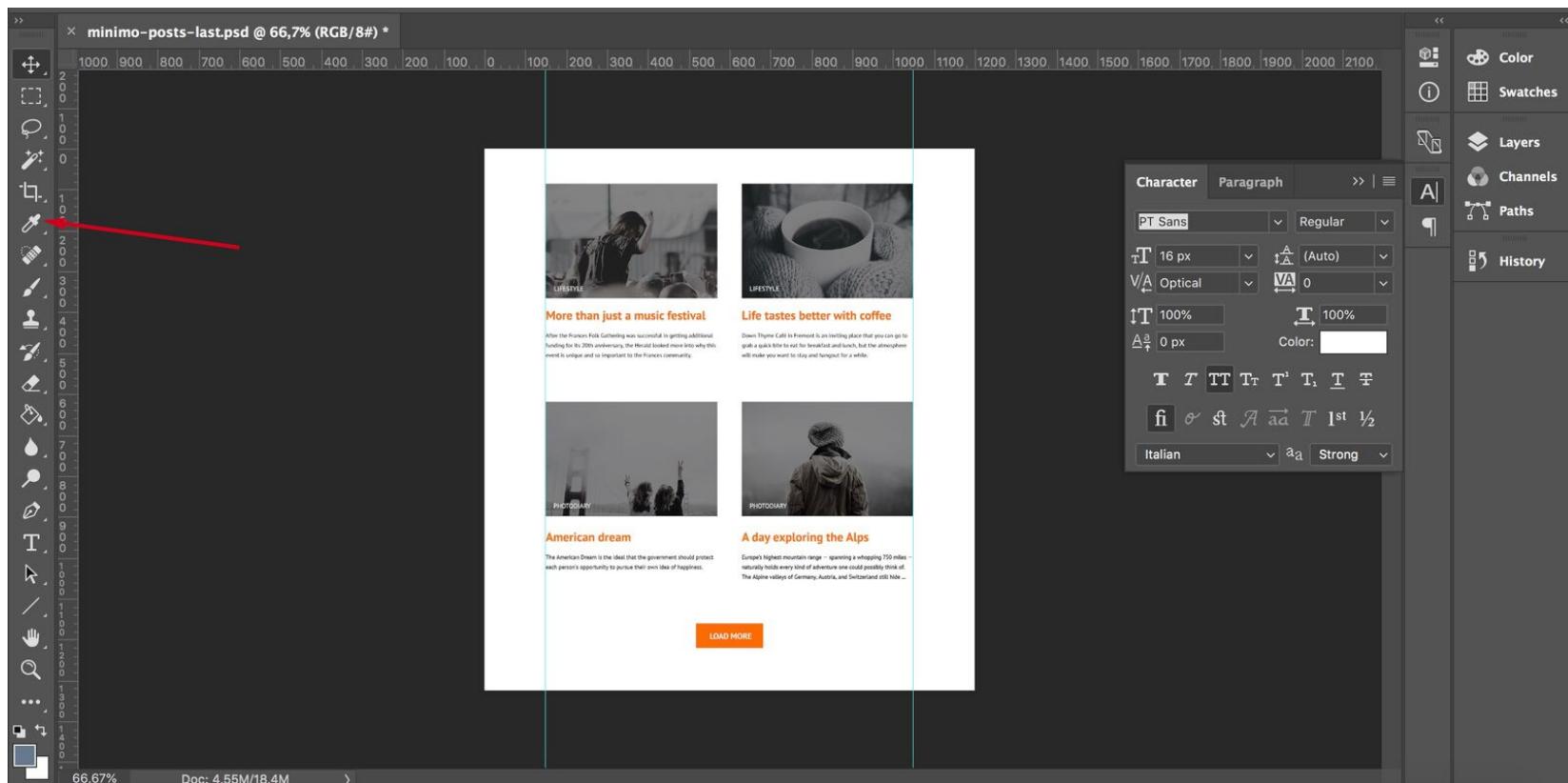
## ЦВЕТ СЛОЯ

Определить цвет слоя можно двумя способами:

- с помощью инструмента **Eyedropper tool (Пипетка)**;
- в панели **Info (Информация)**.

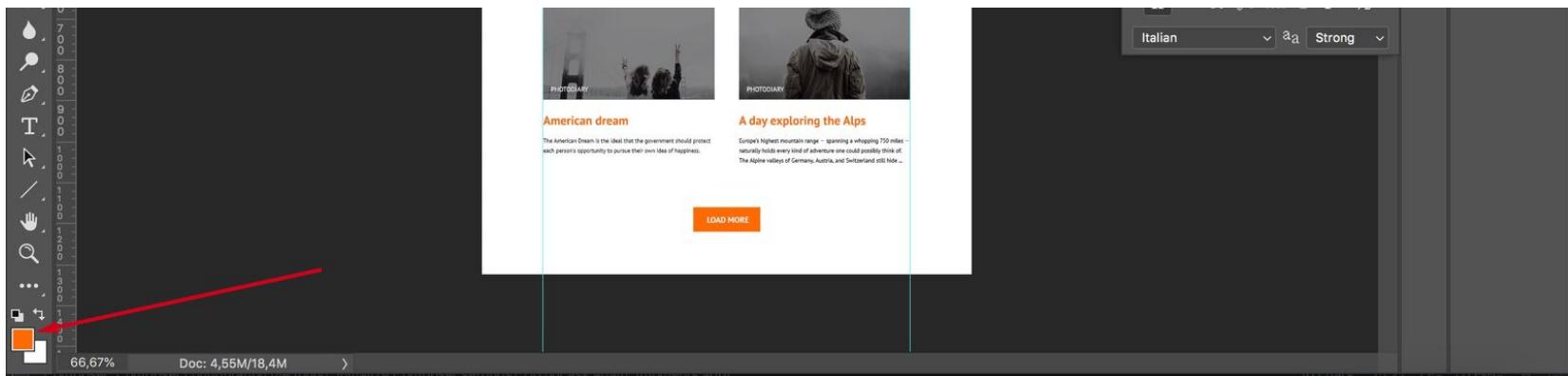
# EYEDROPPER TOOL

Инструмент **Eyedropper tool (Пипетка)** находится слева в панели инструментов:



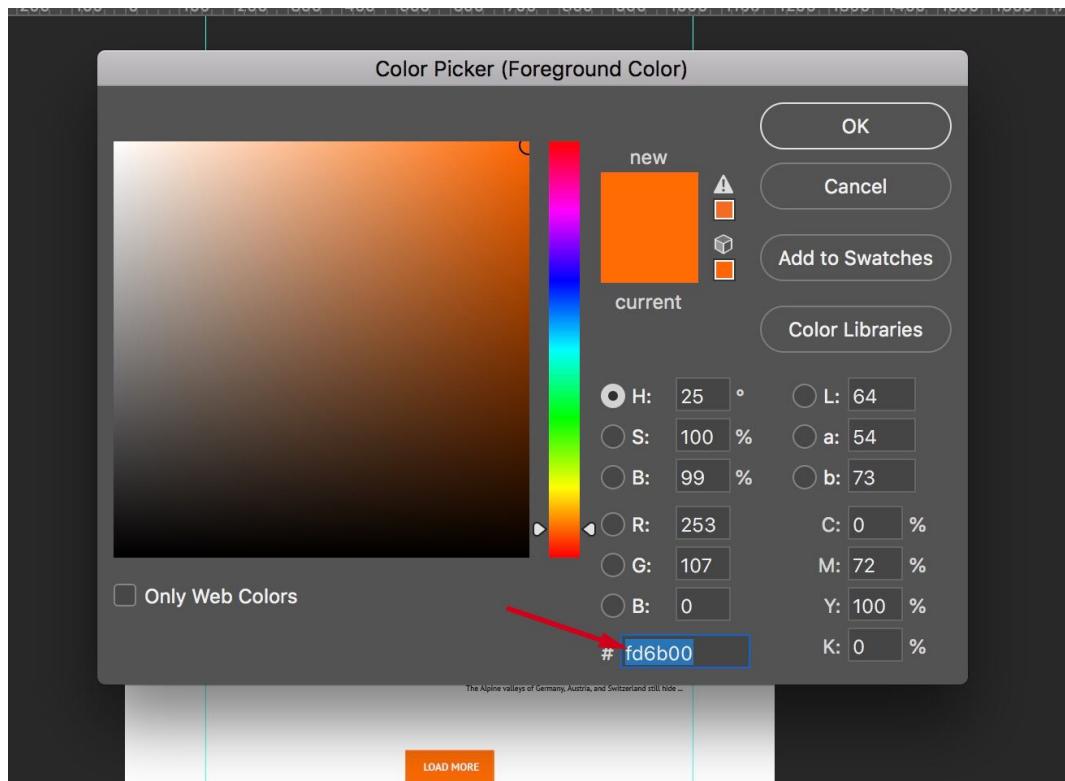
# EYEDROPPER TOOL

Если выбрать этот инструмент, курсор превращается в пипетку и по клику левой кнопкой мыши на любую точку макета слева снизу в панели инструментов появится квадрат с выбранным цветом:



