

АДАПТИВНЫЕ ИЗОБРАЖЕНИЯ



МИХАИЛ ЛАРЧЕНКО / SYTAC



МИХАИЛ ЛАРЧЕНКО

Tech Lead в Sytac



larchanka@me.com



[m_larchanka](https://t.me/m_larchanka)

ПЛАН ЗАНЯТИЯ

1. Flexbox: растягивание и сжатие
2. flex-basis и размеры элемента
3. order
4. Адаптивные изображения: тег picture
5. calc() с относительными единицами измерения
6. Итоги

FLEXBOX: РАСТЯГИВАНИЕ И СЖАТИЕ

ФОРМА ПОИСКА

Важным условием задачи является то, что подсказка в левой части должна занимать две строчки при любой ширине блока.

Не можете что-то найти?
Введите запрос

Например, инфракрасный детектор

НАЙТИ

A search form with a blue header bar. On the left, there is a link "Не можете что-то найти?" and a placeholder "Введите запрос". In the center is a search input field with the placeholder "Например, инфракрасный детектор". On the right is a yellow button labeled "НАЙТИ".

РАЗМЕТКА ДЛЯ ПОЛЯ ПОИСКА

```
1 <form class="search">
2   <label class="search__hint" for="search-field">
3     <span class="search__question">
4       Не можете что-то найти?
5     </span>
6     <span class="search__action">
7       Введите запрос
8     </span>
9   </label>
10  <input id="search-field" type="search"
11    class="search__field"
12    placeholder="Например, инфракрасный детектор"
13    name="query" required>
14  <button class="search__button">Найти</button>
15 </form>
```

СТИЛИ ФОРМЫ

```
1 .search {  
2     display: flex;  
3     align-items: center;  
4 }  
5  
6 .search__hint {  
7     margin-right: 1%;  
8     width: 12%;  
9 }  
10  
11 .search__field {  
12     width: 80%;  
13 }  
14  
15 .search__button {  
16     min-width: 7%;  
17 }
```

ПРОВЕРЯЕМ НА КОМПЬЮТЕРЕ

На экране монитора наша верстка выглядит, как мы хотели:



ПРОВЕРЯЕМ НА ПЛАНШЕТЕ

Но если мы откроем верстку с планшета в горизонтальной ориентации, то увидим следующее:

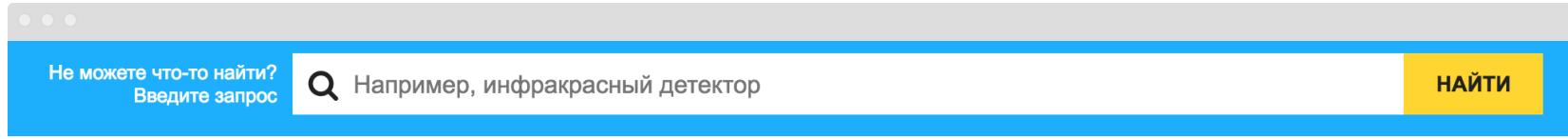


Блок с подсказкой стал слишком узким, и подсказка теперь занимает три строки. На такой ширине экрана по макету необходимо больше места под подсказку.

БОЛЬШЕ МЕСТА ДЛЯ ПОДСКАЗКИ

```
1 .search__hint {  
2   margin-right: 1%;  
3 }  
4 @media screen and (min-width: 1025px) {  
5   .search__hint {  
6     width: 12%;  
7   }  
8   .search__field {  
9     width: 80%;  
10 }  
11  .search__button {  
12    min-width: 7%;  
13 }  
14 }  
15 @media screen and (max-width: 1024px) {  
16   .search__hint {  
17     width: 16%;  
18   }  
19   .search__field {  
20     width: 74%;  
21   }  
22   .search__button {  
23     min-width: 9%;  
24   }  
25 }
```

