

BREAKPOINTS



СЕМЕН БОЙКО



СЕМЕН БОЙКО

Front-end разработчик



simonderus@gmail.com



fb.me/simonboycko

ПЛАН ЗАНЯТИЯ

1. Выбор breakpoints
2. Mobile first
3. Единицы CSS, зависимые от размера окна браузера (viewport)
4. Адаптивные изображения с помощью `srcset`

ВЫБОР BREAKPOINTS

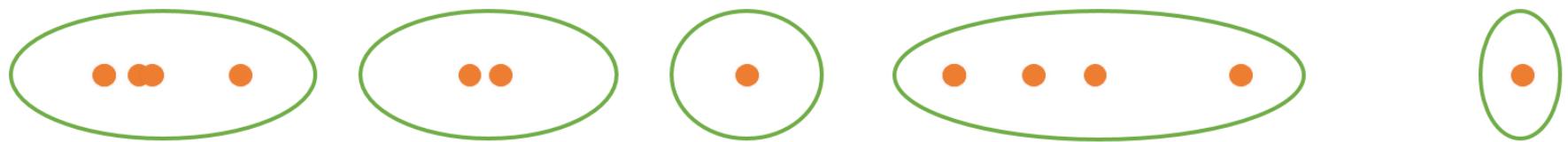
РАЗРОЗНЕННЫЕ ТОЧКИ

Мы видим множество точек, некоторые из которых расположены ближе друг к другу, некоторые – дальше.



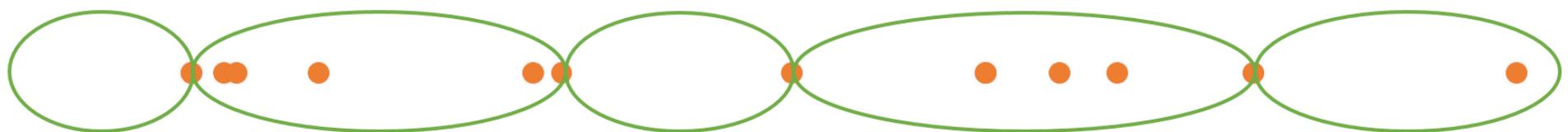
РАЗОБЬЕМ НА ГРУППЫ

Попробуем разбить это множество точек на группы. Самый простой способ – обвести кружками точки, находящиеся близко друг к другу:



ДРУГОЙ СПОСОБ ГРУППИРОВКИ

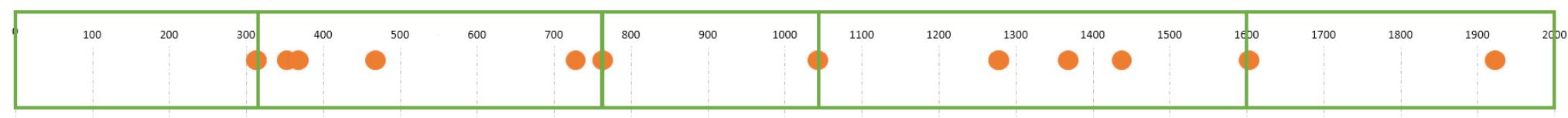
А что, если поделить точки на группы таким образом?



ОПРЕДЕЛЕНИЕ БРЕЙКПОИНТОВ

Вы можете возразить, что это странный способ деления на группы, и будете правы.

Именно так чаще всего поступают, когда стоит задача выбора breakpoints (брейкпоинты) в адаптивной верстке:





Breakpoint – условие, при котором раскладка сайта меняется с одной на другую. Характерно для медиавыражений.

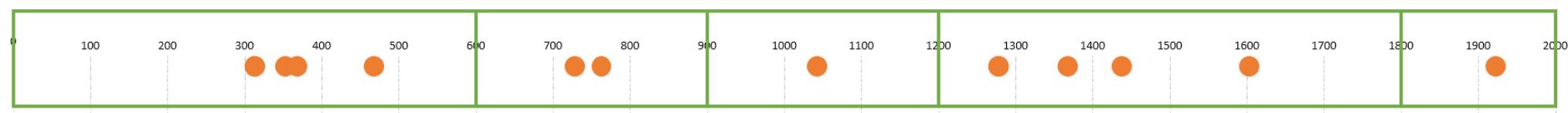
Чаще всего брейкпоинты выбираются в соответствии с размерами экранов популярных устройств – 1024px, 768px, 480px и другими.



ПРАВИЛЬНЫЙ СПОСОБ

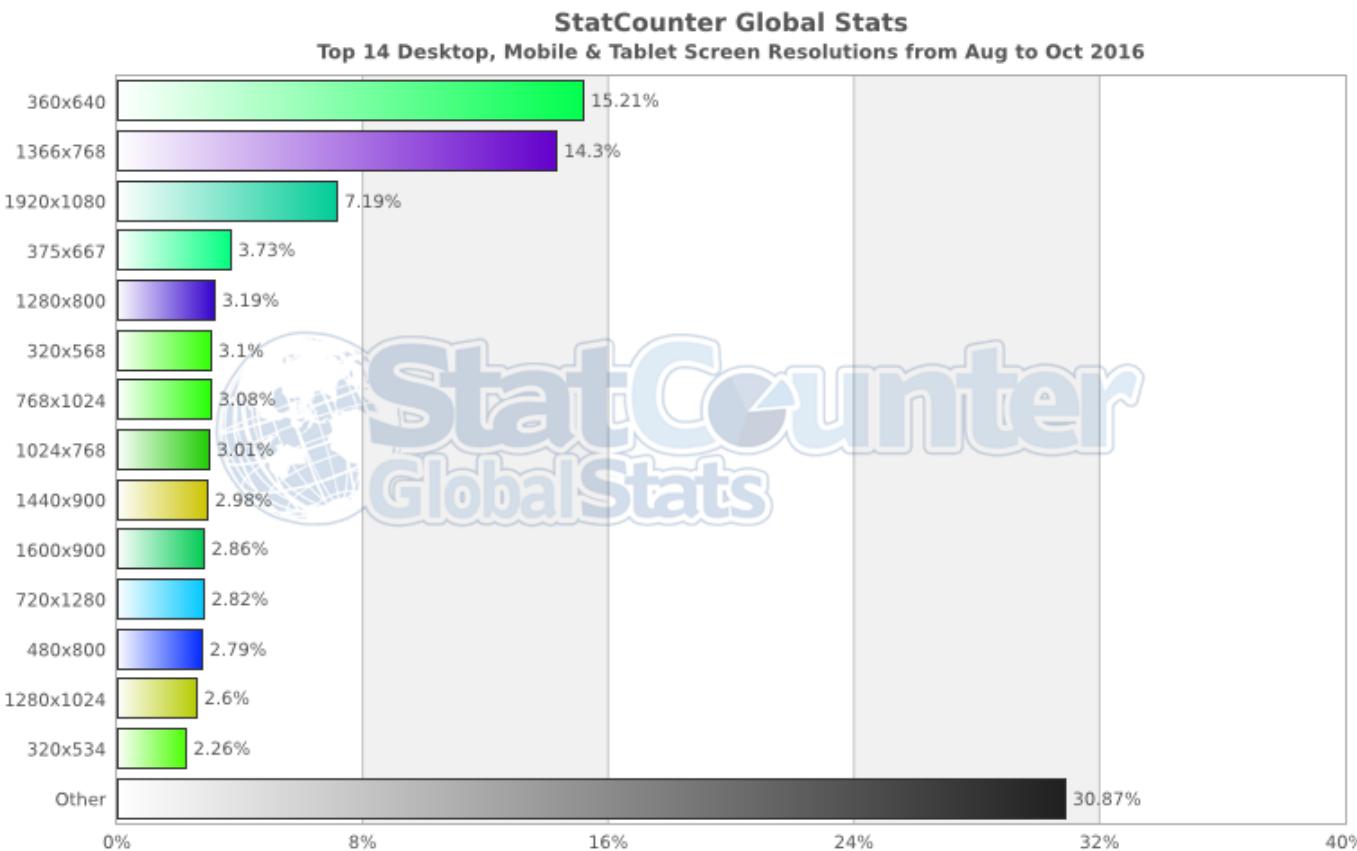
Превратим линии, которыми мы обвели группы точек на рисунке выше, в пограничные линии.

Мы получили точки 600px, 900px, 1200px и 1800px на тот случай, если в дизайн-макете есть что-то особенное для огромных экранов.



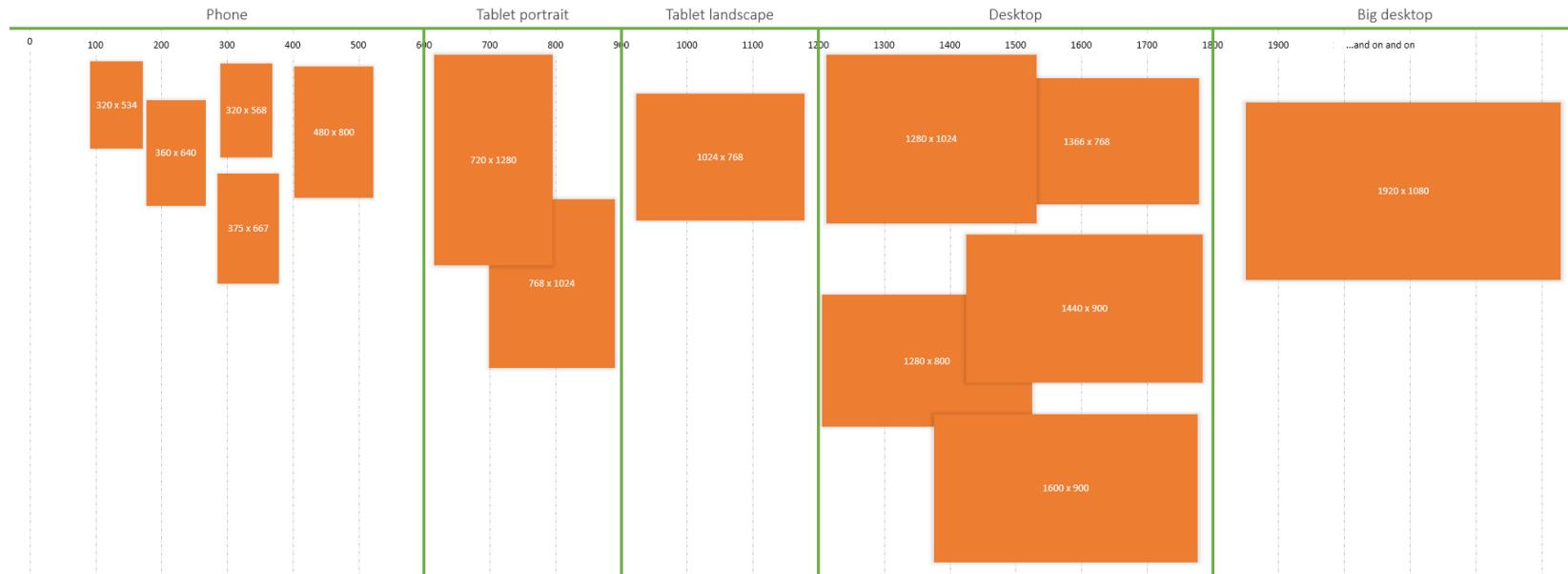
СТАТИСТИКА ПОПУЛЯРНЫХ УСТРОЙСТВ

В получившиеся диапазоны попадут 14 самых популярных устройств,
согласно статистике:



РАСПРЕДЕЛЕНИЕ УСТРОЙСТВ

После того, как определены границы, мы можем нарисовать простую схему, на которой устройства располагаются в границах брейкпоинтов в соответствии с их размерами экранов:



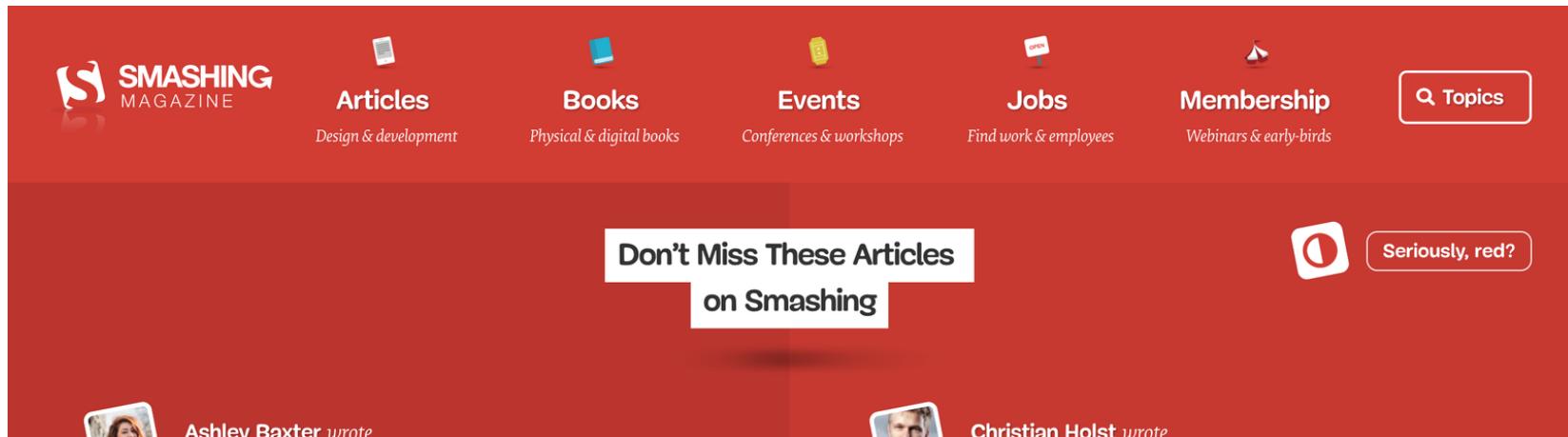
Вместо того, чтобы выбирать какие-то конкретные точки и пользоваться всегда только ими, имеет смысл исходить из особенностей дизайн-макета в каждом конкретном случае.

ОРИЕНТИРУЕМСЯ НА РЕЗУЛЬТАТ

Например, если под какой-то блок не хватает места уже на ширине 1349px, имеет смысл написать особый медиа-запрос для него с breakpoint в этой точке, а не в точке 900.

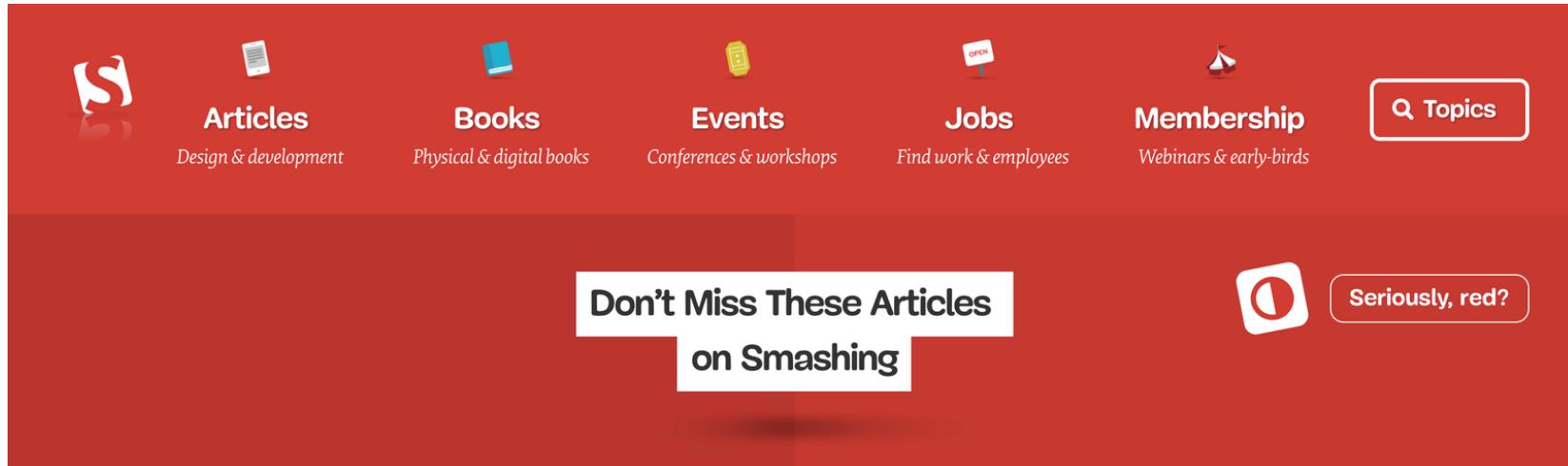
ПРИМЕР BREAKPOINTS

На сайте smashingmagazine.com на ширине 1350px лого слева выводится полностью:



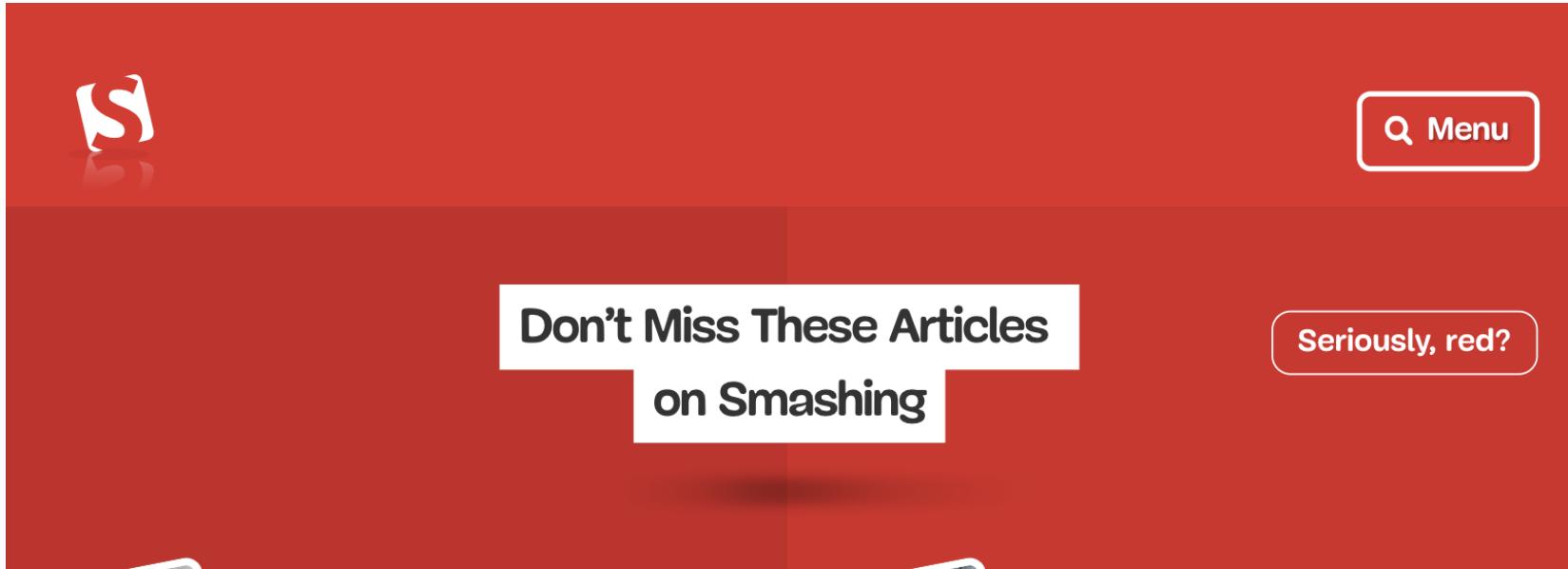
ПРИМЕР BREAKPOINTS

На ширине 1349px лого слева выводится в сокращенном виде:



МЕНЮ СКРЫЛОСЬ

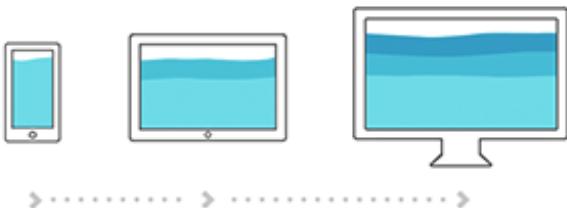
На ширине 900px меню прячется и появляется кнопка Меню :



MOBILE FIRST



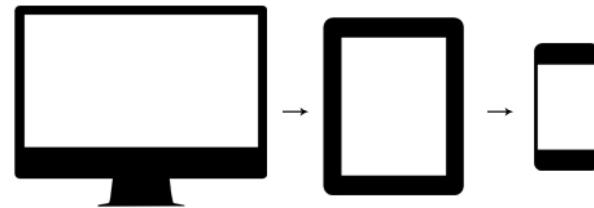
Mobile first – концепция, согласно которой предлагается сначала создавать дизайн для устройств с маленькими экранами – мобильных телефонов, и затем усложнять его для устройств с большими экранами, таких как ноутбуки и настольные компьютеры.



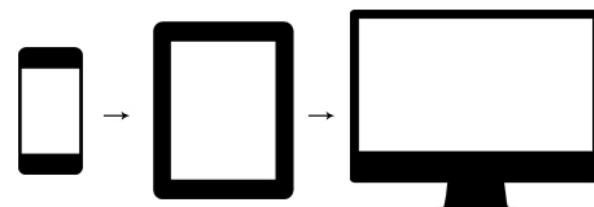
Такой подход означает отказ от привычной концепции создания дизайна для устройств с большими экранами и дальнейшим упрощением дизайна для мобильных устройств.