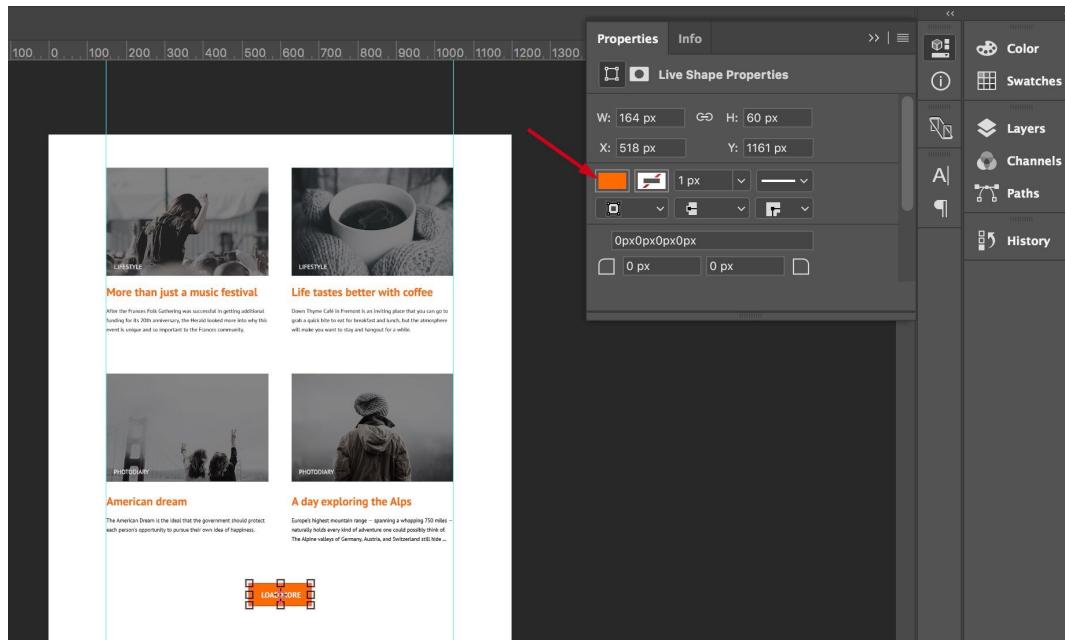
# EYEDROPPER TOOL

По клику на этот квадрат открывается окно, в котором можно увидеть цвет этой точки. Если кликнуть на кнопку «Load more», мы увидим следующее:



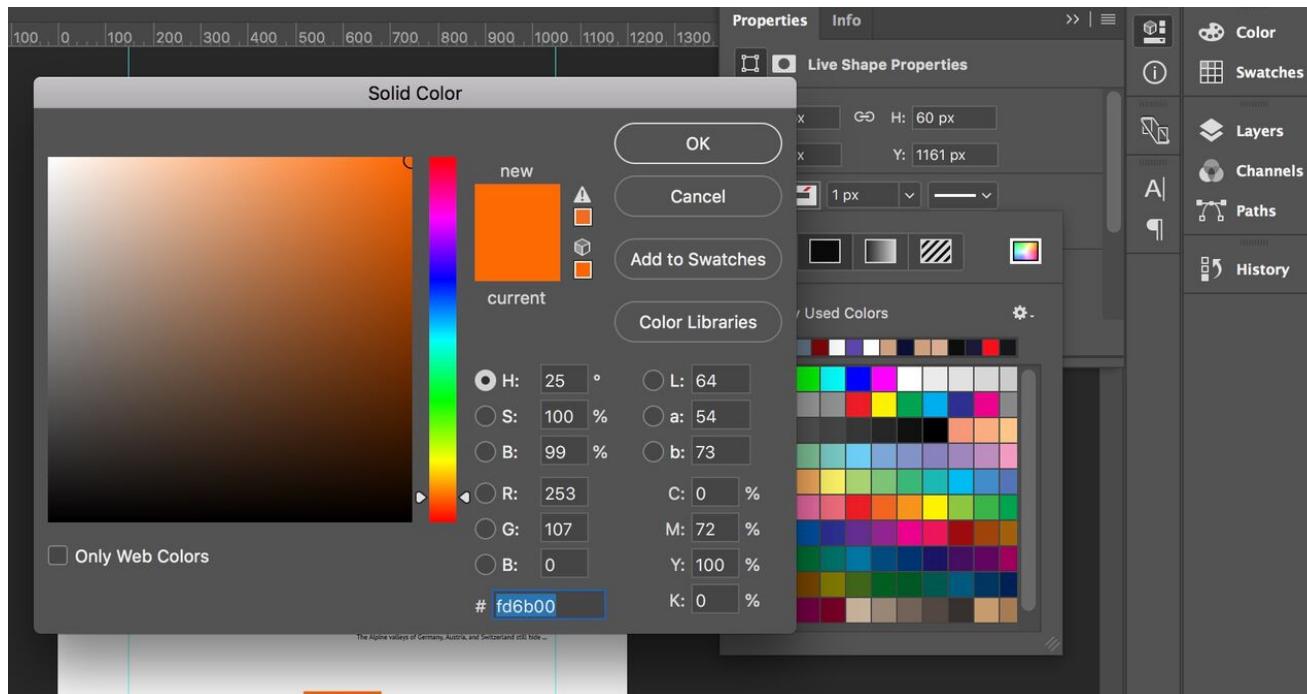
# INFO

Если мы выберем слой фона кнопки, то в панели **Info (Информация)** на вкладке **Properties (Свойства)** появится цвет слоя:



# INFO

Если кликнуть на окошко цвета, и в раскрывшемся окне кликнуть на кнопку **Color Picker (выбор цвета)**, то откроется такое же окно, как при клике инструментом **Eyedropper tool (Пипетка)**:



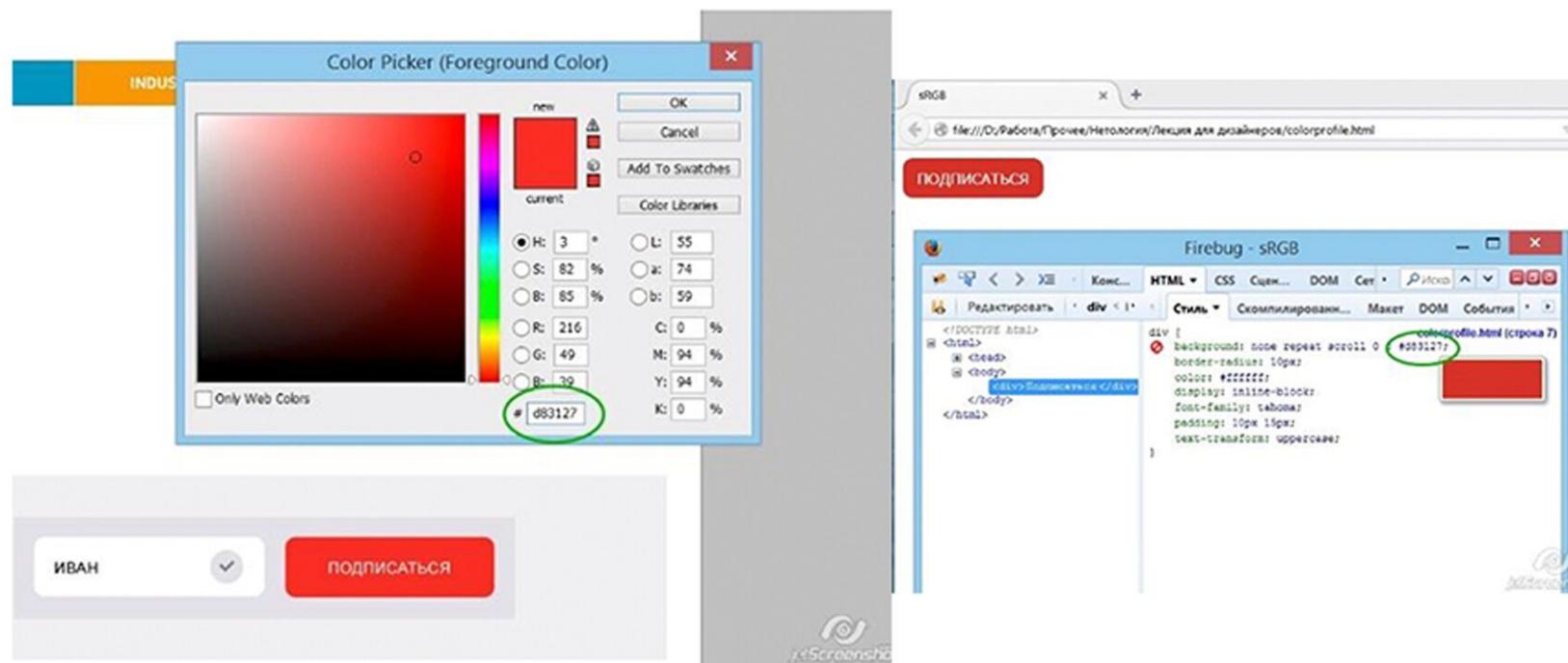
# ЗАПИШЕМ ЦВЕТ В ФАЙЛ

```
1 | .load-more-posts{  
2 |   background-color: #fd6b00;  
3 | }
```

# ПРОВЕРКА ЦВЕТОВОГО ПРОФИЛЯ МАКЕТА

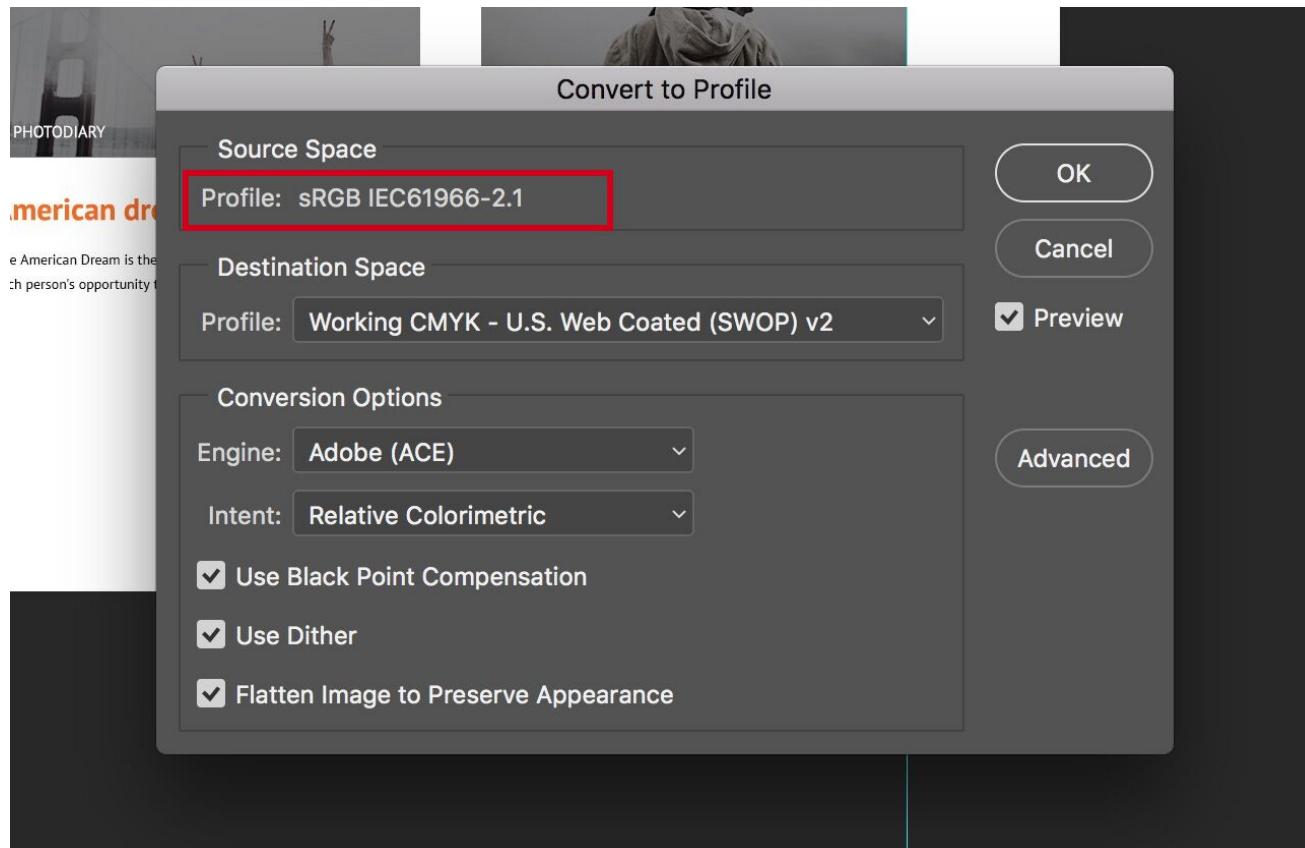
Adobe Photoshop позволяет создавать макеты в двух разных цветовых профилях: **sRGB (safe rgb - «безопасный» rgb)** и **AdobeRGB**.

Цветовой профиль AdobeRGB сильно отличается от цветов, которые отображает браузер.