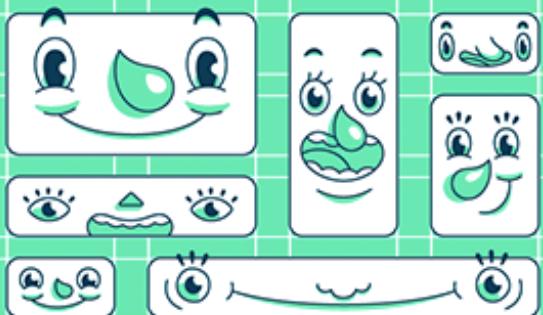
ПОВТОРНАЯ ПРОВЕРКА НА ПЛАНШЕТЕ



Но решение не оптимально, поскольку дважды указаны размеры для нескольких блоков.



Свойство ***flex-shrink*** задает коэффициент гибкого растягивания, который определяет, как будет сужаться элемент относительно остальных flex-элементов во flex-контейнере при распределении отрицательного свободного пространства.



СИНТАКСИС

Свойство `flex-shrink` имеет синтаксис:

```
flex-shrink: <номер>;
```

где `<номер>` – это целое число больше или равное 0.

Значение по умолчанию – `flex-shrink: 1`. При этом flex-элемент может сужаться.

Если установить значение `flex-shrink: 0`, то такой элемент будет всегда сохранять первоначальный размер.

ЗАПРЕЩАЕМ СУЖЕНИЕ

Удалим код с размерами блоков, заданными в процентах, и укажем блоку подсказки и кнопке значения, не позволяющие этим элементам менять свои размеры:

```
1  .search__hint,  
2  .search__button {  
3    flex-shrink: 0;  
4 }
```

ПРОМЕЖУТОЧНЫЙ РЕЗУЛЬТАТ

Чего-то явно не хватает: поле поиска не растягивается, поскольку мы убрали размеры.



flex-grow

Чтобы задать полю ввода возможность растягиваться, используем свойство `flex-grow`. Данное свойство позволяет управлять алгоритмом растягивания flex-блоков при расширении flex-контейнера.



СИНТАКСИС

Свойство имеет вид:

```
flex-grow: <номер>;
```

где `<номер>` это целое число больше или равно 0.

Значение по умолчанию: `flex-grow: 0`. Запрещает элементу растягиваться, когда контейнер увеличивается в размерах.

Если всем flex-элементам задано одно и то же значение `flex-grow`, все элементы в контейнере будут одинакового размера. А если все flex-элементы будут иметь `flex-grow: 1`, а один – `flex-grow: 2`, то элемент с значением 2 будет увеличиваться кратно двум.

РАСТЯНEM ПОЛЕ ПОИСКА

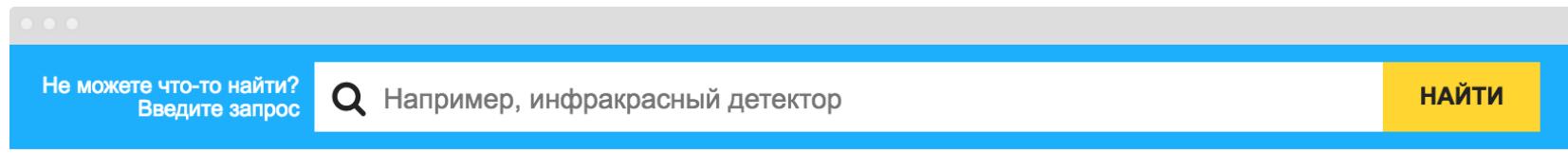
```
1 .search__hint,  
2 .search__button {  
3   flex-shrink: 0;  
4   flex-grow: 0;  
5 }  
6  
7 .search__field {  
8   flex-grow: 1;  
9 }
```

ПРЕКРАСНЫЙ РЕЗУЛЬТАТ

На мониторе:



На планшете:



[Live Demo](#)

flex-basis

Это свойство определяет размеры элемента и может принимать одно из значений:

- значение ширины элемента в любых единицах;
- значение `auto`, что позволяет элементу иметь базовую ширину относительно контента внутри него. Это значение по умолчанию.