РАЗНИЦА ПОДХОДОВ

Подход desktop first:



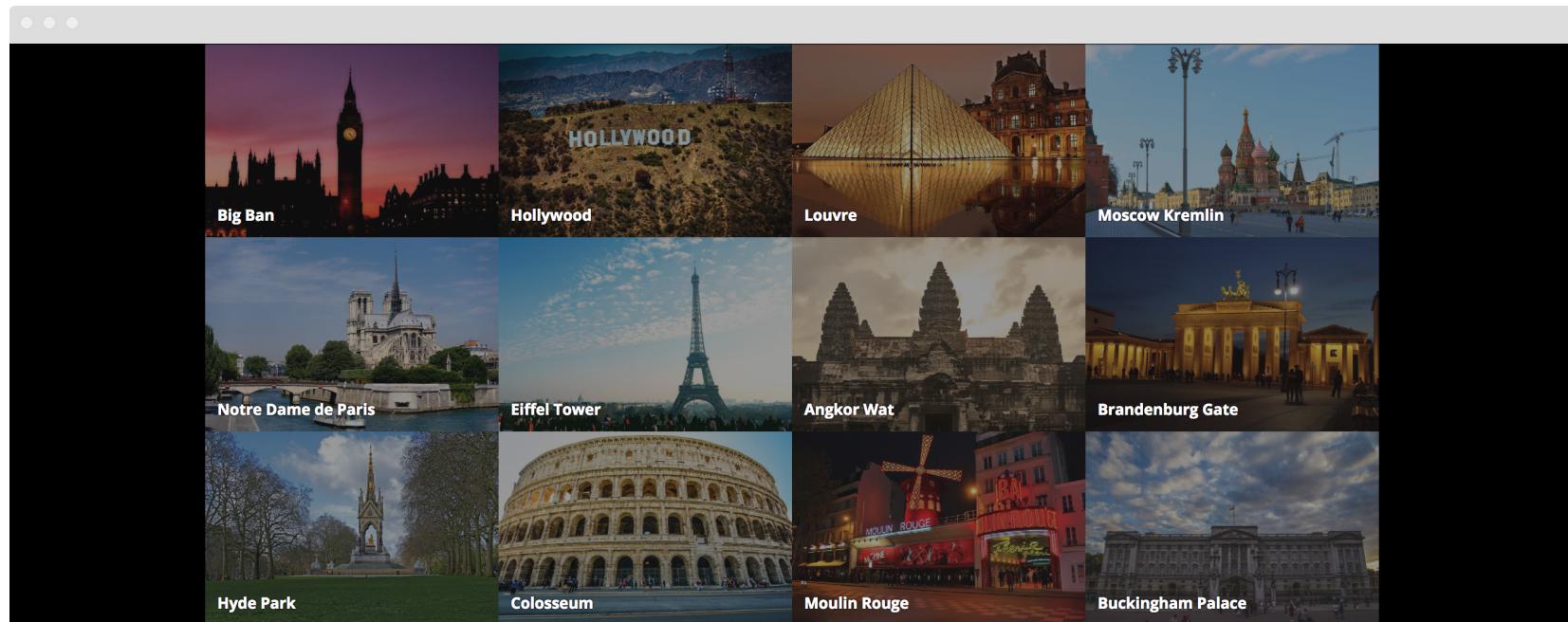
Подход mobile first:



КАРТОЧКИ ДОСТОПРИМЕЧАТЕЛЬНОСТЕЙ

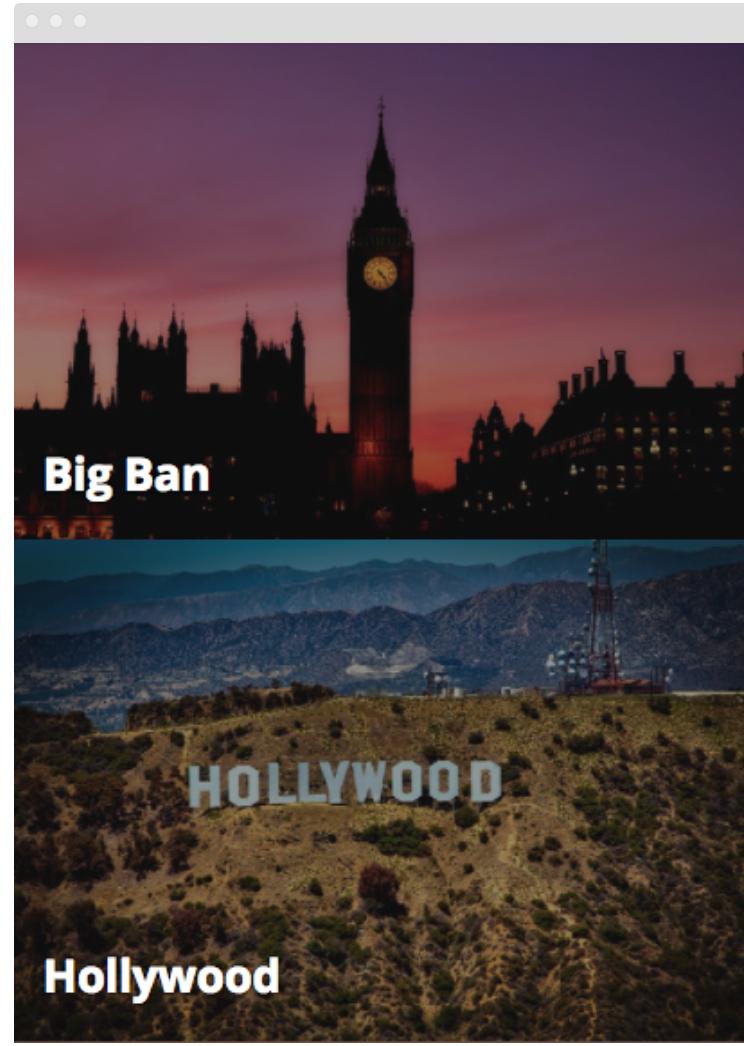
Мы получили макеты блока с карточками достопримечательностей для верстки.

Блок при ширине окна браузера большей или равной 1200px
(максимальная ширина блока равна 1920px):



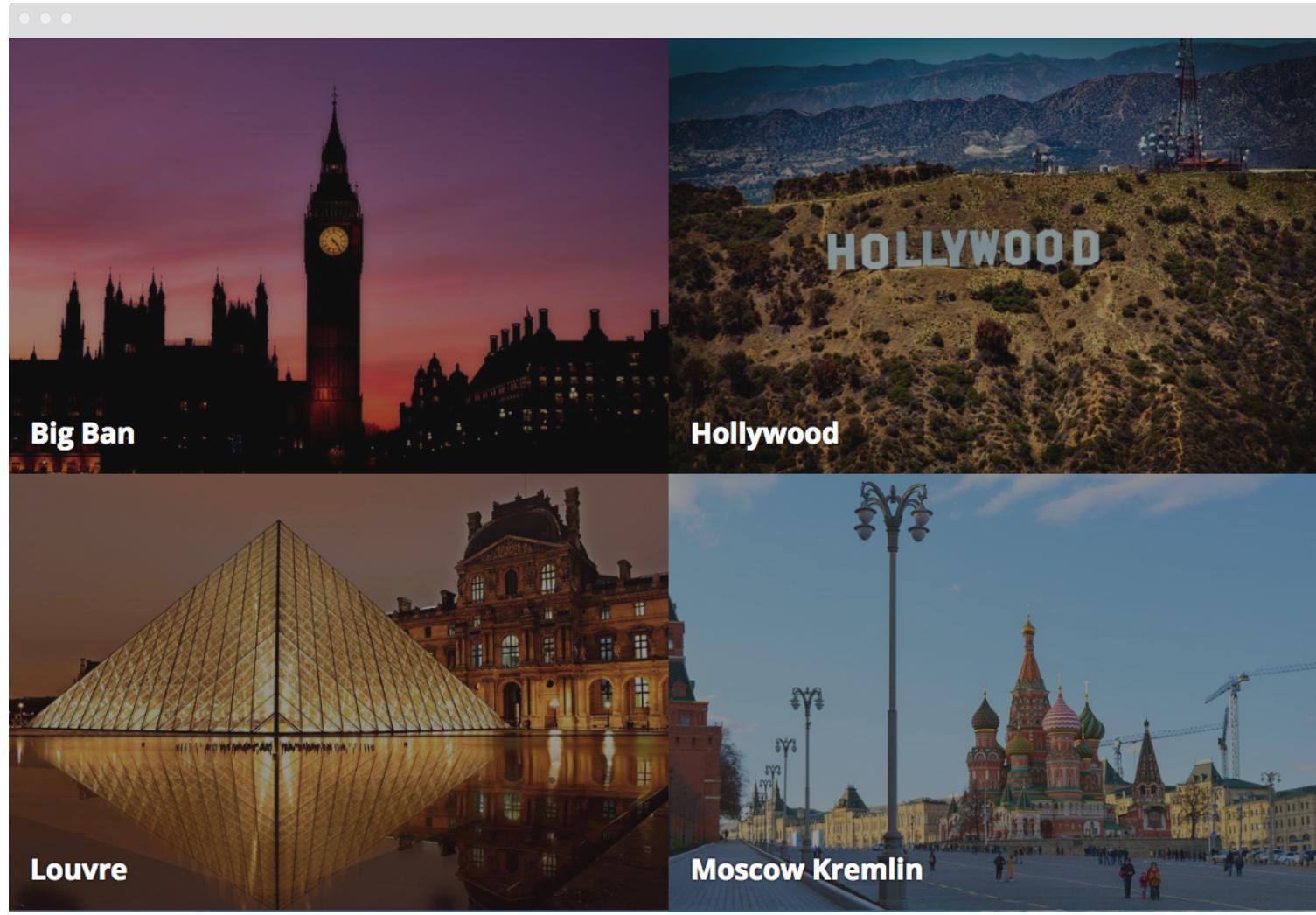
КАРТОЧКИ НА МОБИЛЬНОМ

Блок при ширине окна браузера < 480px:



ПРОМЕЖУТОЧНЫЙ ВАРИАНТ

Блок при ширине окна от 480px до 1199px:



CSS-КОД ПО СТАРИНКЕ

```
1 .container {  
2     max-width: 1920px;  
3     margin-left: auto;  
4     margin-right: auto;  
5     display: flex;  
6     flex-wrap: wrap;  
7 }  
8  
9 @media (min-width: 1200px) {  
10     .card {  
11         width: 25%;  
12     }  
13 }  
14  
15 @media (min-width: 480px) and (max-width: 1199px) {  
16     .card {  
17         width: 50%;  
18     }  
19 }  
20  
21 @media (max-width: 479px) {  
22     .card {  
23         width: 100%;  
24     }  
25 }
```

НАЧНЕМ С МОБИЛЬНЫХ

Можем убрать `display: flex` по умолчанию. Тогда блоки будут занимать 100% ширины родителя:

```
1 .container {  
2   max-width: 1920px;  
3   margin-left: auto;  
4   margin-right: auto;  
5 }  
6  
7 @media (min-width: 480px) {  
8   .container {  
9     display: flex;  
10    flex-wrap: wrap;  
11  }  
12 }  
13  
14 @media (min-width: 1200px) {  
15   .card {  
16     width: 25%;  
17   }  
18 }  
19  
20 @media (min-width: 480px) and (max-width: 1199px) {  
21   .card {  
22     width: 50%;  
23   }  
24 }
```

БОЛЬШЕ ОПТИМИЗАЦИИ!

Максимальная ширина `.container` и `margin`, который располагает блок по центру экрана, нужны нам только на большем из диапазонов:

```
1 @media (min-width: 480px) {  
2     .container {  
3         display: flex;  
4         flex-wrap: wrap;  
5     }  
6 }  
7  
8 @media (min-width: 1200px) {  
9     .container {  
10         max-width: 1920px;  
11         margin-left: auto;  
12         margin-right: auto;  
13     }  
14  
15     .card {  
16         width: 25%;  
17     }  
18 }  
19  
20 @media (min-width: 480px) and (max-width: 1199px) {  
21     .card {  
22         width: 50%;  
23     }  
24 }
```

НАВОДИМ ПОРЯДОК

```
1 @media (min-width: 480px) {  
2     .container {  
3         display: flex;  
4         flex-wrap: wrap;  
5     }  
6 }  
7  
8 @media (min-width: 480px) and (max-width: 1199px) {  
9     .card {  
10         width: 50%;  
11     }  
12 }  
13  
14 @media (min-width: 1200px) {  
15     .container {  
16         max-width: 1920px;  
17         margin-left: auto;  
18         margin-right: auto;  
19     }  
20     .card {  
21         width: 25%;  
22     }  
23 }
```

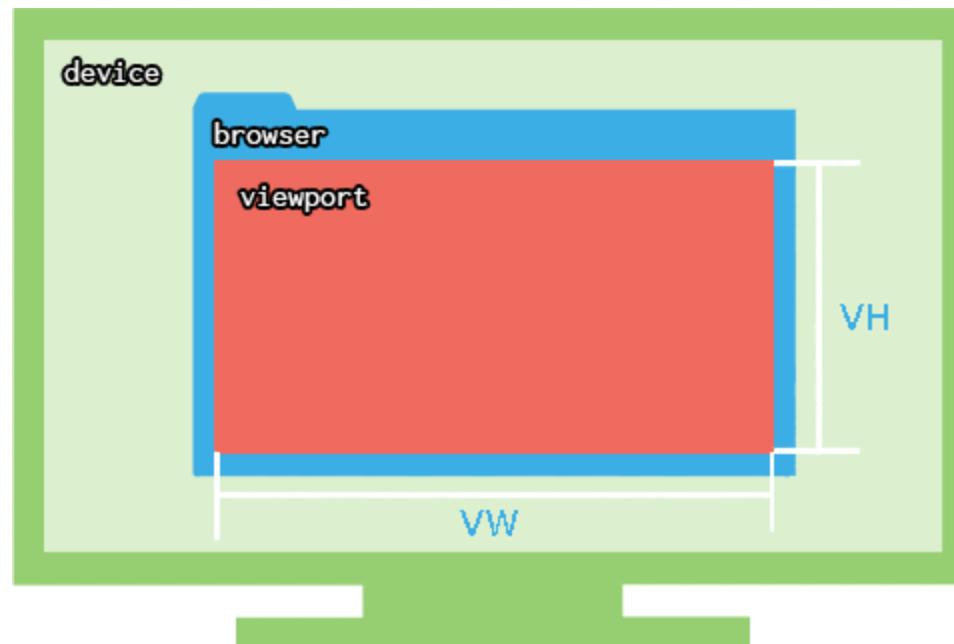
[Live Demo](#)

ЕДИНИЦЫ CSS, ЗАВИСИМЫЕ ОТ РАЗМЕРА ОКНА БРАУЗЕРА (VIEWPORT)

НОВЫЕ ЕДИНИЦЫ ИЗМЕРЕНИЯ

В CSS есть единицы, позволяющие задавать зависимость от размеров окна браузера или **viewport**.

