

# Оптимизация загрузки картинок Обзор подходов



Подлевских Милена Сергеевна  
Инженер-разработчик клиентских приложений

# Зачем оптимизировать загрузку изображений?

- Согласно HTTP Archive, 60% объёма веб-страниц — это изображения.
- Одна из главных причин долгой загрузки — «тяжелые» изображения.
- Скорость загрузки страниц — это один из факторов в алгоритме ранжирования мобильного поиска Google.
- Изображения на странице повышают коэффициент конверсии, поэтому просто убрать их нельзя.

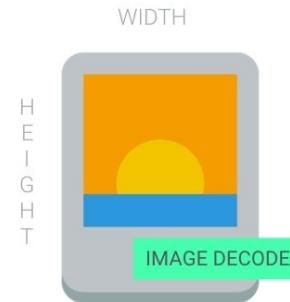
# Что даст оптимизация загрузки изображений?

- Ускоренную загрузку страниц сайта.
- Более высокие позиции в поисковой выдаче.
- Увеличение конверсии из-за меньшего количества закрытий сайта с мыслью «Аaaaa, как долго загружается».
- Снижение нагрузки на хостинг, что также увеличивает скорость работы ресурса.
- Экономию места на диске, что позволит вам сэкономить копеечку на оплате хостинга.

## Image Optimisation



Choose the right format



Size appropriately



Adapt intelligently



Compress carefully



Prioritize critical images



Lazy-load the rest



Take care with tools

# Как узнать, нужно ли оптимизировать изображения?

## Website Speed Test

### Lighthouse в Chrome DevTools

**Lighthouse Opportunities:**

- Remove unused JavaScript (Estimated Savings: 1.77 s)
- Optimize images (Estimated Savings: 0.99 s)

Resource	Current Size	Potential Savings
JPEG	78.2 KB	77 KB
JPEG	62.8 KB	61.8 KB
JPEG	58.8 KB	57.2 KB
JPEG	56.2 KB	55.3 KB
JPEG	55 KB	54.1 KB
JPEG	50.4 KB	49.6 KB
JPEG	44.2 KB	43.6 KB
JPEG	41.3 KB	40.6 KB
JPEG	36.9 KB	36.3 KB

**Page Image Score:** B (Good)

**Total Images Analyzed:** 10

**Total Image Weight:** 179KB

**Potential Compressed Weight:** 138.9 KB (24.1%)

**Opportunities:**

- downsize\_200k\_v1.jpg (JPEG from img.championat.com) - Potential Smart Compression: 87.2 KB (0.8%)
- downsize\_200k\_v1.jpg (JPEG from img.championat.com) - Potential Smart Compression: 38.4 KB (50.2%)
- downsize\_200k\_v1.jpg (JPEG from img.championat.com) - Potential Smart Compression: 36.2 KB (55.7%)

# Какой формат изображения лучше выбрать?

GIF



PNG



JPG



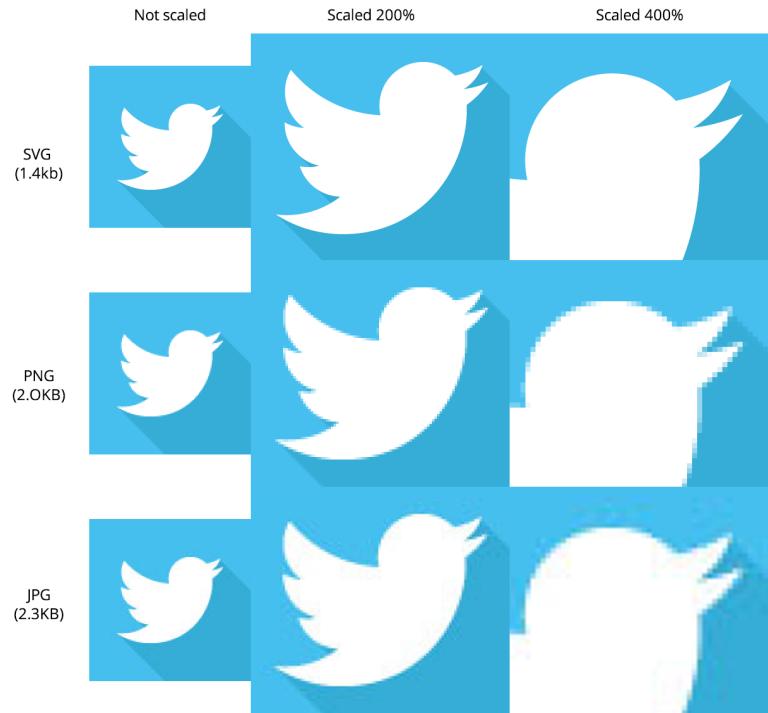
# Форматы изображений

	PNG	JPG	GIF	WebP
Сжатие с потерями	●	●	●	●
Сжатие без потерь	●	●	●	●
Прозрачность	●	●	●	●
Анимация	●	●	●	●