ШОРТКАТ `flex`

Все три свойства – `flex-grow`, `flex-shrink` и `flex-basis` – могут быть записаны с помощью шортката `flex`:

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

Рассмотрим возможные значения свойства `flex` подробнее.

flex: auto

Данное значение эквивалентно `flex: 1 1 auto`.

Такой элемент становится абсолютно гибким, он будет расширяться в зависимости от размера контейнера, забирая оставшееся место.

flex: 0 auto

Значение эквивалентно `flex: 0 1 auto` и `flex: initial`.

Такой элемент получит размеры исходя из указанных свойств `width` и `height` или же исходя из размеров контента, если `width` и `height` не заданы. Элемент не тянется при расширении контейнера, но может сжиматься.

flex: <номер>

Эквивалентно `flex: <номер> 1 0px`.

Делает элемент гибким, позволяя расширяться пропорционально указанному в виде числа коэффициенту.

flex: none

Данное значение эквивалентно `flex: 0 0 auto`.

Как мы помним, значение `0` для `flex-grow` и `flex-shrink` запрещают элементу сжиматься и растягиваться. Таким образом, `flex: none` позволяет сделать flex-элемент полностью «негибким».

ДЕЛАЕМ КОД КОРОЧЕ

В нашем примере это как раз необходимо для блоков кнопки и подсказки. Мы уже задали этим блокам `flex-shrink: 0` и `flex-grow: 0`, а `flex-basis` по умолчанию имеет значение `auto`, так что мы можем переписать код короче:

```
1 .search__hint,  
2 .search__button {  
3   flex: none;  
4 }  
5  
6 .search__field {  
7   flex-grow: 1;  
8 }
```

flex-basis и РАЗМЕРЫ ЭЛЕМЕНТА

ЧЕТЫРЕ БЛОКА ФИКСИРОВАННОЙ ШИРИНЫ

Рассмотрим пример из четырех блоков в flex-контейнере фиксированной ширины:

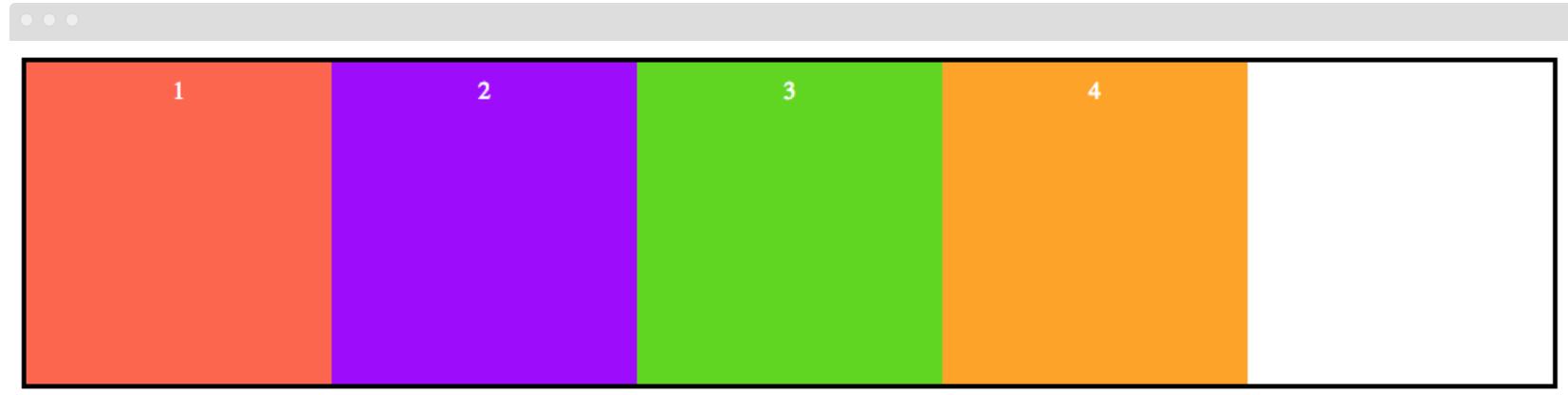
```
1 .container {  
2   display: flex;  
3   width: 1000px;  
4 }
```