Они удобны, когда есть сложности с использованием других относительных единиц.



ЭКРАН ПРИВЕТСТВИЯ

Решим задачу верстки экрана приветствия, который должен занимать весь экран устройства. Напишем разметку:

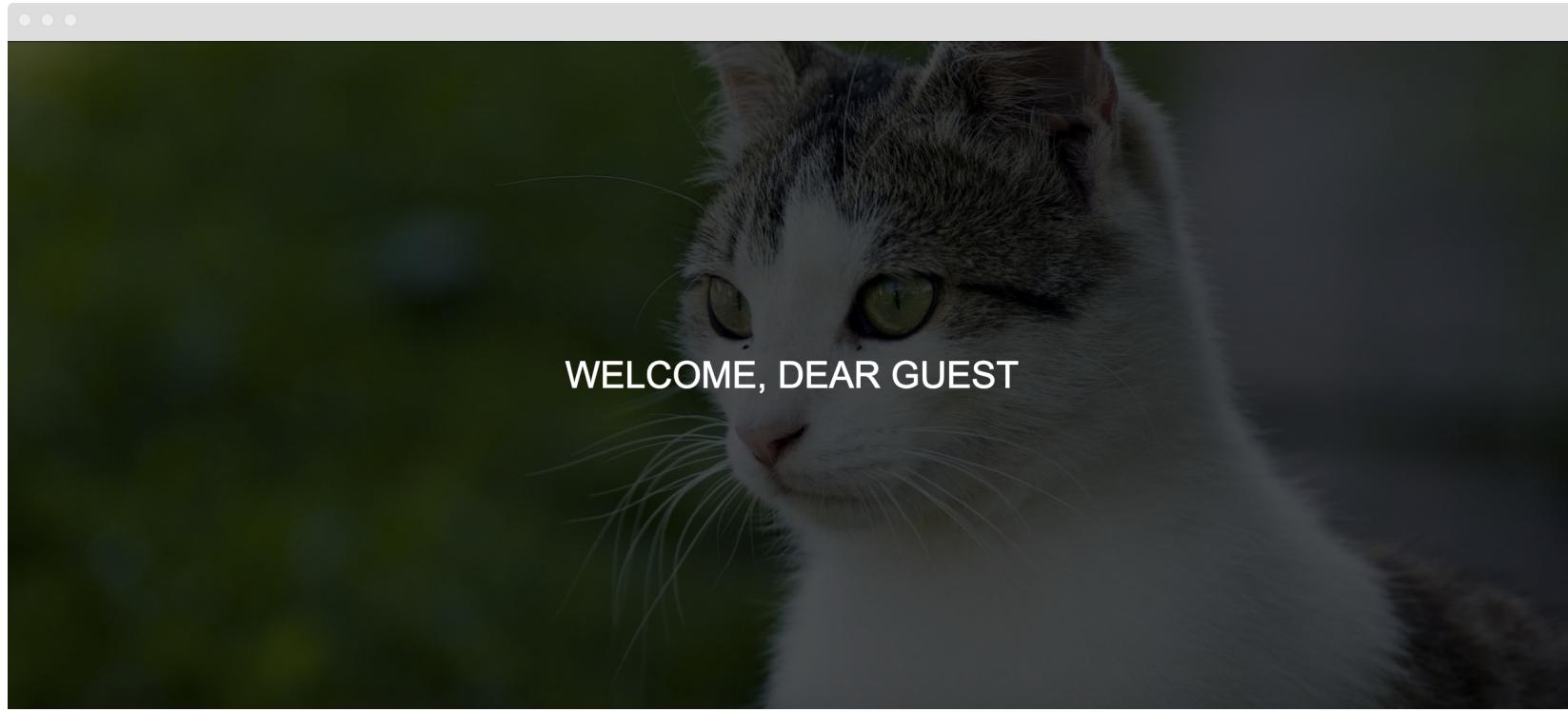
```
1 <body>
2   <div class="hero">
3     <div class="hero__container">
4       <span class="hero__label">
5         Welcome, dear guest
6       </span>
7     </div>
8   </div>
9 </body>
```

ИСПОЛЬЗУЕМ %

```
1 body, html {  
2     height: 100%;  
3 }  
4  
5 .hero {  
6     width: 100%;  
7     height: 100%;  
8     position: relative;  
9     background: url("cat.jpg") no-repeat  
10    50% 50% / cover #000000;  
11 }
```

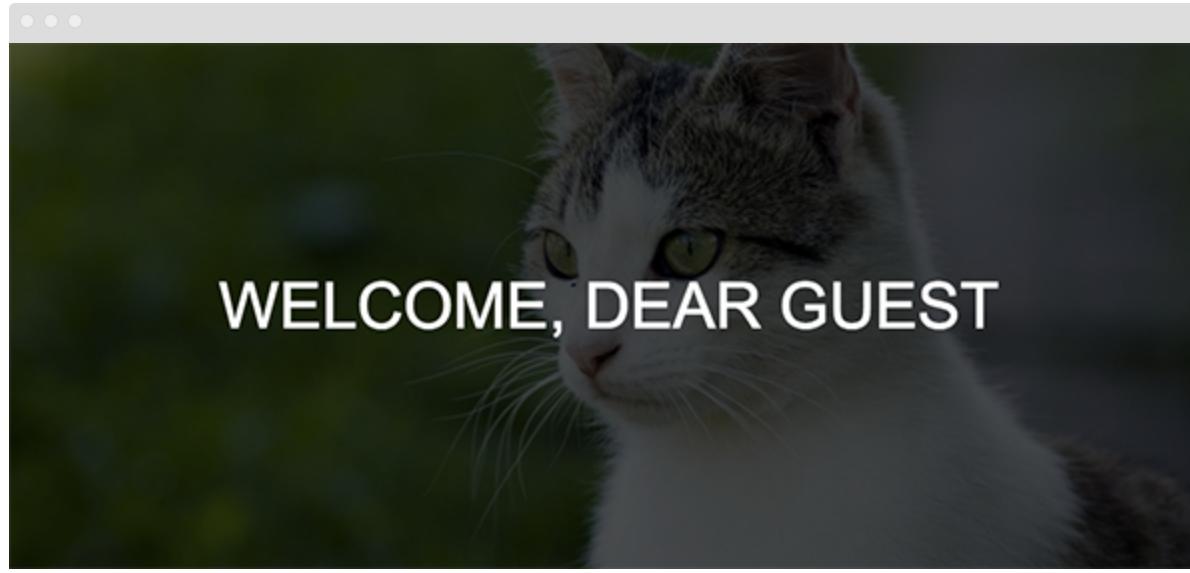
ЖЕЛАЕМЫЙ РЕЗУЛЬТАТ

Верстка при ширине экрана 1280px:



И НА МОБИЛЬНОМ ТОЖЕ

Верстка на мобильном телефоне в ландшафтной ориентации:



НЕДОСТАТОК

Однако, это решение имеет недостатки – например, если экран будет не единственным, получится, что высота элементов `html` и `body` будет меньше, чем высота контента, что может привести к ошибкам.

С помощью единиц `vh` и `vw` мы можем избавить наше решение от недостатков.

КАК РАБОТАЮТ `vh` / `vw`

`1vh` равен 1% от высоты окна браузера, `1vw` – 1% от ширины окна браузера.

Важно: за ширину окна браузера принимается ширина без полосы прокрутки.

ИЗМЕНЯЕМ CSS

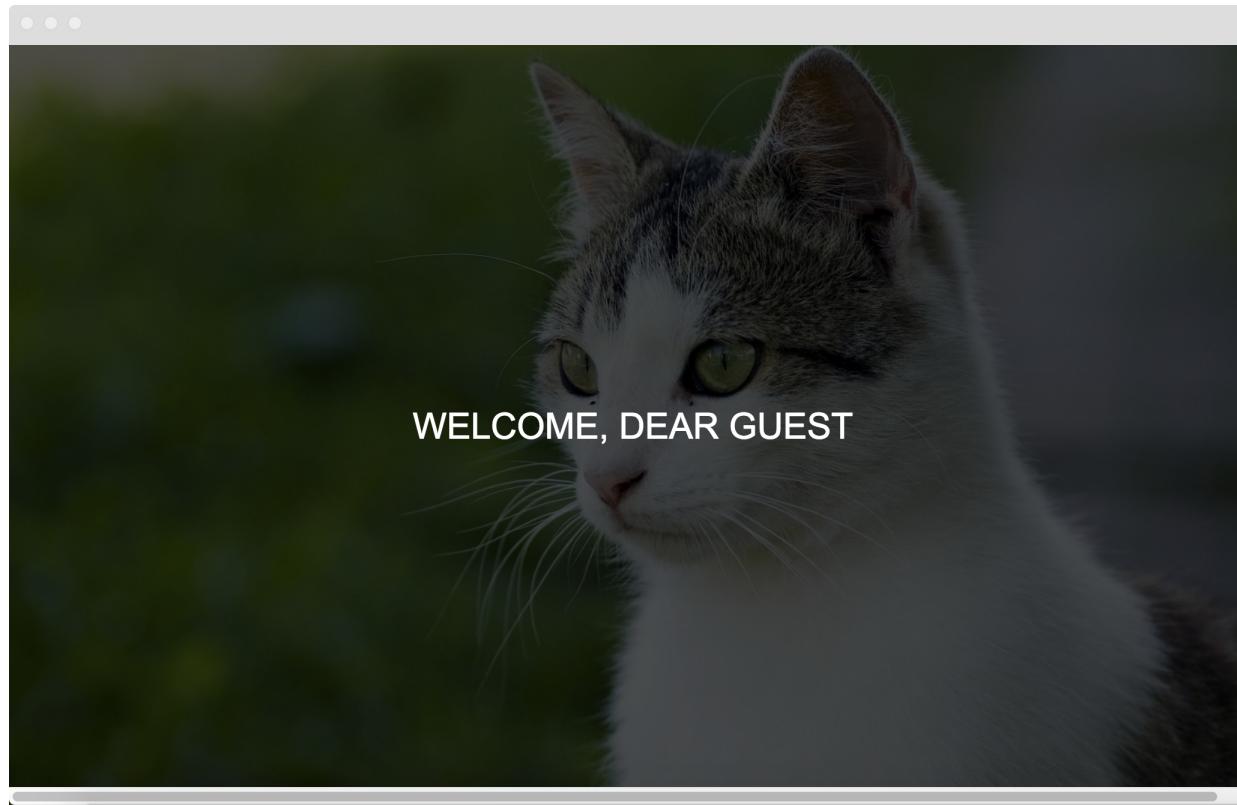
```
1 .hero {  
2     width: 100vw;  
3     height: 100vh;  
4     background: url("cat.jpg") no-repeat  
5         50% 50%/cover #000000;  
6     display: flex;  
7 }
```

[Live Demo](#)

Теперь ширина и высота блока `.hero` зависят только от ширины и высоты окна браузера, а не ширины и высоты родительского элемента. Все работает как раньше, но кода стало меньше.