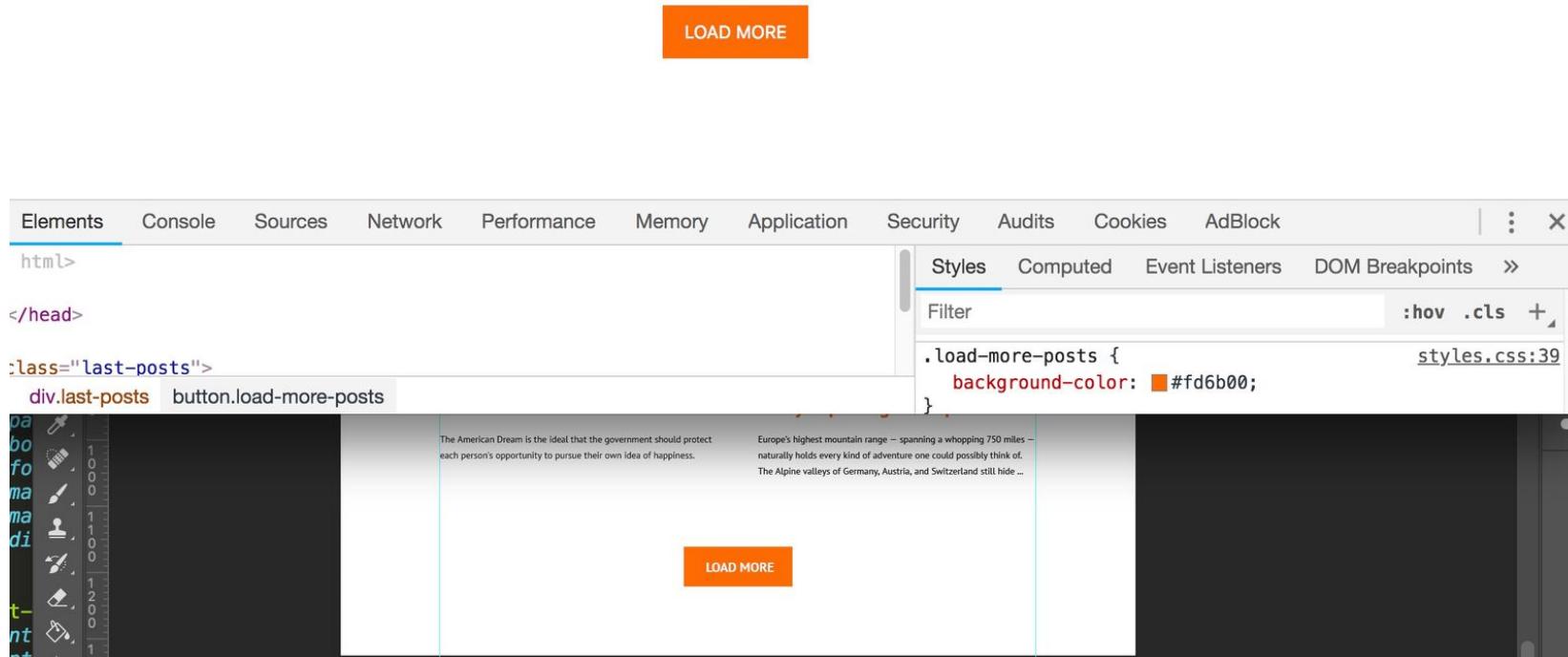
# ПРОВЕРЯЕМ ЦВЕТОВОЙ ПРОФИЛЬ

Выбрать пункт меню **Edit (Правка) - Convert to profile (Конвертация профиля)**. Откроется окно, в котором можно увидеть текущий цветовой профиль:



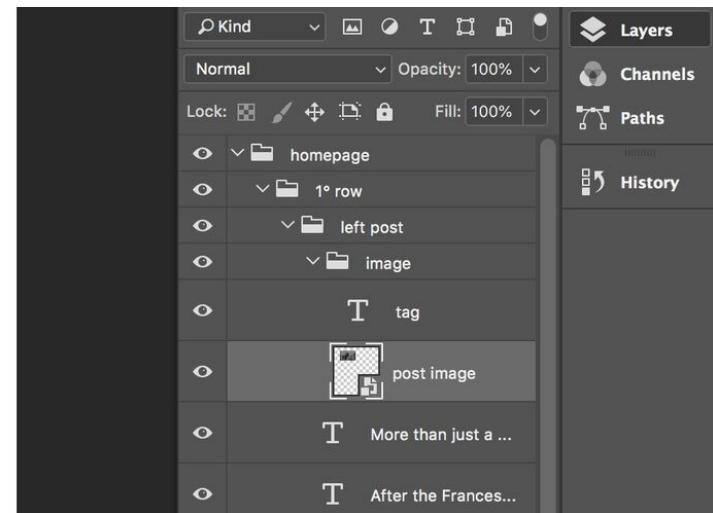
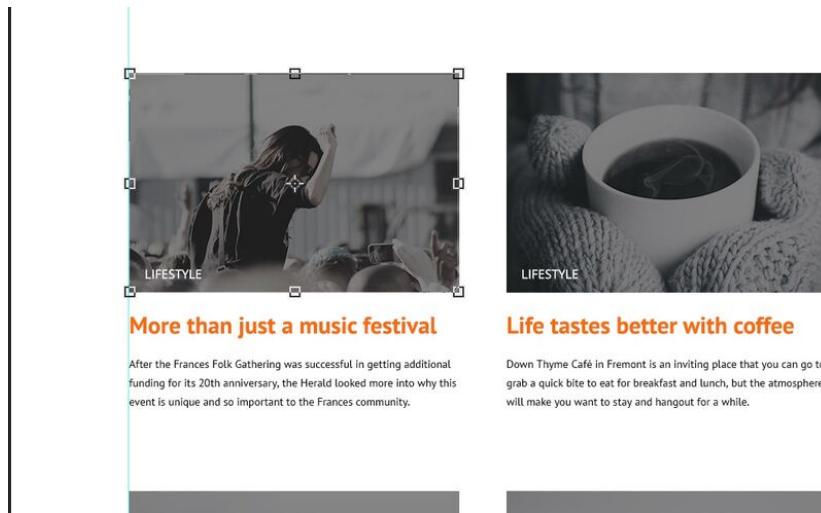
# ПРОВЕРЯЕМ ЦВЕТОВОЙ ПРОФИЛЬ

В нашем макете установлен цветовой профиль sRGB, поэтому мы смело можем использовать все цвета, и они будут похожи на макет:



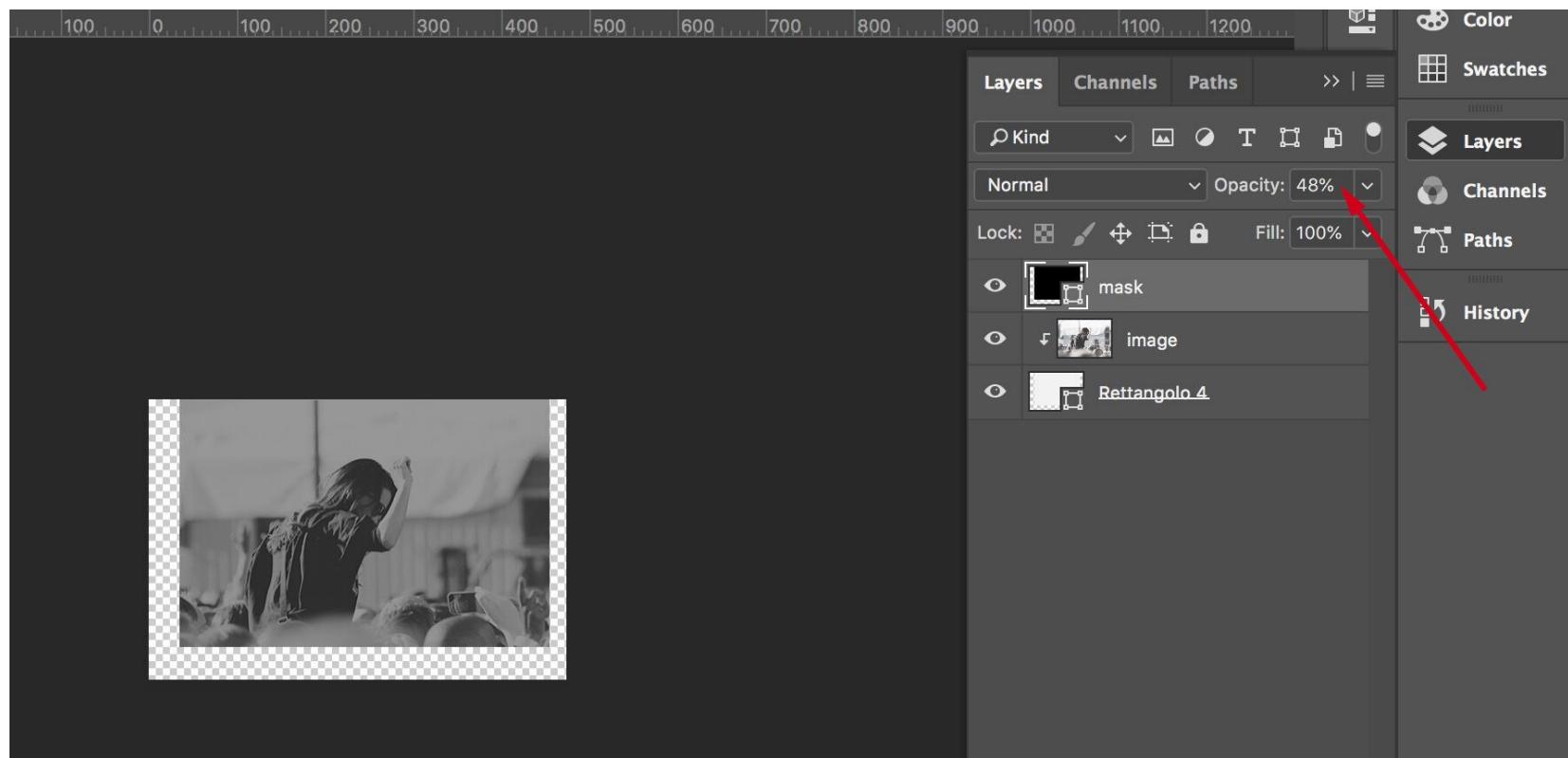
# ПРОЗРАЧНОСТЬ

Поверх каждого изображения в макете находится цветовая маска. Маска представляет собой цветовой слой с прозрачностью, и нужна для того, чтобы светлый текст не “сливался” со светлыми участками фото.