Зададим каждому блоку значения `width` и `height`:

```
1 .block {  
2   width: 200px;  
3   height: 200px;  
4 }
```

ОЖИДАЕМЫЙ РЕЗУЛЬТАТ

Получилось четыре flex-элемента фиксированной ширины:



ДОБАВИМ `flex-basis`

Свойство `flex-basis` может принимать значение `auto` (и это – значение по умолчанию) или значение ширины в любых единицах. Не меняя значения `width` и `height`, добавим блокам `flex-basis` со значением, отличным от `width`:

```
1 .block {  
2   flex-basis: 250px;  
3   width: 200px;  
4   height: 200px;  
5 }
```

flex-basis РЕШАЕТ

Видим, что ширина блоков стала такой, как указано для `flex-basis`.

Так как главная ось идет сейчас слева направо, `flex-basis` задает ширину – если flex-элементу указан `flex-basis`, ширина блока будет определяться только его значением.



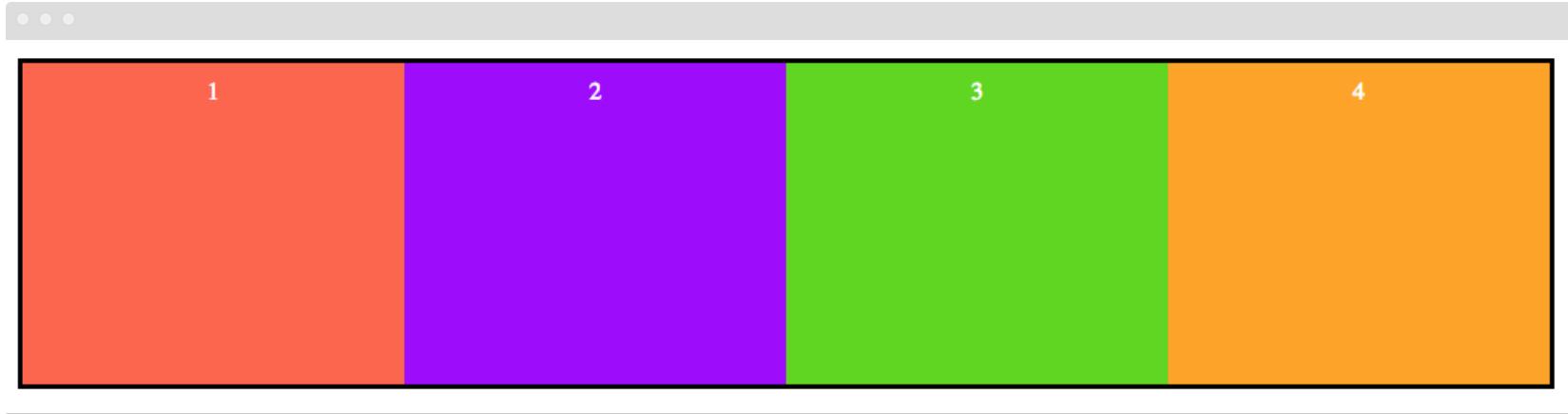
ШИРИНА НЕ ИГРАЕТ РОЛИ

А вот значение свойства `width` в данном случае оказалось проигнорировано.