ПРОБЛЕМА СО СКРОЛЛОМ

Если у блока задана минимальная высота, которая больше высоты окна, то блок `.hero` перестает помещаться в окно браузера по ширине из-за скролла, и помимо вертикальной полосы появится еще и горизонтальная полоса прокрутки:



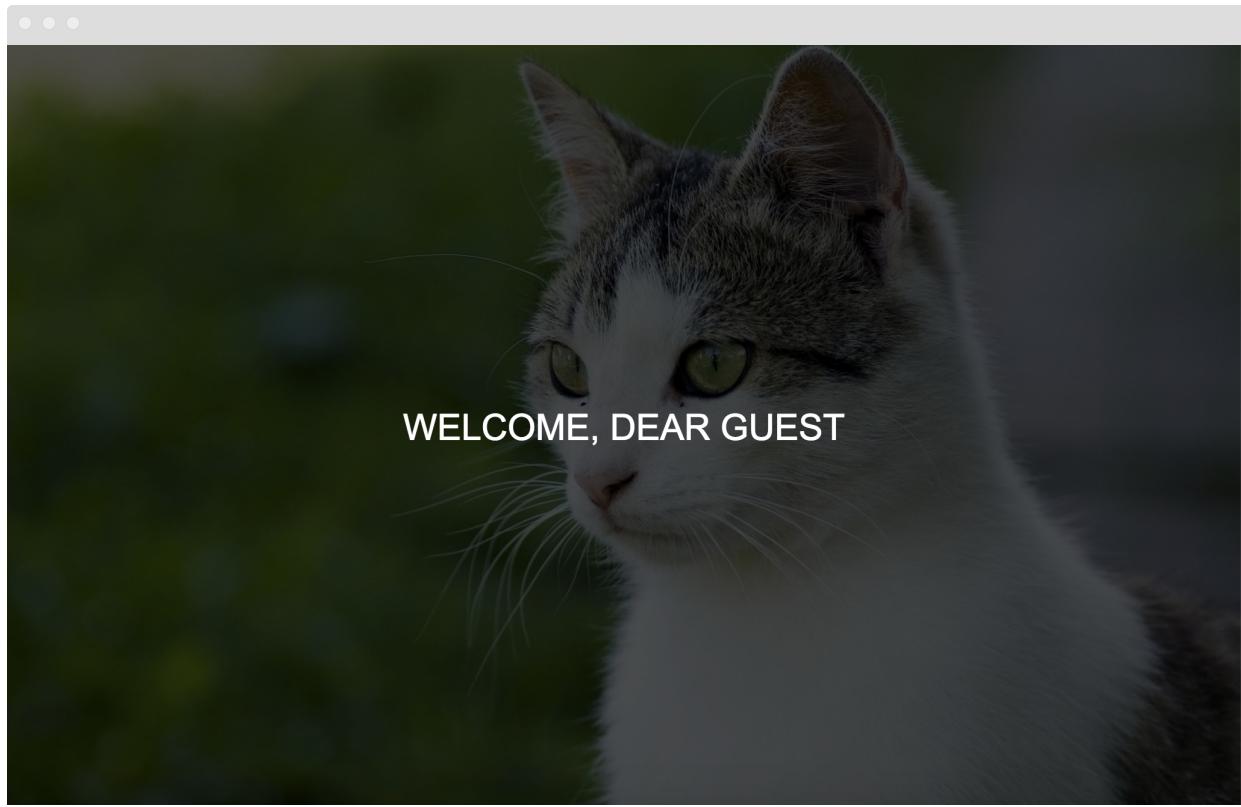
% ИНОГДА ЛУЧШЕ

Чтобы избежать появления горизонтальной прокрутки при наличии вертикальной, лучше не указывать ширину блока в `vw`, а оставить ее в %:

```
1 .hero {  
2   width: 100%;  
3   height: 100vh;  
4   background: url("cat.jpg") no-repeat  
5     50% 50%/cover #000000;  
6   display: flex;  
7 }
```

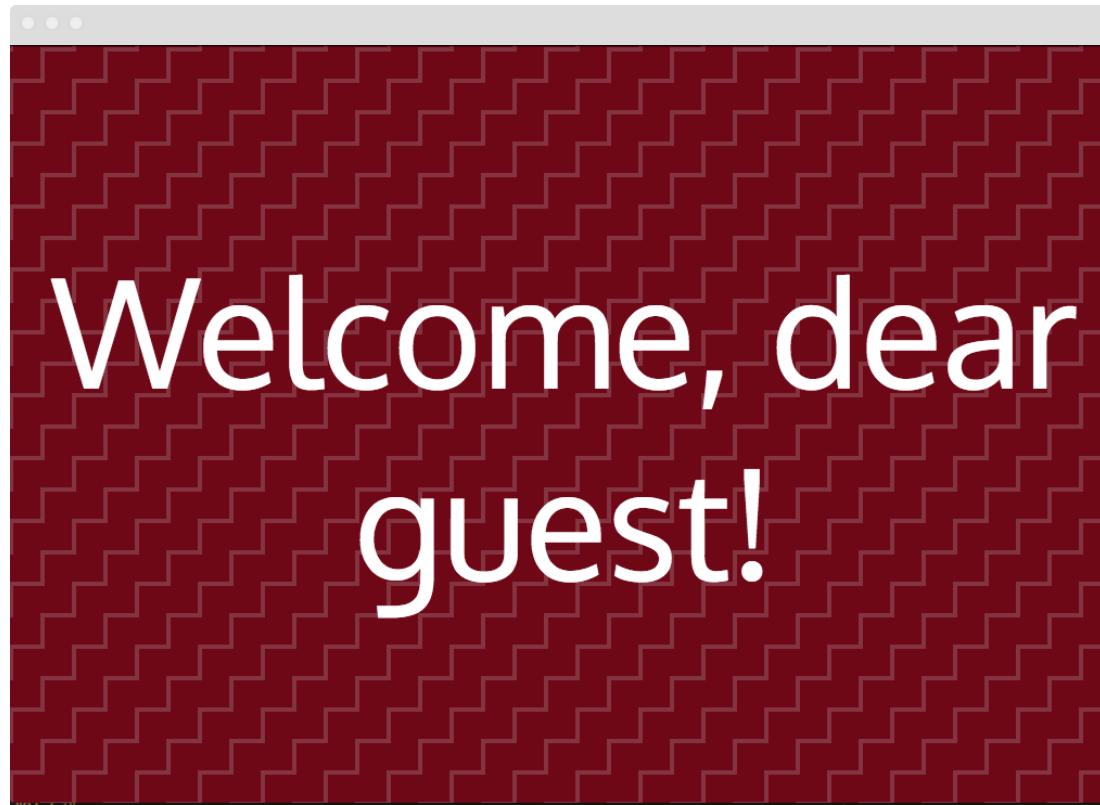
БЕЗ ГОРИЗОНТАЛЬНОГО СКРОЛЛА

В таком случае, даже если по вертикали блок не будет помещаться в окно – горизонтальной прокрутки у нас не появится:



БЛОК ПРИВЕТСТВИЯ

Единицы измерения, зависимые от параметров окна браузера, полезны при решении задач выбора размера шрифта, который был бы читаем на любом размере экрана. Рассмотрим решение такой задачи на примере блока приветствия:



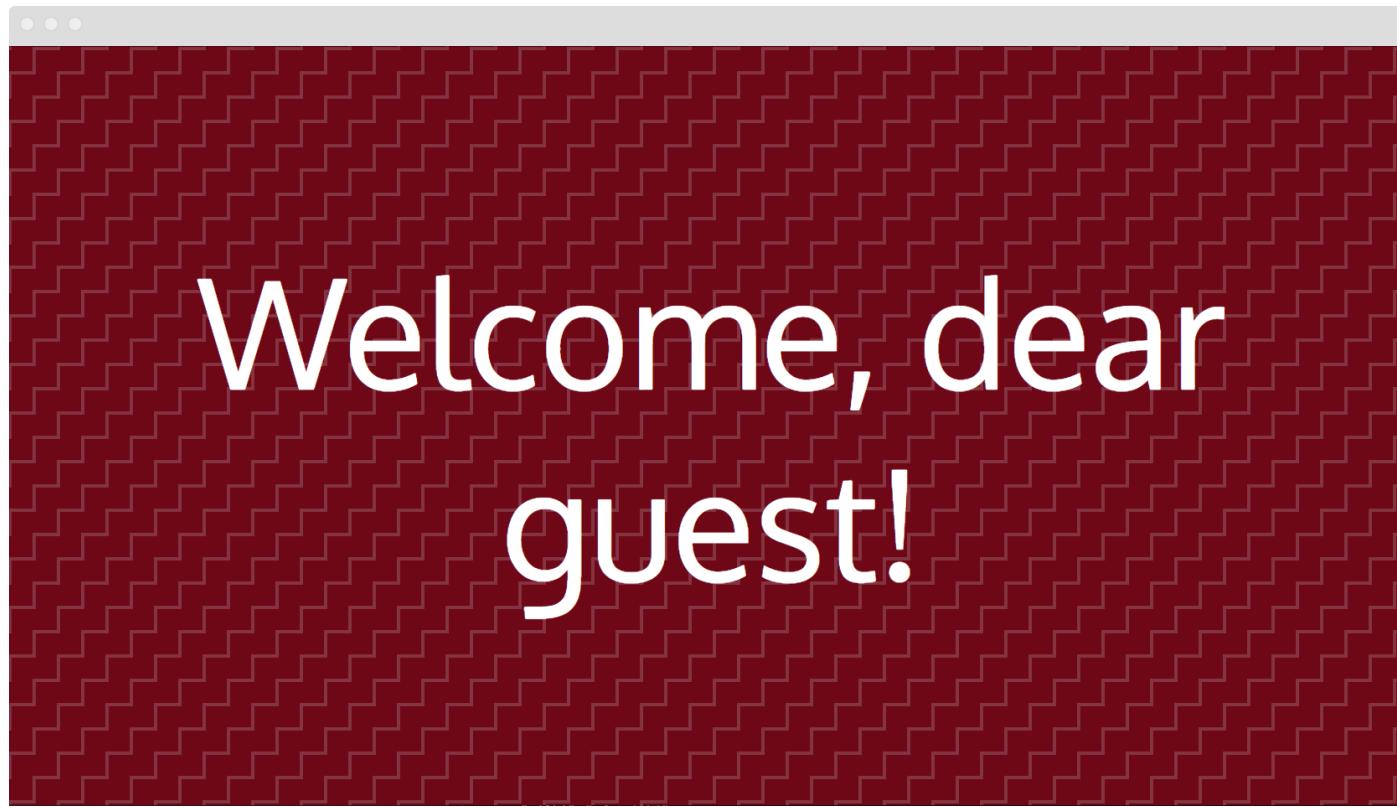
АДАПТИВНЫЙ ТЕКСТ

Чтобы размер шрифта надписи был не слишком крупным на широкоэкраных мониторах, и при этом не слишком мелким на телефонах, воспользуемся единицами `vh`:

```
1 .greeting__message {  
2   font-size: 20vh;  
3 }
```