● No      ● Yes

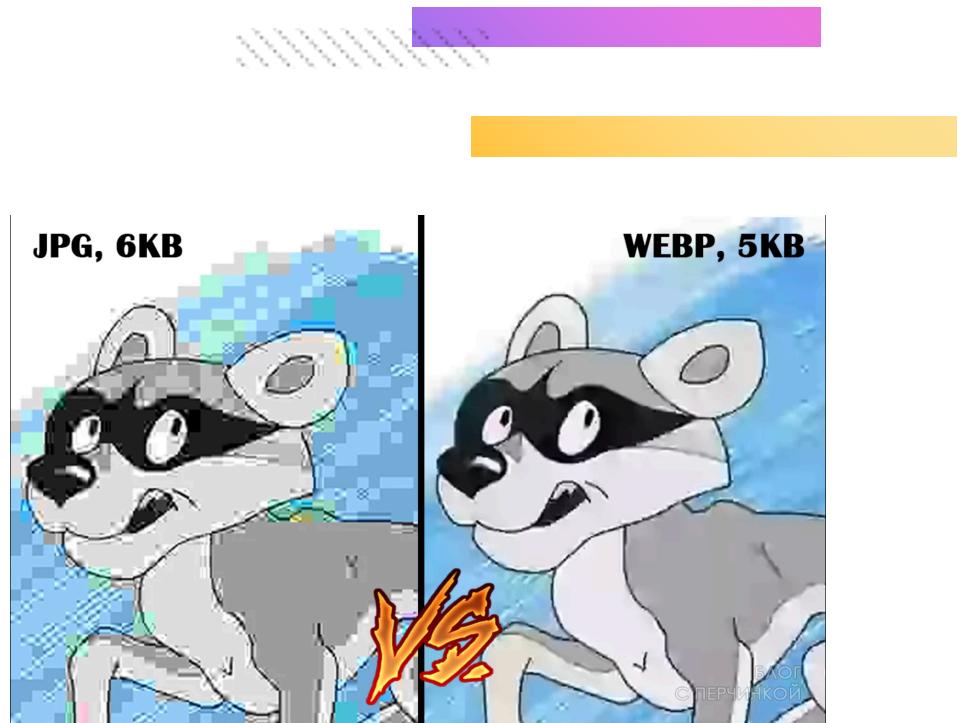
# Форматы изображений

- Для логотипов, иллюстраций, диаграмм или значков и иконок лучше использовать svg - для векторной и png - для растровой графики.
- Для фотографических мотивов, не обладающих прозрачностью можно выбрать хорошо сжатый прогрессивный JPEG.

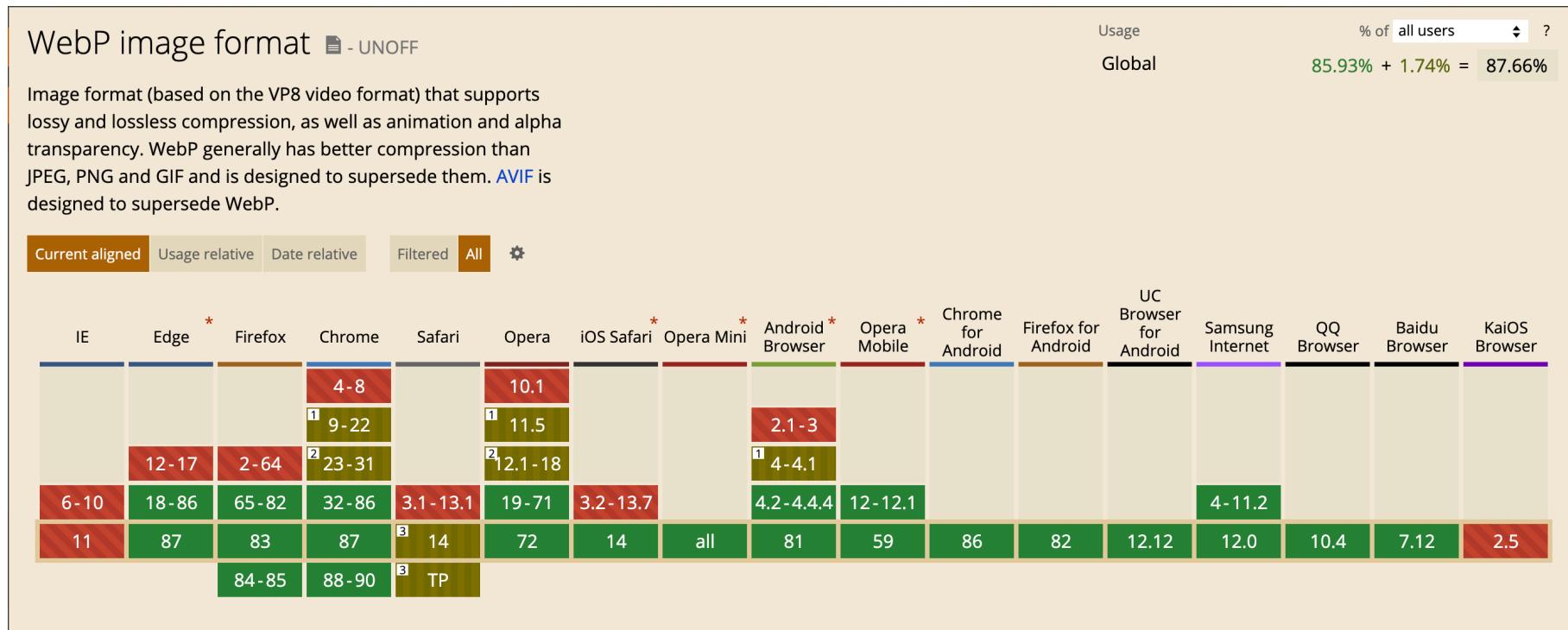


# Формат WebP

- Позволяет сжать картинку без потерь до 25% сильнее, чем тот же JPEG.
- Google сообщил об экономии 30–35% на WebP по сравнению с другими схемами сжатия с потерями
- Не поддерживается всеми браузерами.