Это легко проверить – давайте вообще удалим это свойство:

```
1 .block {  
2   flex-basis: 250px;  
3   height: 200px;  
4 }
```

НИЧЕГО НЕ ИЗМЕНИЛОСЬ

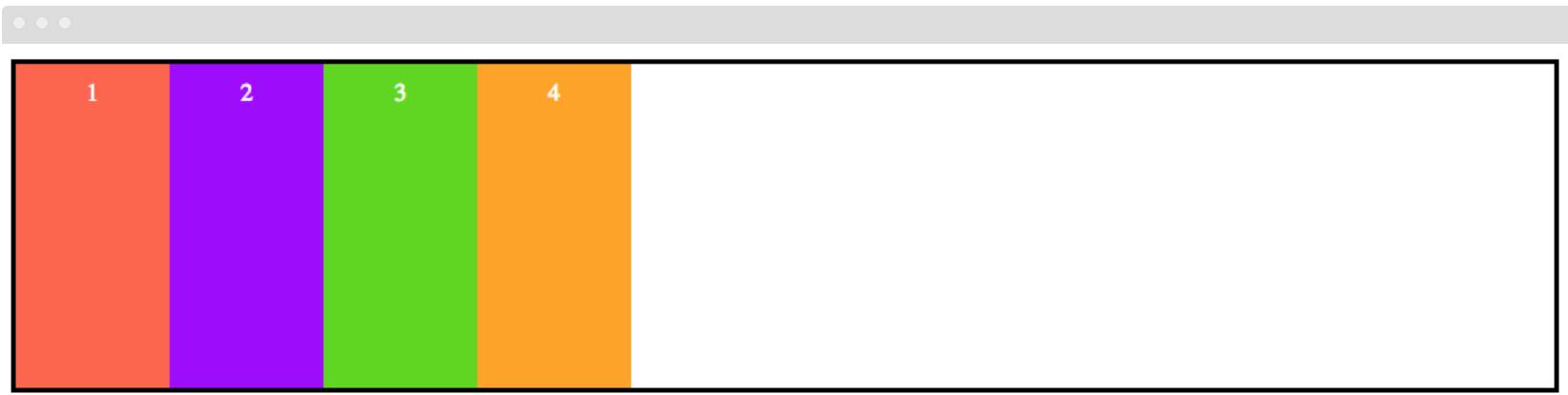


УКАЖЕМ max-width

```
1 .block {  
2   flex-basis: 250px;  
3   max-width: 100px;  
4   height: 200px;  
5 }
```

max-width ВЫИГРЫВАЕТ

Несмотря на то, что `flex-basis` равен `250px`, ширина каждого блока станет `100px`:



ЗАДАДИМ min-width

А теперь посмотрим, как будет работать `flex-basis` с `min-width`.

Уменьшим значение `flex-basis` до `100px` и укажем `min-width`:

```
1 .block {  
2     flex-basis: 100px;  
3     min-width: 250px;  
4     height: 200px;  
5 }
```

МИНИМАЛЬНАЯ ШИРИНА БЛОКОВ

В этом случае ширина блоков будет равна `250px`, потому что даже `flex-basis: 100px` не может запретить блокам иметь ширину менее определенной свойством `min-width`:



ВЫВОДЫ

Таким образом, можно сделать вывод, что свойство `width` для flex-элементов имеет смысл только лишь в случае, когда `flex-basis` не указан (то есть, имеет значение по умолчанию – `auto`).

flex-basis – НЕ ФИКСИРОВАННАЯ ШИРИНА

При этом `flex-basis` не может задать фиксированный размер блока. В наших примерах суммарная ширина блоков была меньше ширины контейнера или же равна ширине контейнера. Но в ситуации, когда блокам не хватит места (например, 6 блоков при `flex-basis: 250px` и ширине контейнера `1000px`) – по умолчанию блоки будут сжиматься, чтобы вместиться в контейнер:



РАВНОЦЕННО ДЛЯ `height`

Все, что мы рассмотрели выше, также относится и к свойству `height` в случае, когда изменена основная ось направления flex-элементов с помощью значений свойства `flex-direction: column` или `column-reverse`.