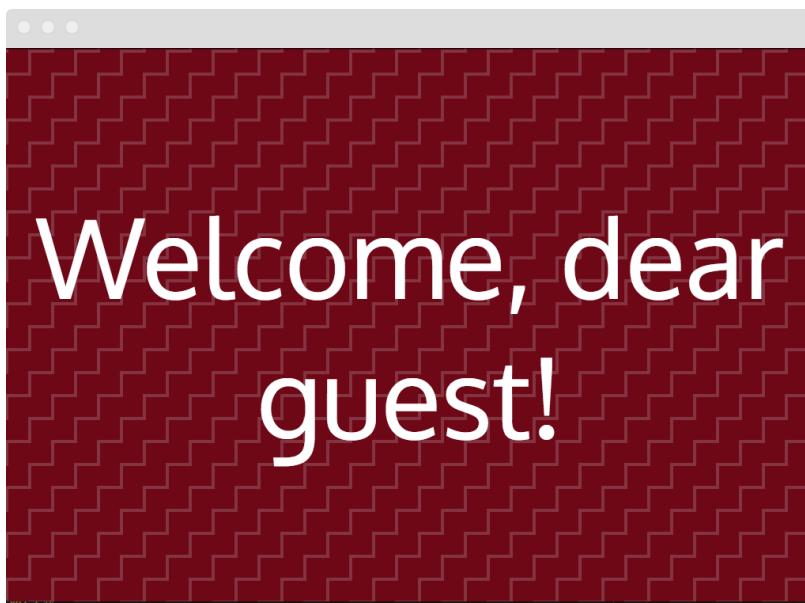
НАДПИСЬ НА БОЛЬШОМ ЭКРАНЕ

Приветствие в окне браузера 1680px*1050px:

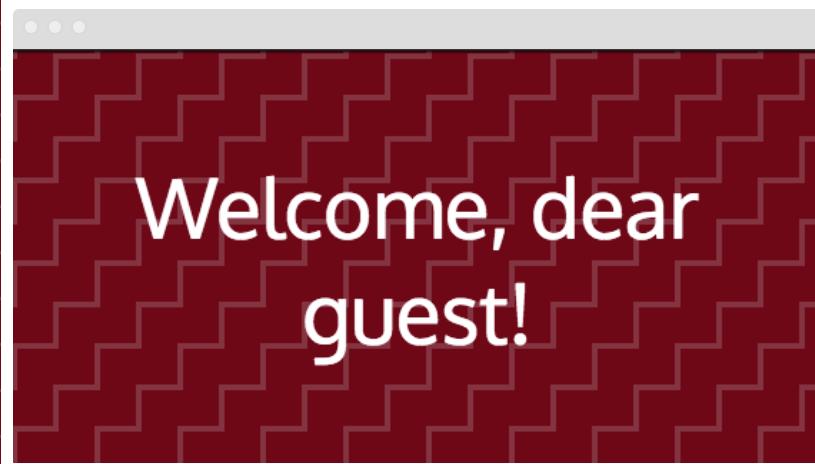


ХОРОШО НА ПЛАНШЕТЕ И ТЕЛЕФОНЕ

Приветствие на iPad в
горизонтальной ориентации:

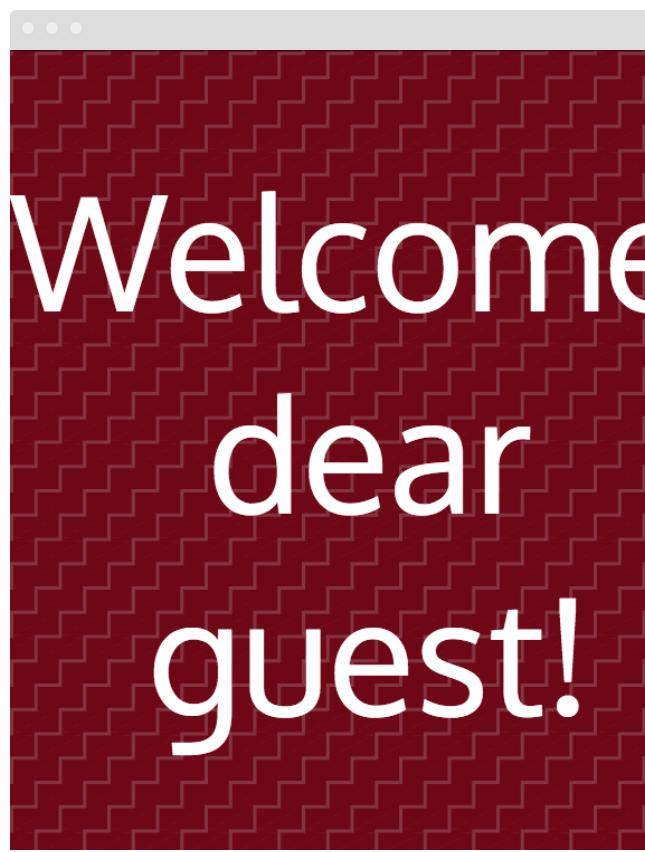


Приветствие на мобильном
телефоне в горизонтальной
ориентации:

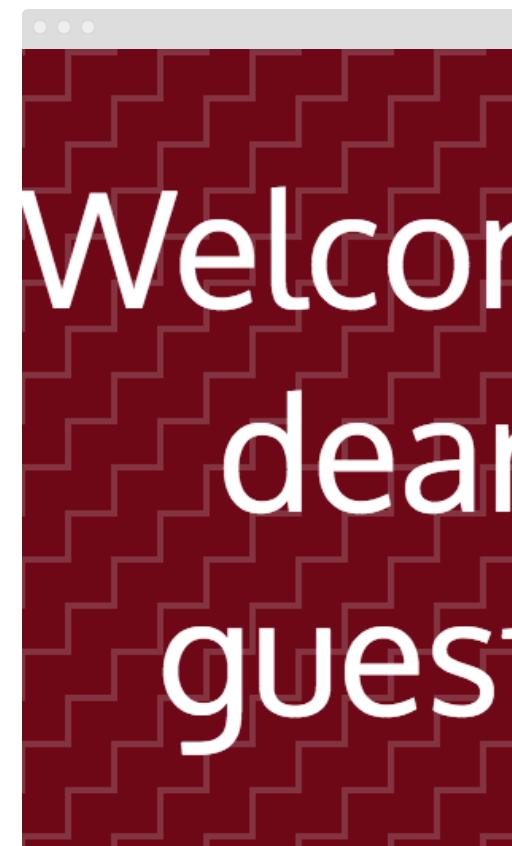


ВЕРТИКАЛЬНАЯ ОРИЕНТАЦИЯ

Приветствие на iPad в вертикальной
ориентации:



Приветствие на мобильном
телефоне в вертикальной
ориентации:



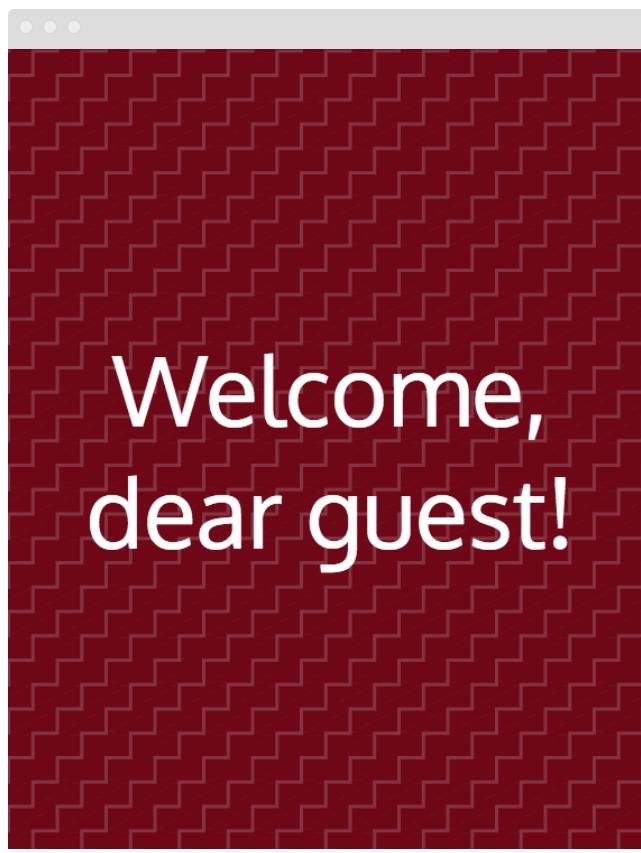
МЕНЯЕМ РАЗМЕР ШРИФТА

Чтобы сделать размер надписи не таким крупным в вертикальной ориентации, будем определять соотношение ширины и высоты окна браузера с помощью медиазапросов. И в случае вертикальной ориентации зададим размер шрифта не в зависимости от высоты, а в зависимости от ширины экрана с помощью единиц `vw`:

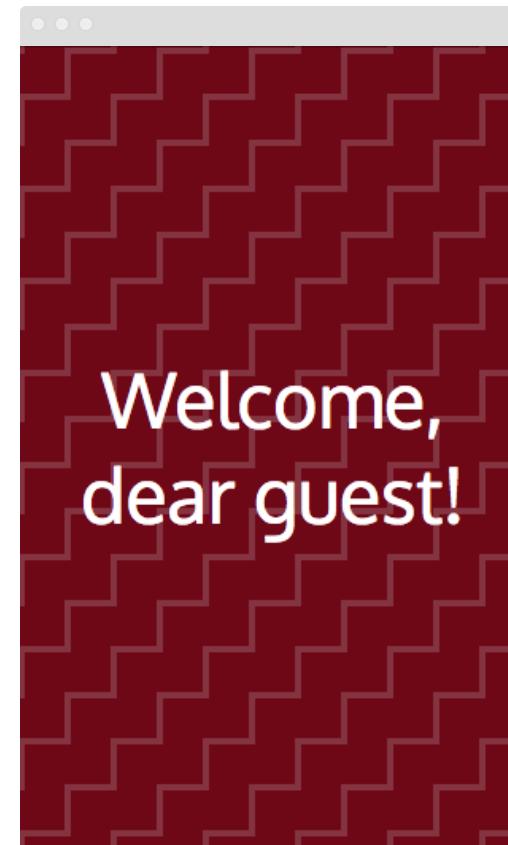
```
1 @media (orientation: landscape) {  
2     .greeting__message {  
3         font-size: 20vh;  
4     }  
5 }  
6  
7 @media (orientation: portrait) {  
8     .greeting__message {  
9         font-size: 15vw;  
10    }  
11 }
```

ОТЛИЧНЫЙ РЕЗУЛЬТАТ

Приветствие на iPad в вертикальной
ориентации:



Приветствие на мобильном
телефоне в вертикальной
ориентации:

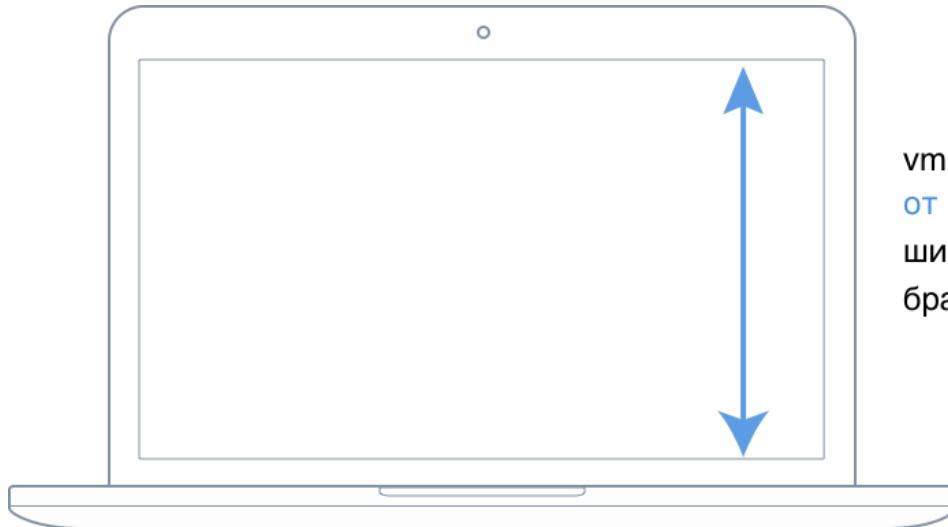


vmin / vmax

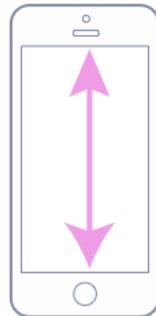
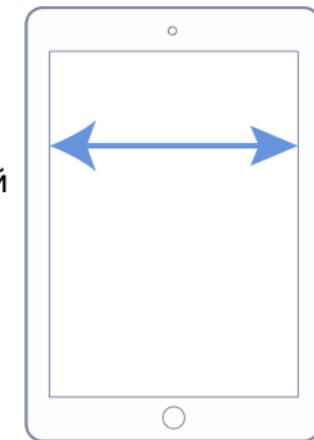
Единицы `vmin` и `vmax` рассчитываются по следующим правилам:

- `1vmin` равен 1% от меньшего из двух значений ширины или высоты окна браузера. Если устройство находится в вертикальной (портретной) ориентации, то меньшей из сторон окна будет ширина, в этом случае `1vmin = 1vw`. В горизонтальной (ландшафтной) ориентации меньшей будет высота окна, и, соответственно, в этом случае `1vmin = 1vh`.
- `1vmax` равен 1% от большего из двух значений ширины или высоты окна браузера. При вертикальной ориентации устройства `1vmax = 1vh`, при горизонтальной — `1vmax = 1vw`.

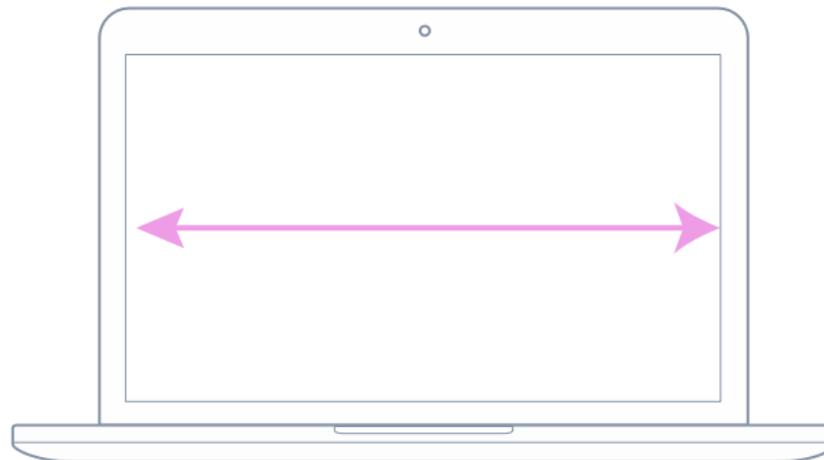
ПРИНЦИП РАБОТЫ В КАРТИНКАХ



vmin: рассчитывается
от меньшего из значений
ширины или высоты
браузера



vmax: рассчитывается
от большего из значений
ширины или высоты
браузера



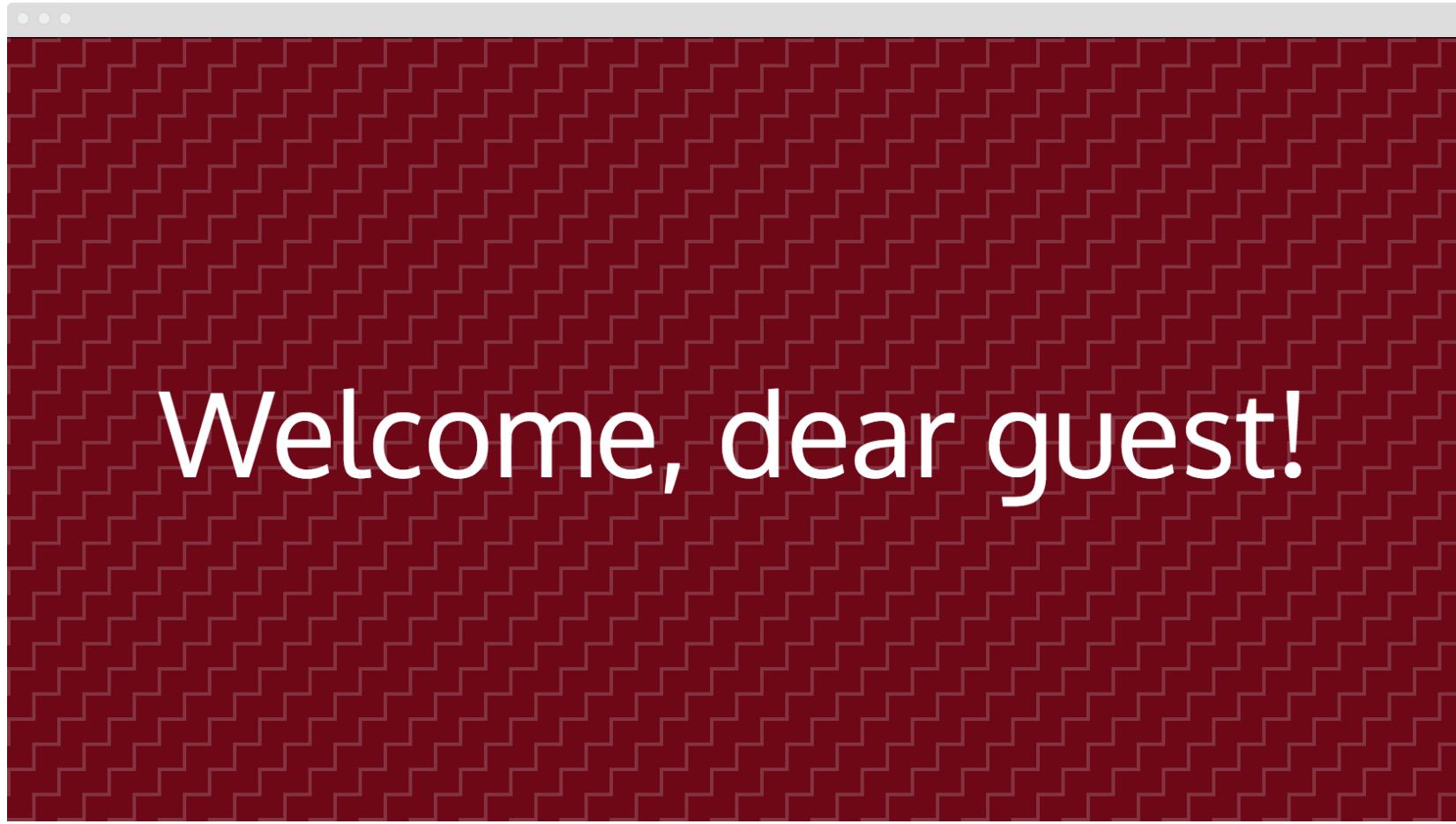
УБИРАЕМ ЛИШНИЙ КОД

Уберем медиазапросы из нашего кода и используем `vmin` для задания размера шрифта приветственной надписи в блоке:

```
1 .greeting__message {  
2   font-size: 15vmin;  
3 }
```

Медиазапросы в этом решении больше не нужны, потому что при вычислении `15vmin` браузер сначала определит, какая из сторон его окна (ширина или высота) меньше, и только после этого, взяв за 100% найденную сторону, рассчитает значение в px.

ОТЛИЧНО НА ШИРОКИХ МОНИТОРАХ



[Live Demo](#)

АДАПТИВНЫЕ ИЗОБРАЖЕНИЯ С ПОМОЩЬЮ `srcset`

НЕЧЕТКАЯ КОНТЕНТНАЯ КАРТИНКА

Ранее мы уже сталкивались с проблемой нечеткости фоновых изображений на retina-дисплеях, разбирались в причинах и знаем решение. Но как же поступить в случае, когда у нас есть контентная картинка, сверстанная тегом `img`, и она выглядит нечетко на дисплеях повышенной четкости?

БАННЕР

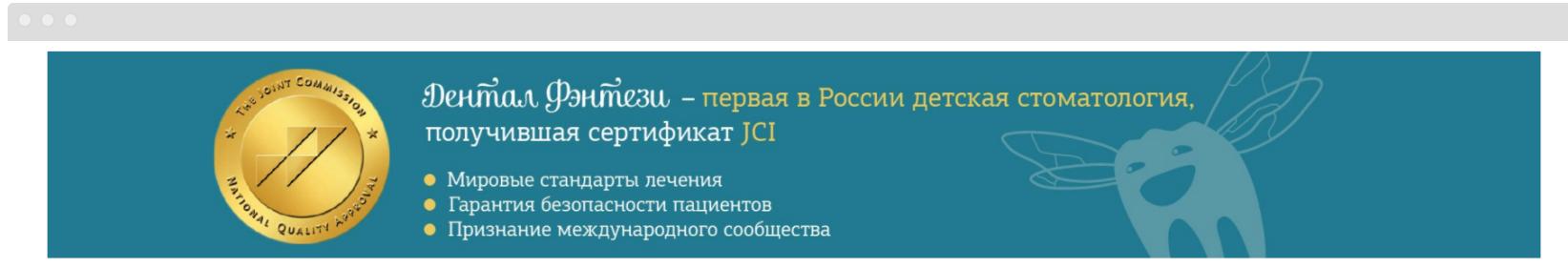
Сверстаем баннер для сайта, адаптировав его для дисплеев с разными коэффициентами плотности пикселей.

Запишем разметку:

```
1 <div class="clinic-banner">
2   
5 </div>
```

НА ОБЫЧНЫХ ЭКРАНАХ

На экранах со стандартным DPR все хорошо:



«МЫЛО» НА RETINA-ДИСПЛЕЯХ



Изображение размыто. Так получилось потому, что коэффициент DPR экрана равен 2.

РАЗНЫЕ КАРТИНКИ ДЛЯ РАЗНЫХ DPR

У тега `img` существует атрибут `srcset`, позволяющий указать набор путей до файлов. В зависимости от коэффицента плотности пикселей будут показываться разные картинки.

ДЕСКРИПТОР ДЛЯ КАЖДОГО ФАЙЛА

Пути до файлов разделяются запятыми. Для каждого изображения указывается особый дескриптор вида `nх`, где `n` – число, определяющее DPR. Дескриптор отделяется пробелом от пути к соответствующему ему изображению. Таким образом, мы можем указать:

```
1 
```

ВЫБОР ФАЙЛА

При такой записи атрибут `srcset` будет использован для определения коэффициента плотности пикселей устройства, и в зависимости от этого:

- при коэффициенте, равном 1, в качестве `src` изображения будет использовано `images/sample.jpg`;
- при коэффициенте, равном 2, в качестве `src` изображения будет использовано `images/sample-2x.jpg`;
- при коэффициенте, равном 3 – `images/sample-3x.jpg`.