# ПРОЗРАЧНОСТЬ

Т.к. этот слой является смарт-объектом (smart object), чтобы посмотреть его составные части, нам нужно дважды кликнуть на иконку слоя и выделим слой маски.



# ЗАПИШЕМ КОД ДЛЯ МАСКИ

```
1 .last-post-image-wrapper:before{  
2     content: "";  
3     position: absolute;  
4     width: 100%;  
5     height: 100%;  
6     top: 0;  
7     left: 0;  
8     background-color: #000000;  
9     opacity: 0.48;  
10    }
```

---

# ИЗОБРАЖЕНИЯ

# ФОРМАТЫ ИЗОБРАЖЕНИЙ

Мы уже знаем, что вставить изображение в верстку мы можем с помощью тега `img`. Для этого нужно записать код:

```

```

А если нам нужно задать изображение для фона, то мы должны записать такой код:

```
background-image: url("../images/gallery/cat.jpg");
```

## JPG/JPEG

Самый популярный формат файла для верстки. Лучше всего **jpg** подходит для хранения изображений и фонов, для которых *не нужна прозрачность*.

jpg и jpeg – это имена одного и того же формата, их можно заменять один на другой, при этом файл поврежден не будет. jpg – более старое расширение, которое использовалось на первых Windows.

# PNG

Используется когда изображение должно выводиться поверх какого-то фона:

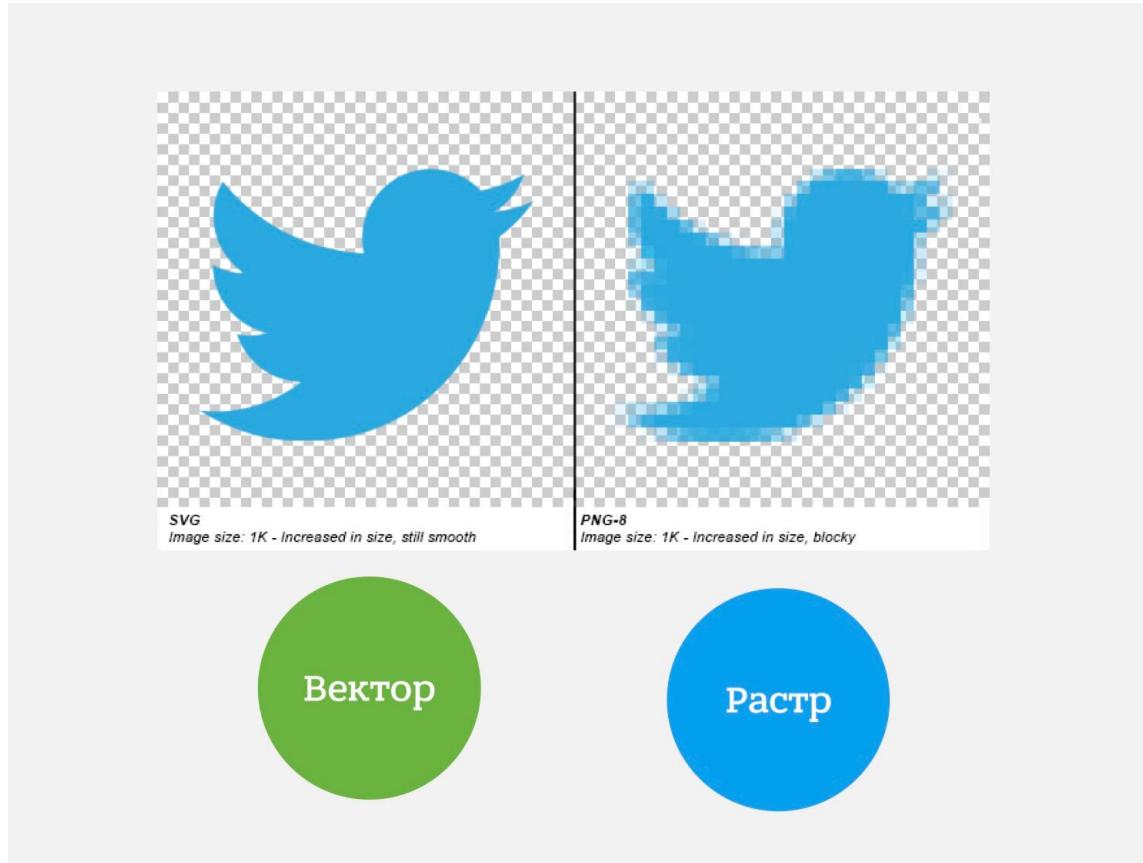


Даже если цвет фона страницы изменится, изображение будет выглядеть хорошо:



# SVG

Формат **svg** был создан для хранения векторной графики:



# СОХРАНЕНИЕ ИЗОБРАЖЕНИЯ

Посмотрим на наш макет:



LIFESTYLE

**More than just a music festival**

After the Frances Folk Gathering was successful in getting additional funding for its 20th anniversary, the Herald looked more into why this event is unique and so important to the Frances community.



LIFESTYLE

**Life tastes better with coffee**

Down Thyme Café in Fremont is an inviting place that you can go to grab a quick bite to eat for breakfast and lunch, but the atmosphere will make you want to stay and hangout for a while.



PHOTODIARY

**American dream**

The American Dream is the ideal that the government should protect each person's opportunity to pursue their own idea of happiness.



PHOTODIARY

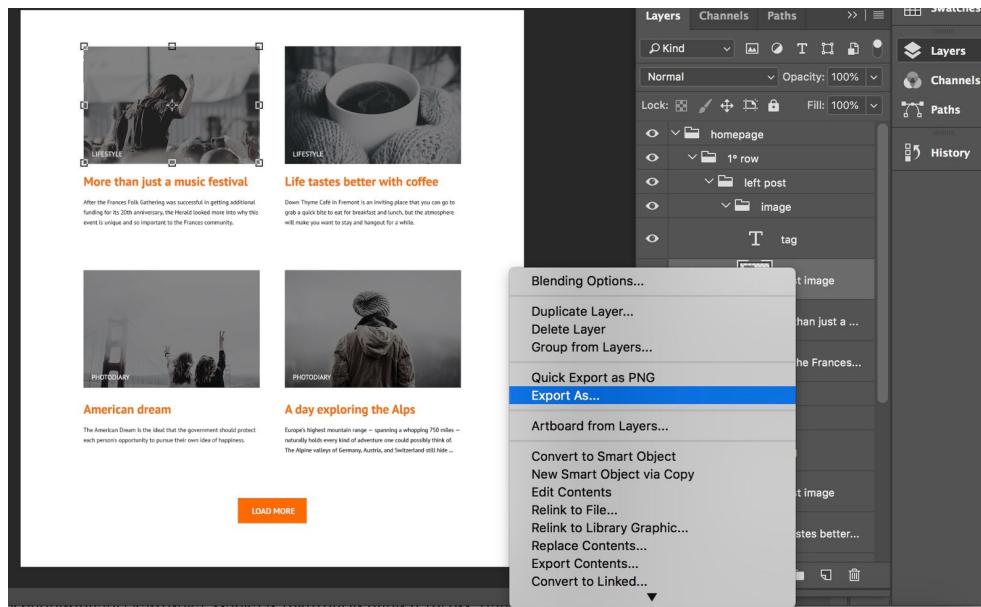
**A day exploring the Alps**

Europe's highest mountain range – spanning a whopping 750 miles – naturally holds every kind of adventure one could possibly think of. The Alpine valleys of Germany, Austria, and Switzerland still hide ...