# Поддержка браузерами WebP



# Сжатие GIF-анимаций и почему <video> лучше



MP4  
401 KB



GIF  
3.1 MB

## Сжатие и потеря качества

- Сжимать изображение всегда рекомендуется из исходника.
- Каждый дополнительный раунд сжатия приведёт к дополнительной потере качества.
- Исходные файлы храните их в формате без потерь.



степень сжатия по шкале # '%\$@&!~ шакалов

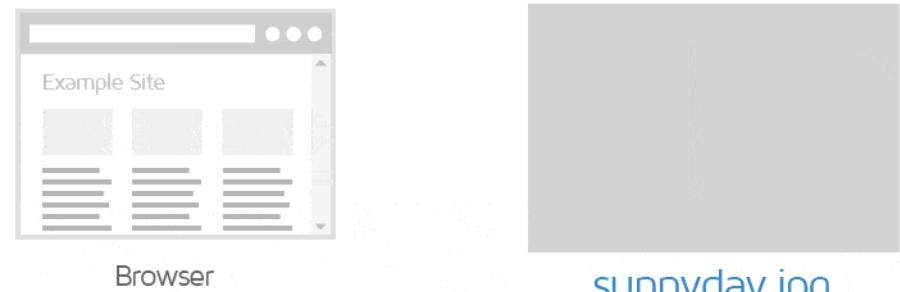


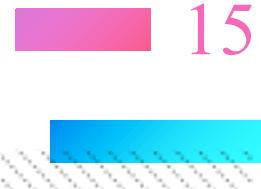
# Использование оптимального размера

# Размер изображений

- Не стоит отдавать слишком большие изображения.
- Пропуск атрибутов `width` или `height` тоже может негативно повлиять на производительность

## 1. Browser requests and downloads image





# Соотношение пикселей устройств

**Device Pixel Ratio**

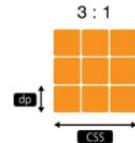
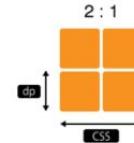
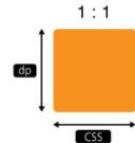
dpr=1.0



dpr=2.0 (e.g Moto G)