ФОЛЛБЕК ДЛЯ СТАРЫХ БРАУЗЕРОВ

Но даже если у `img` есть `srcset` с набором изображений на все случаи жизни, атрибут `src` все равно нужен – для того, чтобы старые браузеры, не поддерживающие атрибут `srcset`, могли использовать его как фоллбэк и пользователь все равно увидел бы изображение.

РАСШИРИМ КОД

Добавим изображению баннера из нашей задачи атрибут `srcset`, в котором пропишем пути к исходной картинке и к картинке повышенного качества:

```
1 <div class="clinic-banner">
2   
7 </div>
```

ВСЕ ЧЕТКО

Теперь, если коэффициент плотности пикселей экрана будет кратен двум, браузер выведет изображение `banner_df_01_1080_2x.jpg`.

Изображение повышенного качества на retina-дисплее:

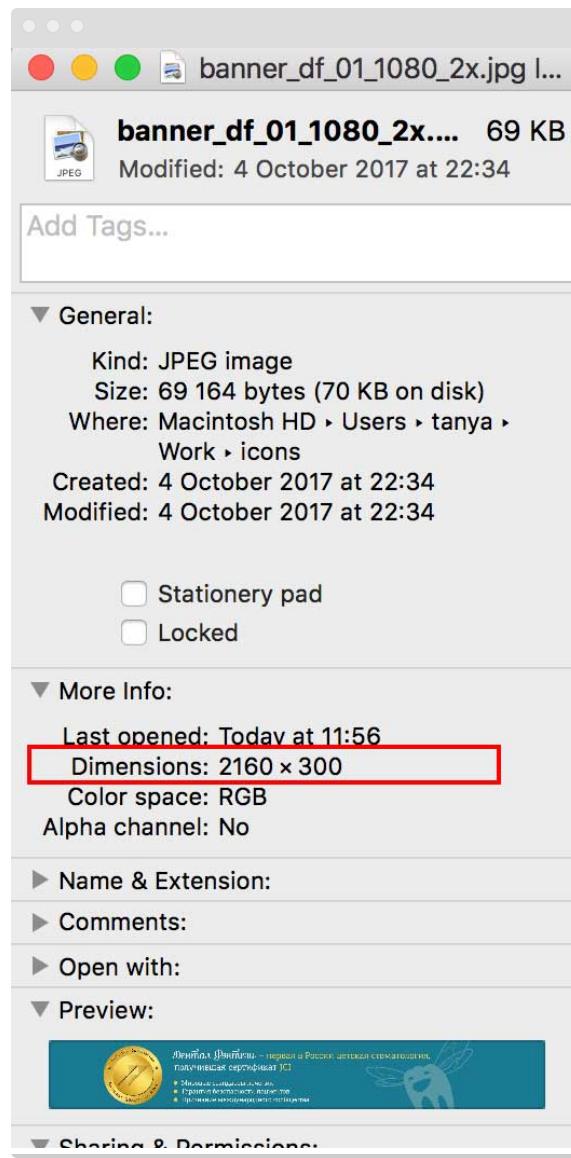


Когда мы просматриваем баннер на дисплее с DPR 2, браузер будет использовать в качестве `src` тот путь, рядом с которым указан дескриптор `2x`.

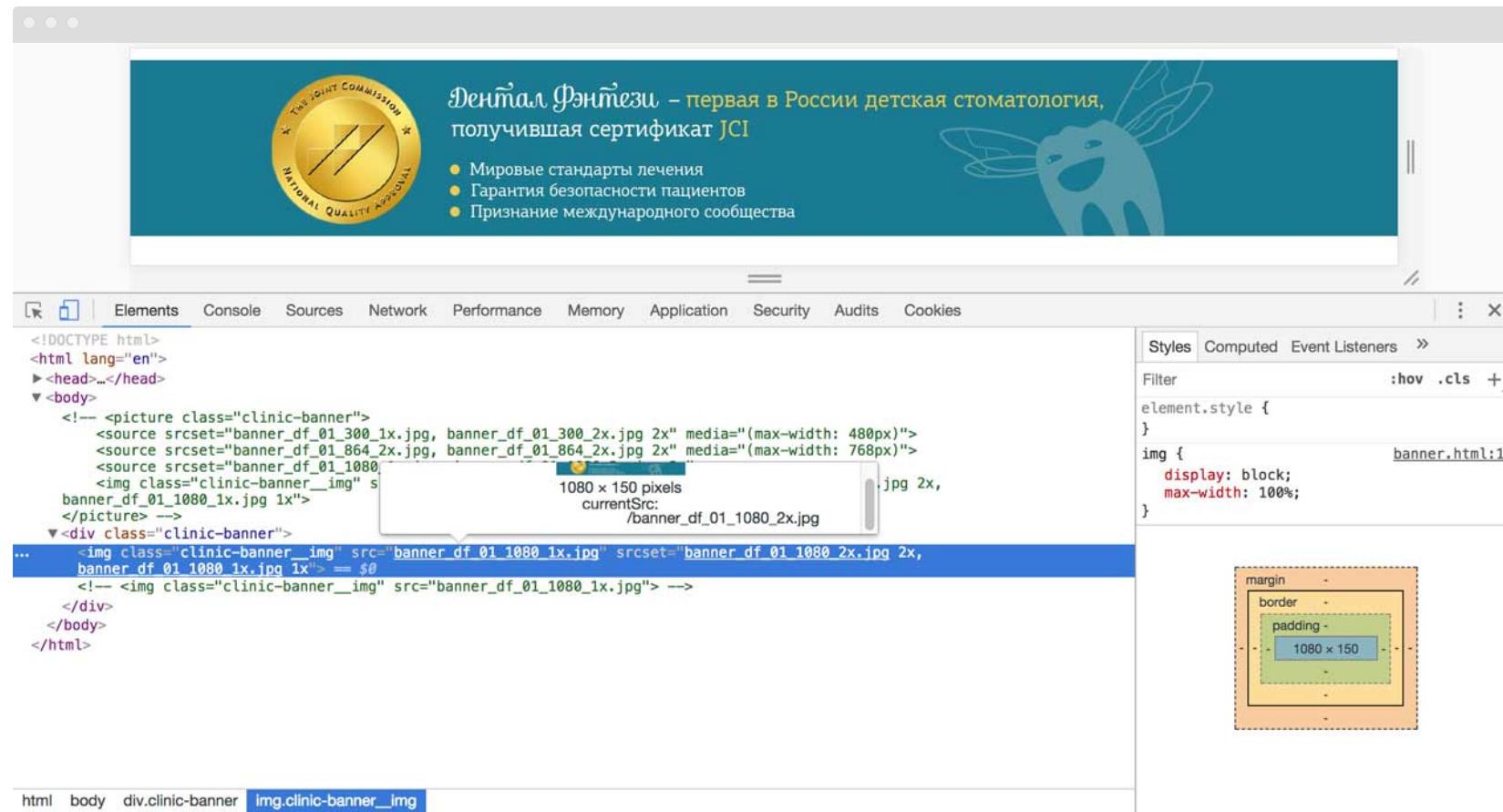
РАЗМЕРЫ ДЕЛЯТСЯ

Важно учесть, что при этом браузер разделит исходную ширину и высоту изображения на величину дескриптора. Несмотря на то, что размер изображения 2160px*300px, в инспекторе мы увидим, что «физический» размер составил 1080px*150px.

ИСХОДНЫЕ РАЗМЕРЫ



РЕАЛЬНЫЕ РАЗМЕРЫ



Дентал Фэнтези – первая в России детская стоматология, получившая сертификат JCI

- Мировые стандарты лечения
- Гарантия безопасности пациентов
- Признание международного сообщества

The Joint Commission
NATIONAL QUALITY APPROVED

1080 x 150 pixels
currentSrc: /banner_df_01_1080_2x.jpg

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <!-- <picture class="clinic-banner">
      <source srcset="banner_df_01_300_1x.jpg, banner_df_01_300_2x.jpg 2x" media="(max-width: 480px)">
      <source srcset="banner_df_01_864_2x.jpg, banner_df_01_864_2x.jpg 2x" media="(max-width: 768px)">
      <source srcset="banner_df_01_1080_1x.jpg 1x">
      <img class="clinic-banner__img" s=...>
    </picture> -->
    <div class="clinic-banner">
       = $0
      <!--  -->
    </div>
  </body>
</html>
```

Styles Computed Event Listeners >
Filter :hov .cls +
element.style {
}
img {
 display: block;
 max-width: 100%;
}

margin -
border -
padding -
1080 x 150

ВЗАИМОДЕЙСТВИЕ С ДИЗАЙНЕРОМ

Очень важно при приемке адаптивных макетов проверить, подготовил ли дизайнер изображения для каждого коэффициента плотности пикселей, и если нет – запросить эти изображения.

ИТОГИ

ВЫБОР *breakpoints*

- **Breakpoint** – условие, при котором раскладка сайта меняется с одной на другую. Характерно для медиавыражений. Чаще всего брейкпоинты выбираются в соответствии с размерами экранов популярных устройств – 1024px, 768px, 480px и другими.
- Вместо того, чтобы выбирать какие-то конкретные точки и пользоваться всегда только ими, имеет смысл исходить из особенностей дизайн-макета в каждом конкретном случае.

MOBILE FIRST

- **Mobile first** – концепция, согласно которой предлагается сначала создавать дизайн для устройств с маленькими экранами – мобильных телефонов, и затем усложнять его для устройств с большими экранами, таких как ноутбуки и настольные компьютеры.
- Такой подход означает отказ от привычной концепции создания дизайна для устройств с большими экранами и дальнейшим упрощением дизайна для мобильных устройств.

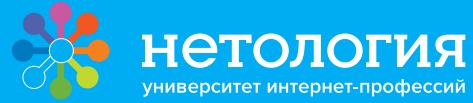
ЕДИНИЦЫ CSS, ЗАВИСИМЫЕ ОТ РАЗМЕРА ОКНА БРАУЗЕРА (VIEWPORT)

- В css есть единицы, позволяющие задавать зависимость от размеров окна браузера или **viewport**.
- **1vh** равен 1% от высоты окна браузера, **1vw** – 1% от ширины окна браузера. **Важно:** за ширину окна браузера принимается ширина без полосы прокрутки.
- **1vmin** равен 1% от меньшего из двух значений ширины или высоты окна браузера. Если устройство находится в вертикальной (портретной) ориентации, то меньшей из сторон окна будет ширина, в этом случае $1vmin = 1vw$. В горизонтальной (ландшафтной) ориентации меньшей будет высота окна, и, соответственно, в этом случае $1vmin = 1vh$.
- **1vmax** равен 1% от большего из двух значений ширины или высоты окна браузера. При вертикальной ориентации устройства $1vmax = 1vh$, при горизонтальной – $1vmax = 1vw$.

АДАПТИВНЫЕ ИЗОБРАЖЕНИЯ С ПОМОЩЬЮ srcset

- У тега `img` существует атрибут `srcset`, позволяющий указать набор путей до файлов. В зависимости от коэффицента плотности пикселей будут показываться разные картинки.
- Пути до файлов разделяются запятыми. Для каждого изображения указывается особый дескриптор вида `nx`, где `n` – число, определяющее DPR. Дескриптор отделяется пробелом от пути к соответствующему ему изображению.

```
1 
```



Задавайте вопросы и напишите отзыв о лекции!

СЕМЕН БОЙКО



simonderus@gmail.com



fb.me/simonboycko