[LOAD MORE](#)

# СОХРАНЯЕМ ИЗОБРАЖЕНИЕ

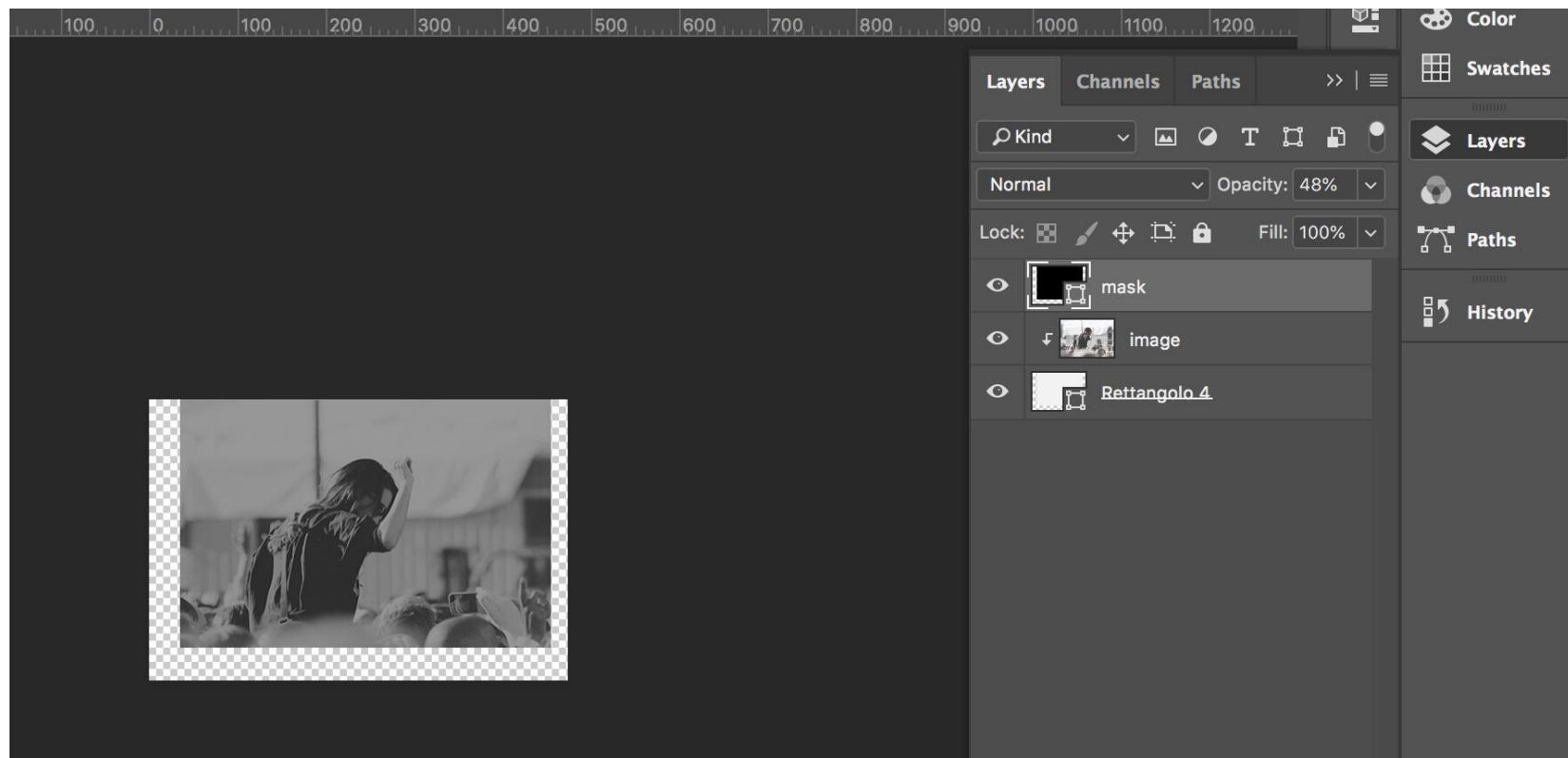
Чтобы сохранить изображение, воспользуемся пунктом **Export as** из меню слоя:



Но если мы откроем сохраненное изображение, то увидим, что это не то, что мы хотели.

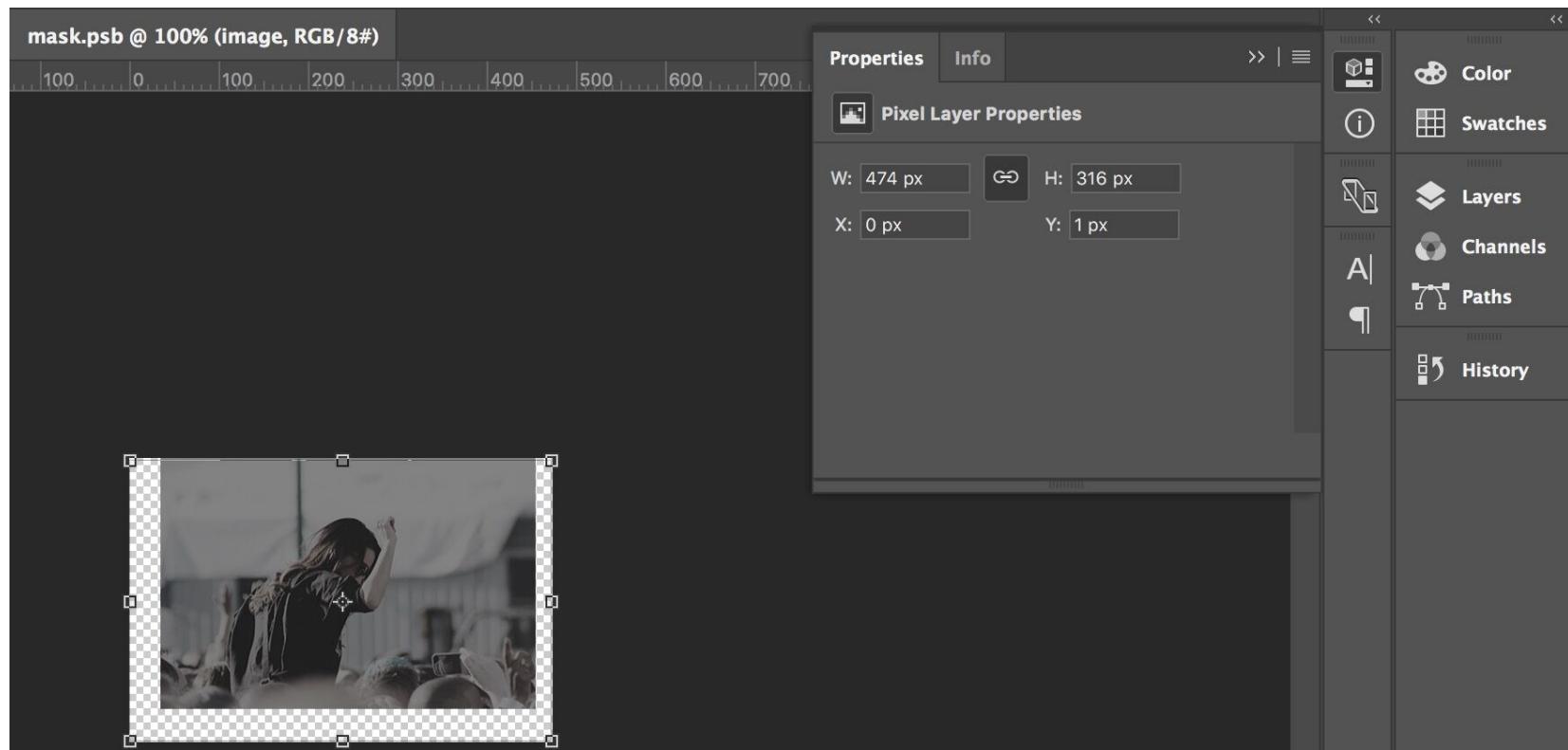
# СОХРАНЯЕМ ТОЛЬКО КАРТИНКУ

Для этого снова откроем смарт-объект в новом окне:



# СОХРАНЯЕМ ТОЛЬКО КАРТИНКУ

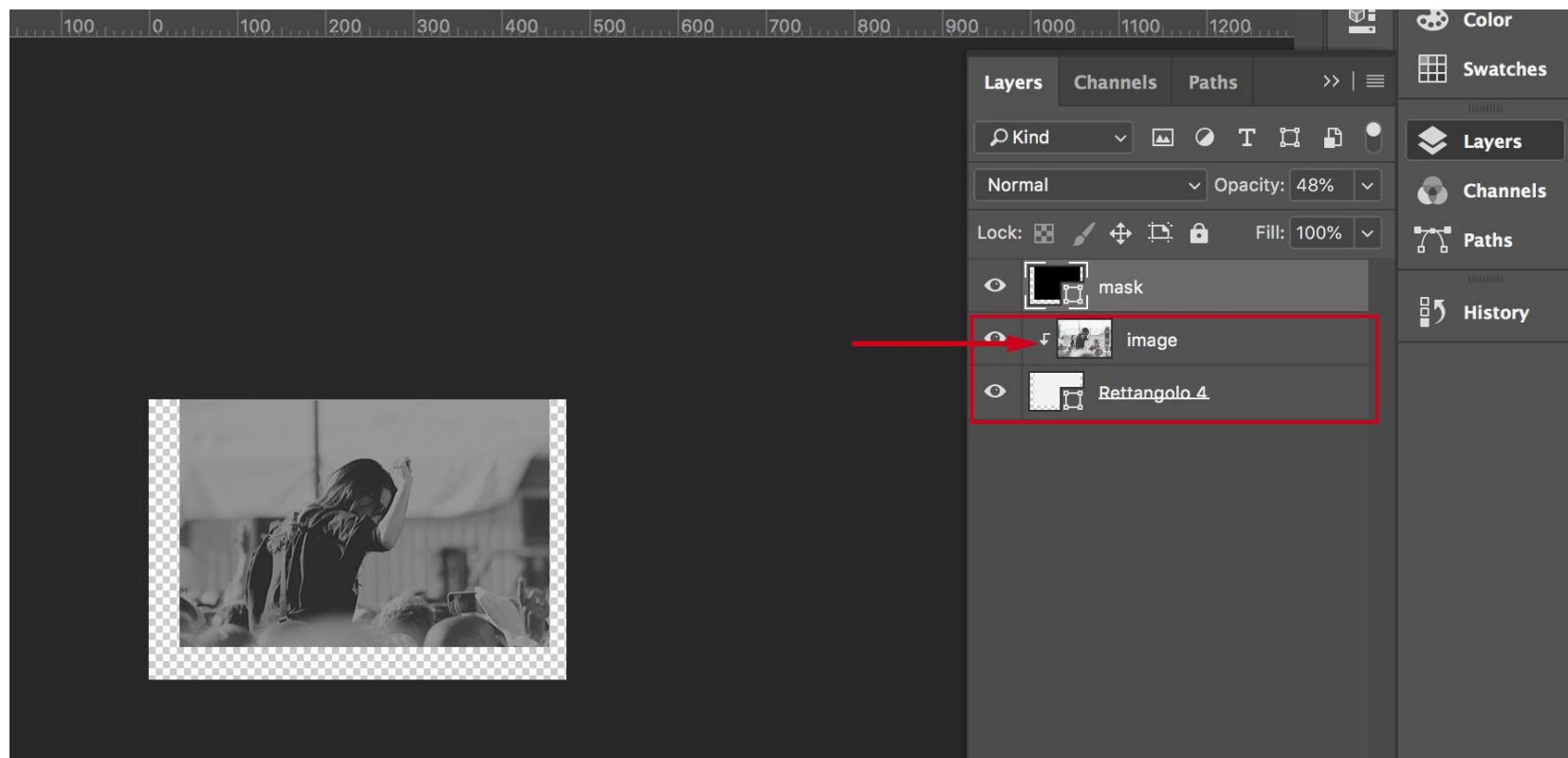
Наш блок с картинкой имеет размер 420\*280, но если мы посмотрим внимательнее, то увидим, что исходное изображение имеет больший размер:



# СОХРАНЯЕМ ТОЛЬКО КАРТИНКУ

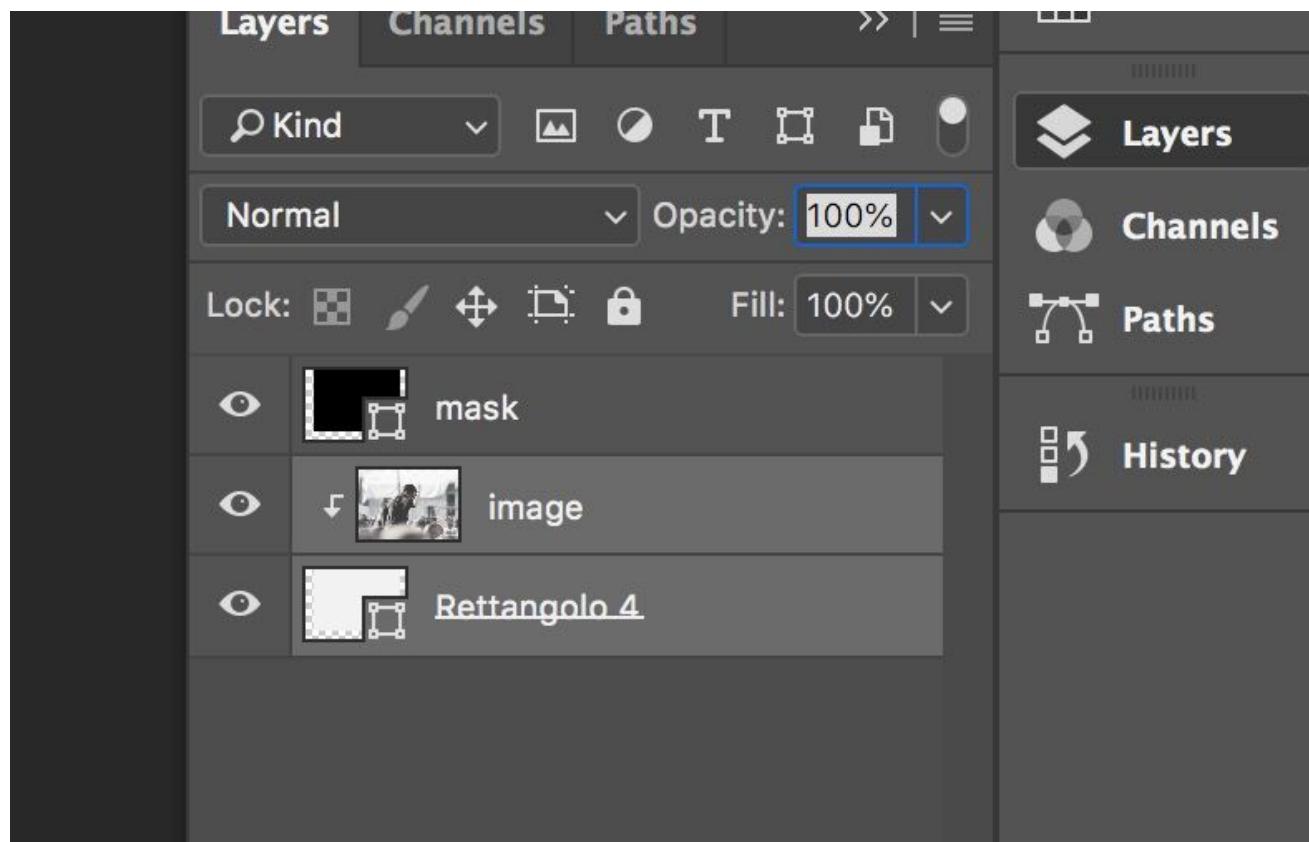
А как же получилось так, что картинка в макете имеет размер 420\*280?