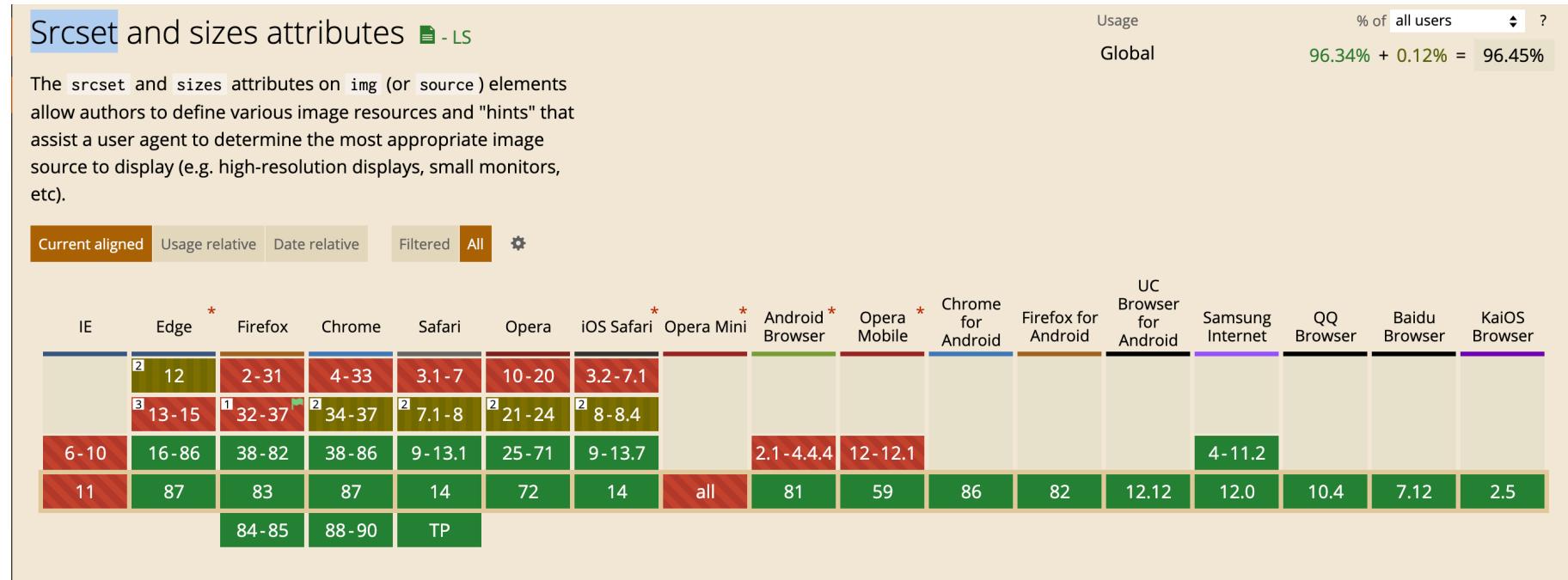


dpr=3.0 (e.g iPhone 6 Plus)



```
1   
```

# Поддержка браузерами атрибута Srcset

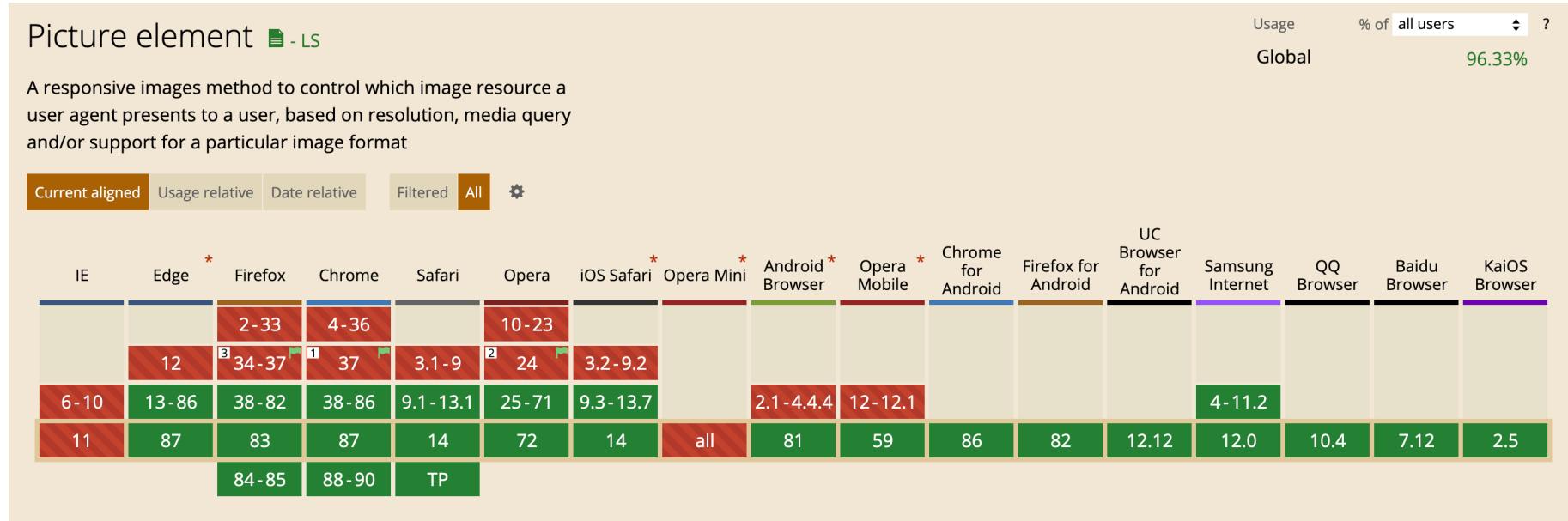


# Художественное преобразование

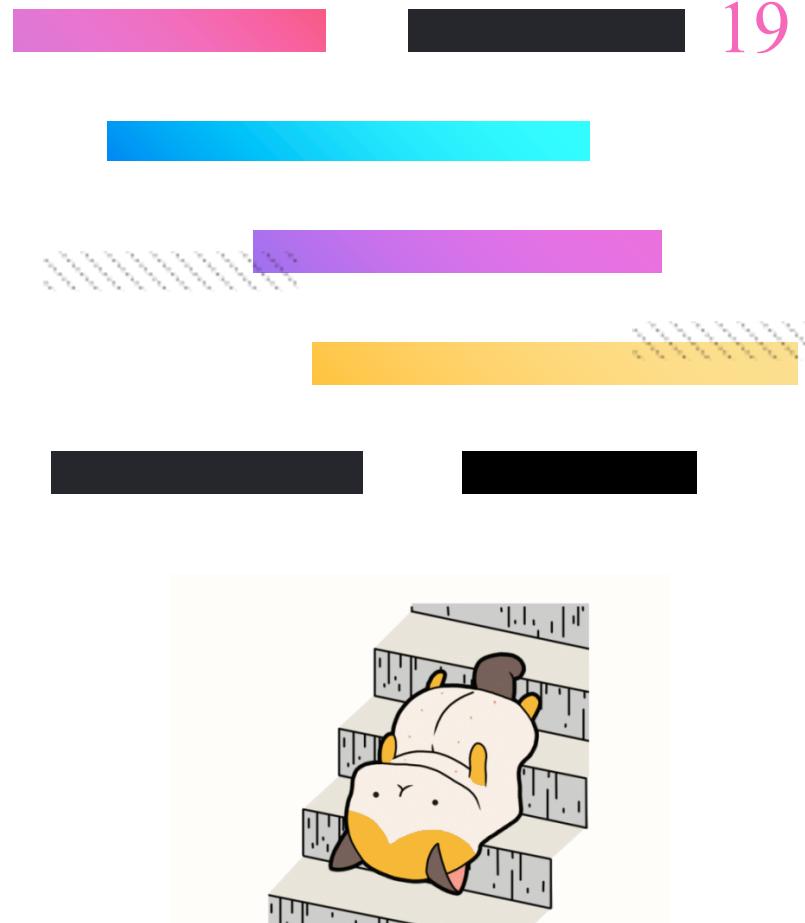
```
9  <picture>
10 <source media="(max-width: 799px)" srcset="dog-image-480w.jpg">
11 <source media="(min-width: 800px)" srcset="dog-image-800w.jpg">
12 
13 </picture>
14
```



# Поддержка браузерами элемента Picture



# Ленивая загрузка изображений



# Преимущества отложенной загрузки



- Экономия трафика.
- Экономия заряда батареи.
- Увеличение скорости загрузки.