ЭКСПЕРИМЕНТ С БЛОКАМИ В КОЛОНКУ

Укажем контейнеру следующие свойства:

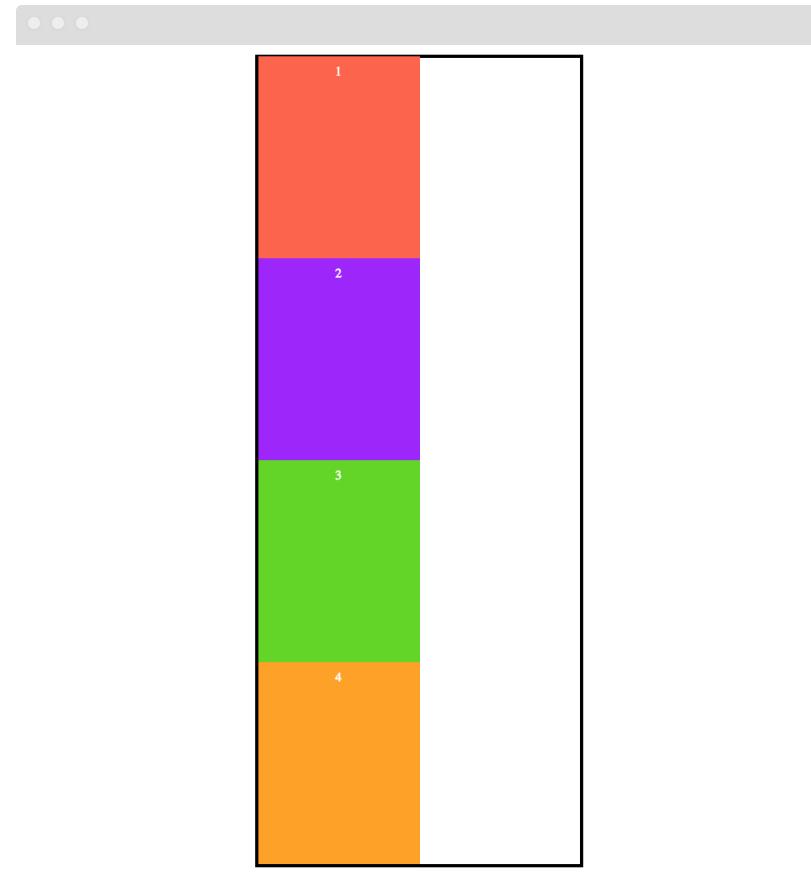
```
1 .container {  
2   display: flex;  
3   height: 1000px;  
4   flex-direction: column;  
5 }
```

А блокам зададим:

```
1 .block {  
2   width: 200px;  
3   height: 200px;  
4   flex-basis: 250px;  
5 }
```

ГЛАВНАЯ ОСЬ ИЗМЕНИЛАСЬ

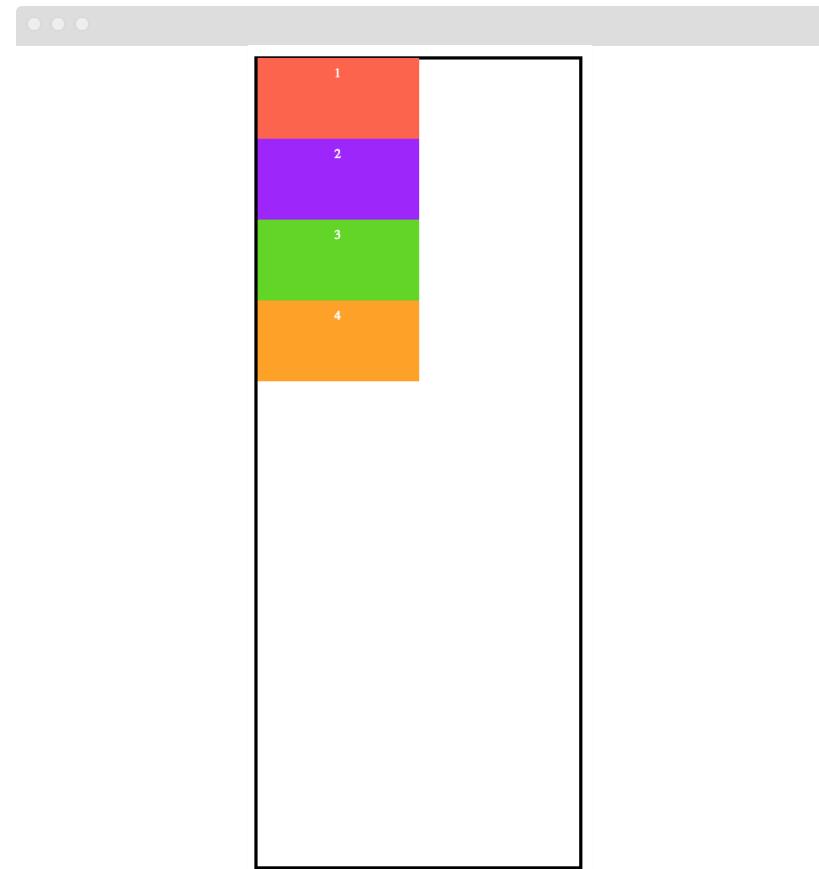
Мы видим, что блоки имеют `200px` ширины и `250px` высоты. Это произошло потому, что мы поменяли направление главной оси, и теперь именно `flex-basis` задает высоту блоков, даже при наличии свойства `height`.