Все дело в ограничивающей размер маске, в панели слоев она называется **rectangle** и к ней от изображения показана стрелка:



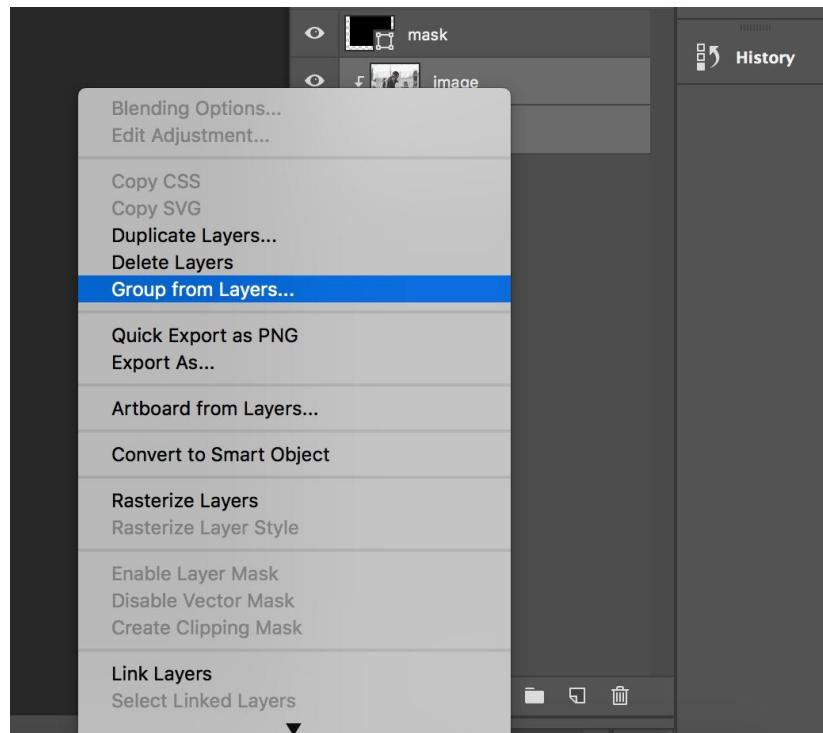
# ОБЪЕДИНИМ МАСКУ И КАРТИНКУ

ЭТО МОЖНО СДЕЛАТЬ, ВЫДЕЛИВ ОБА СЛОЯ



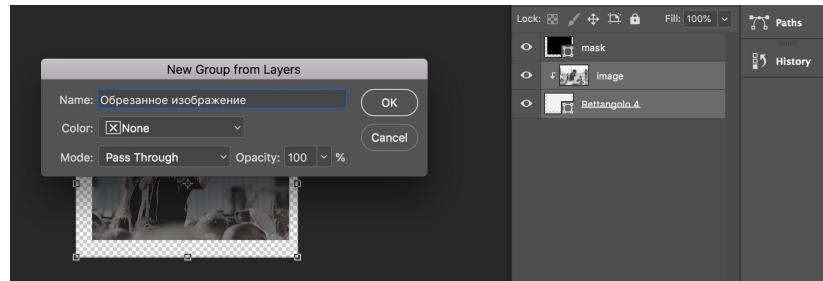
# ОБЪЕДИНИМ МАСКУ И КАРТИНКУ

Теперь кликнем по слоям правой кнопкой мыши, и в появившимся меню выберем **Group from layers** (**Сгруппировать слои**):

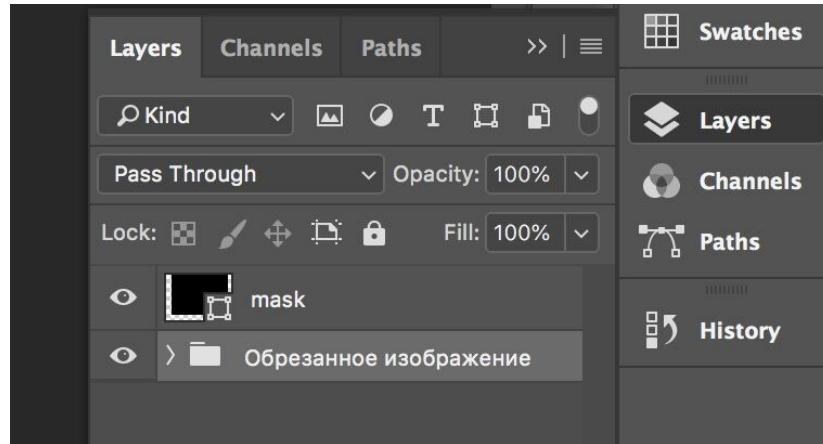


# ОБЪЕДИНИМ МАСКУ И КАРТИНКУ

Назовем группу «обрезанное изображение»:

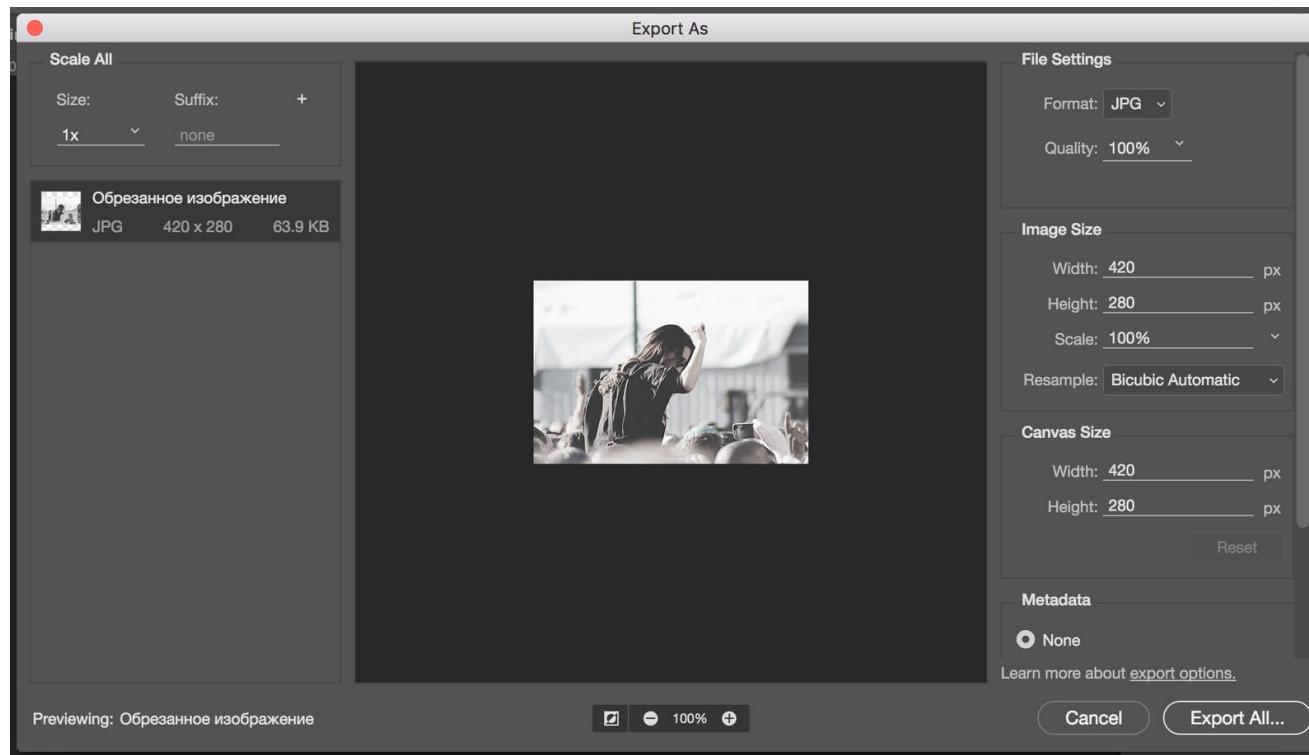


В панели слоев появилась группа:



# ЭКСПОРТ

Кликнем правой кнопкой мыши на группу, и выберем **Export as**, а в появившемся окне укажем формат **jpg** , нажмем **Export All** (экспортировать) и выберем, куда сохранять изображение:



---

# ГОТОВО

Нужный нам файл в формате **jpg**, правильного размера и без лишних слоев.



---

# ИТОГИ

- Свойства `justify-content` и `align-items` отвечают за позиционирование элемента по основной и дополнительной осям;
- Элемент с `position: fixed` фиксируется при прокрутке страницы, его ширина и высота рассчитываются по содержимому, однако ее можно задать, не видим для родителя и соседних элементов;
- Свойство `display: none` полностью скрывает элемент (ширина и высота элемента принимают значение `0`, дочерние интерактивные элементы не будут доступны с клавиатуры);

---

# ИТОГИ

- Дочерним селектором называют группу из двух элементов, когда один из них является родителем, а другой ребенком.
- Вложенным селектором называют группу из нескольких элементов, когда один из них является родителем, а другие потомками.
- Используя ключевое слово `!important`, мы сообщим браузеру, какое значение ему стоит принять без учета специфичности.
- Конструкция `@font-face` служит для подключения шрифтов.



**Ваши вопросы?**

**СЕМЕН БОЙКО**



[simonderus@gmail.com](mailto:simonderus@gmail.com)



[fb.me/simonboycko](https://fb.me/simonboycko)