# Scrolling

Complete Oia Santorini Travel Guide: Everything You need to...  
3,6 тыс.

10 Very Best Things To Do In Bath, England

Cappadocia, Turkey: Travel & Photography Guide | Through...

Vacation Travel Guide: The Best Islands In The Bahamas

Best Travel Blogs: The Top 30 List

15 jaw-droppingly beautiful waterfalls in Iceland

https://www.pinterest.ru/pin/310346442044091825/



# Click

City quiz

Start quiz





# Нативная отложенная загрузка в браузере

- "eager"- получить изображение прямо при загрузке страницы
- "lazy"- получить изображение, когда оно находится в области просмотра или рядом с ним
- "auto"- позволить браузеру решать, как загружать (используется, если атрибут не указан)

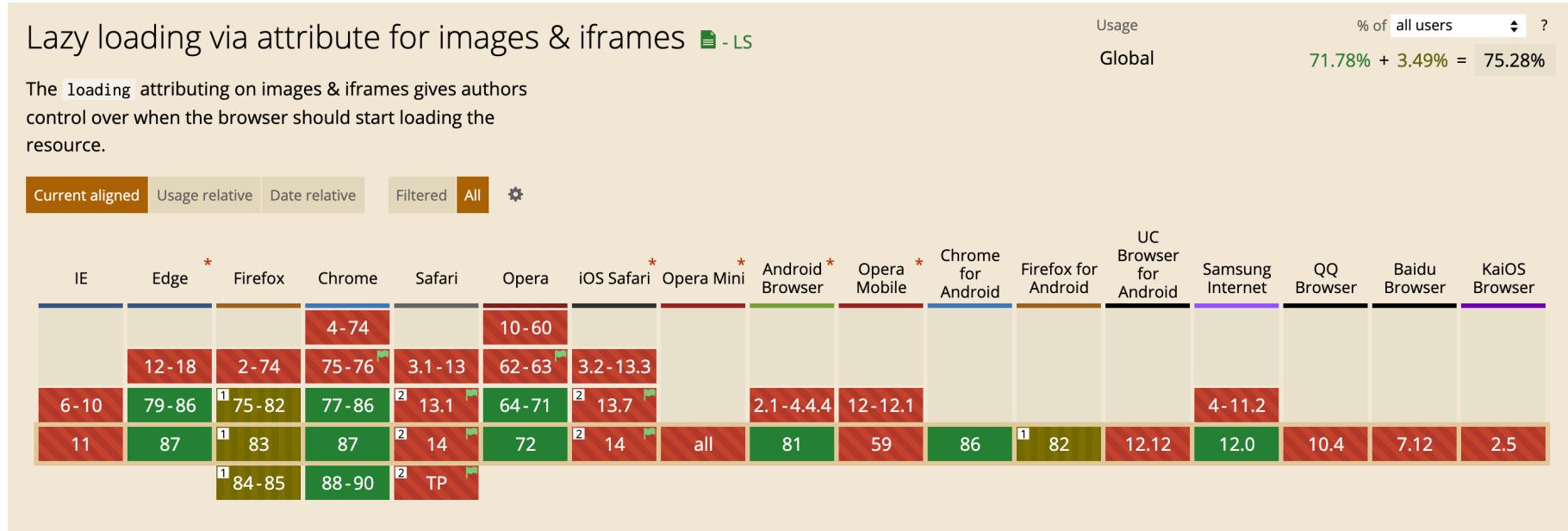
```
20
```

```
21     
```

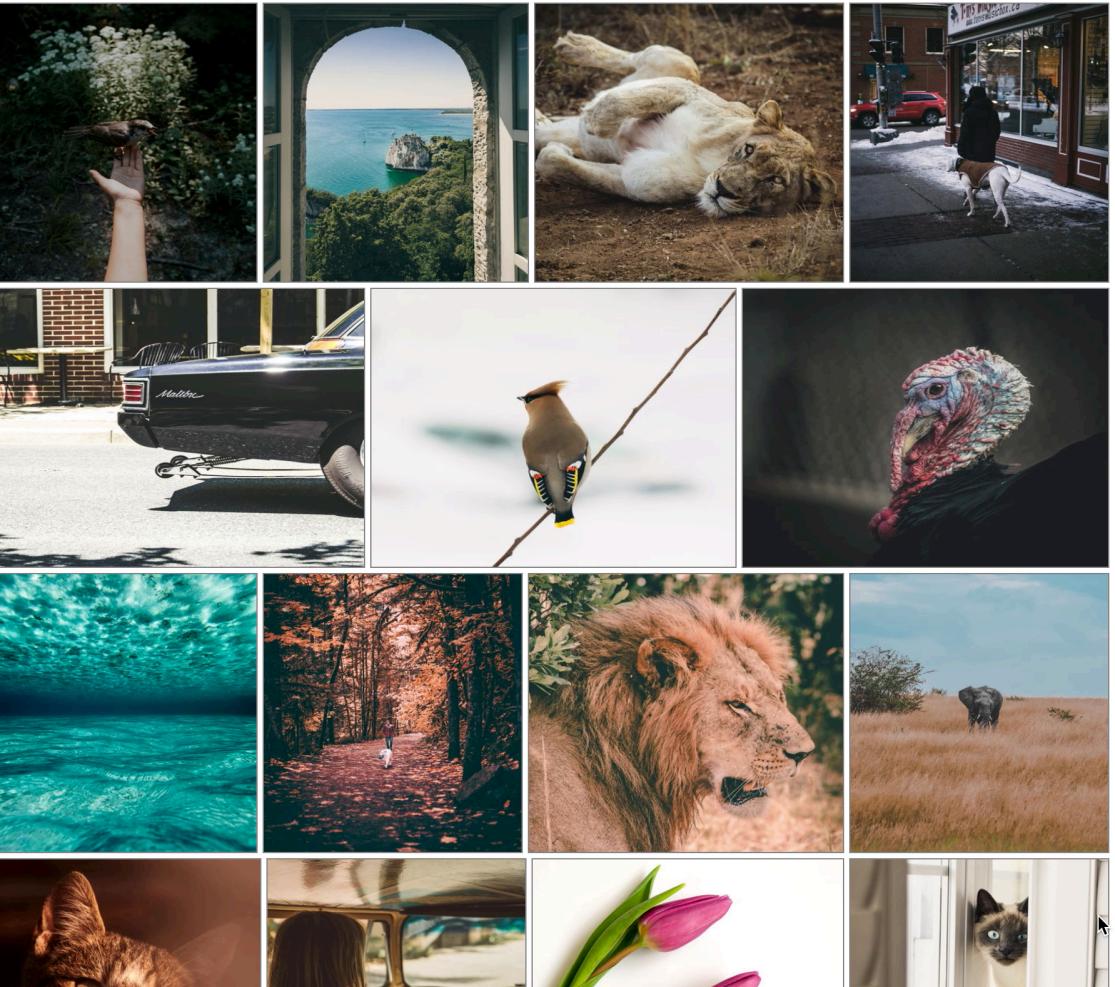
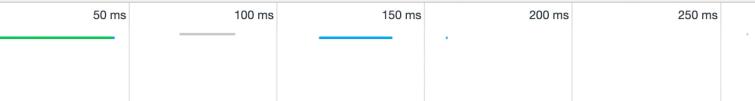
```
22
```

```
23
```

# Поддержка браузерами атрибута loading



## Photos

Filter  Hide data URLsAll | XHR JS CSS **Img** Media Font Doc WS Manifest Other  Has blocked cookie: Blocked Requests

Name	Status	Type	Initiator	Size	T..	Waterfall
Image86.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image87.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image88.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image89.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image90.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image91.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image92.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image93.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image94.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image95.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image96.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image97.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image98.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image99.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image100.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image101.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image102.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image103.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image104.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image105.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image106.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image107.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image108.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image109.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image110.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image111.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
Image112.jpg	200	jpeg	<a href="#">lazy_loading....</a> (mem...	0...		<span style="color: #00f;"> </span>
ic_menu_24px.svg	200	svg+xml	main.css (mem...	0...		<span style="color: #00f;"> </span>

113 / 117 requests | 0 B / 66 B transferred

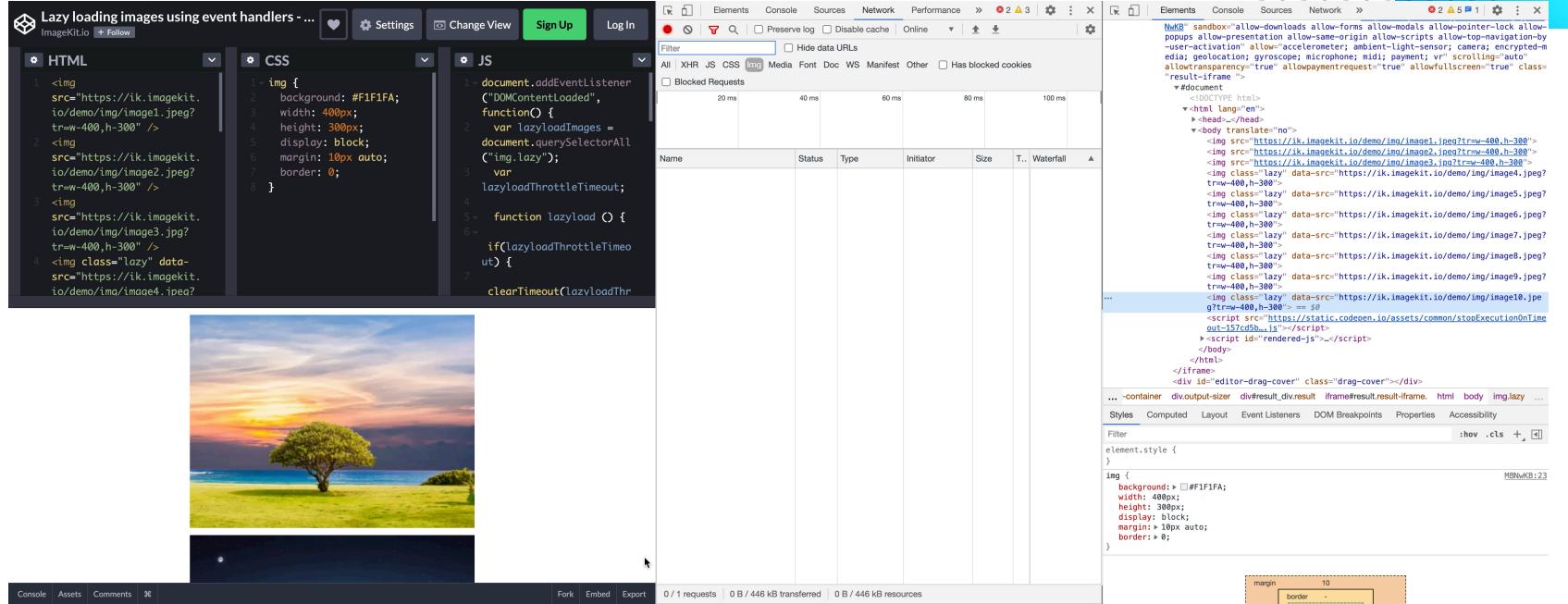
7.5 MB / 7.6 MB resources | Finish: 256 ms | DOMContentLoaded

# Загрузка изображений с помощью событий JavaScript

```
22
23     
24
```

```
25     [].forEach.call(document.querySelectorAll('img[data-src]'),
26     function(img) {
27         img.setAttribute('src', img.getAttribute('data-src'));
28         img.onload = function() {
29             img.removeAttribute('data-src');
30         };
31     }
32 );
```

# Загрузка изображений с помощью событий JavaScript

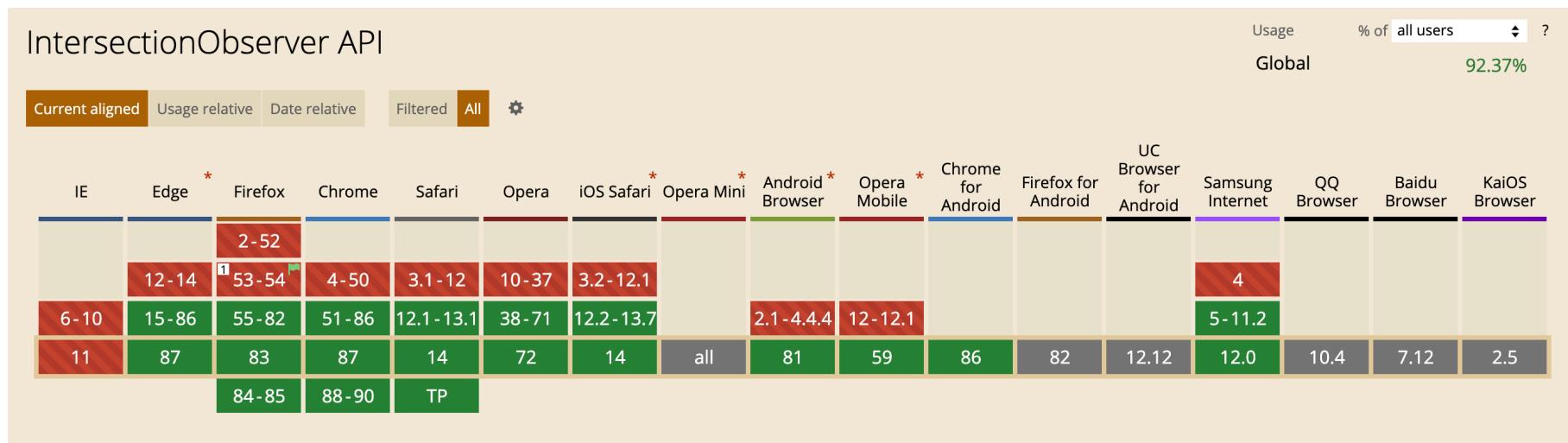


# Загрузка изображений с помощью Intersection Observer API

```
33
34     
35
```

```
35
36     lazyloadImages = document.querySelectorAll(".lazy");
37     var imageObserver = new IntersectionObserver(function(entries, observer) {
38         entries.forEach(function(entry) {
39             if (entry.isIntersecting) {
40                 var image = entry.target;
41                 image.src = image.dataset.src;
42                 image.classList.remove("lazy");
43                 imageObserver.unobserve(image);
44             }
45         });
46     });
47
48     lazyloadImages.forEach(function(image) {
49         imageObserver.observe(image);
50     });
51
```

# Поддержка браузерами Intersection Observer API



# Загрузка изображений с помощью Intersection Observer API

The screenshot illustrates the process of lazy loading images using IntersectionObserver. The browser's developer tools are open, showing the following tabs:

- Elements**: Shows the DOM structure.
- Console**: Shows the JavaScript console output.
- Sources**: Shows the source code of the page.
- Network**: Shows a list of network requests, including several image files being loaded.
- Performance**: Shows performance metrics for the page.
- Sign Up**: A button for signing up to the service.
- Log In**: A button for logging in.

The main content area displays the page's code and a preview of the image. The CSS panel contains the following rule:

```
img {background-color: #F1F1FA; width: 400px; height: 300px; display: block; margin: 10px auto; border: 0;}
```

The JS panel contains the following code:

```
document.addEventListener('DOMContentLoaded', function() {
  var lazyloadImages = new IntersectionObserver(function(entries, observer) {
    if ('IntersectionObserver' in window) {
      lazyloadImages = document.querySelectorAll('.lazy');
      var imageObserver = new IntersectionObserver(function(entries, observer) {
        entries.forEach(function(entry) {
          if (entry.isIntersecting) {
            var img = entry.target;
            img.src = img.getAttribute('data-src');
            img.classList.remove('lazy');
            observer.unobserve(img);
          }
        });
      });
      lazyloadImages.forEach(function(img) {
        imageObserver.observe(img);
      });
    }
  });
});
```

The preview area shows a large tree on a grassy field at sunset, which is the image being lazy-loaded.

On the right side, the Styles tab shows the CSS rule for the tree image:

```
img {
  background-color: #F1F1FA;
  width: 400px;
  height: 300px;
  display: block;
  margin: 10px auto;
  border: 0;
}
```

The Network tab shows the following requests:

Name	Status	Type	Initiator	Size	T..	Waterfall
https://ik.imagekit.io/demo/img/image1.jpg?tr=w-400,h-300%	20 ms	Image	https://ik.imagekit.io/demo/img/image1.jpg?tr=w-400,h-300%	~400B	40 ms	
https://ik.imagekit.io/demo/img/image2.jpg?tr=w-400,h-300%	40 ms	Image	https://ik.imagekit.io/demo/img/image2.jpg?tr=w-400,h-300%	~400B	60 ms	
https://ik.imagekit.io/demo/img/image3.jpg?tr=w-400,h-300%	60 ms	Image	https://ik.imagekit.io/demo/img/image3.jpg?tr=w-400,h-300%	~400B	80 ms	
https://ik.imagekit.io/demo/img/image4.jpg?tr=w-400,h-300%	80 ms	Image	https://ik.imagekit.io/demo/img/image4.jpg?tr=w-400,h-300%	~400B	100 ms	

# Ленивая загрузка фоновых изображений CSS

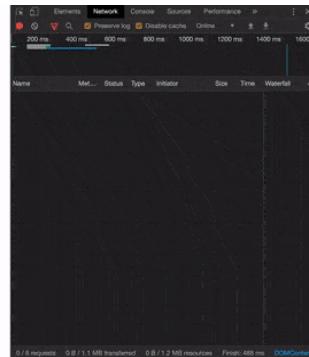
```
55  
56  
57     #bg-image.lazy {  
58         background-image: none;  
59         background-color: #F1F1FA;  
60     }  
61  
62     #bg-image {  
63         background-image: url('/path/image.jpg');  
64         max-width: 600px;  
65         height: 400px;  
66     }
```

```
57     lazyloadImages = document.querySelectorAll(".lazy");  
58     var imageObserver = new IntersectionObserver(function(entries, observer) {  
59         entries.forEach(function(entry) {  
60             if (entry.isIntersecting) {  
61                 var image = entry.target;  
62                 image.classList.remove("lazy");  
63                 imageObserver.unobserve(image);  
64             }  
65         });  
66     });  
67  
68     lazyloadImages.forEach(function(image) {  
69         imageObserver.observe(image);  
70     });  
71 
```

# Ленивая загрузка фоновых изображений CSS

# Использование JavaScript-библиотек

- [lazysizes](#)
- [lozad.js](#)
- [blazy](#)
- [yall.js](#)
- [react-lazyload](#)



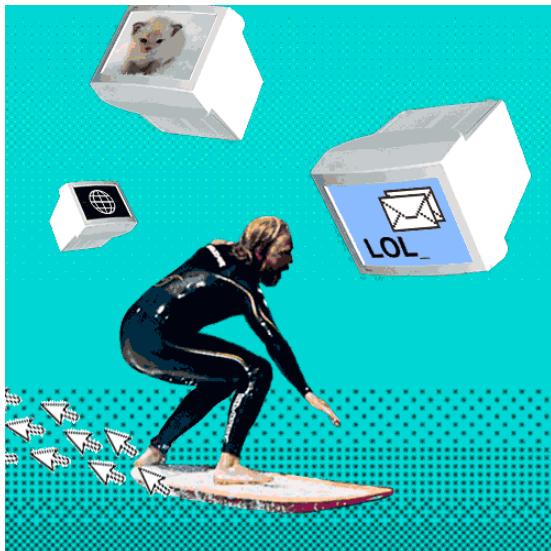
Большой вес библиотек с открытым исходным кодом ухудшает скорость загрузки страниц

# Предосторожности при использовании ленивой загрузки

- Избегайте отложенной загрузки изображений в области видимости.
- Не применяйте её для главной картинки на странице.
- Начинайте загрузку следующих изображений сразу после того, как на экране отобразилось предыдущее.



# Краткие итоги



Выбирайте правильный формат изображений, подходящий под ваши цели. Не шакальте фото!



Используйте изображения подходящего размера и разрешения под разные устройства.



Не ленитесь использовать ленивую загрузку для вашего контента, выбирайте подходящий вариант.



Не забывайте про ограничения поддержки разными браузерами, тестируйте ваши интерфейсы!

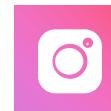
*Всем быстрых сайтов!*



# Контакты



@weirddark



weirddark\_



*Подлевских Милена  
Сергеевна*

Инженер-разработчик клиентских  
приложений в Топ-100