МАКСИМАЛЬНАЯ ВЫСОТА

Укажем `max-height`:

```
1 .block {  
2   flex-basis: 250px;  
3   max-height: 100px;  
4   width: 200px;  
5 }
```

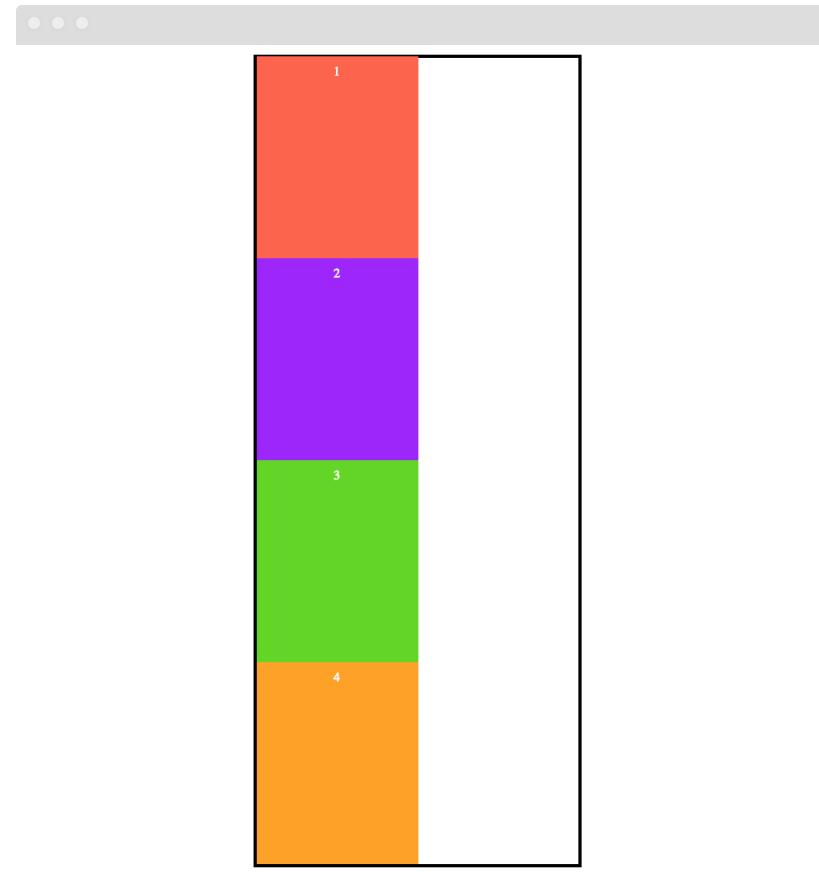


МИНИМАЛЬНАЯ ВЫСОТА

И если мы укажем `min-height`:

```
1 .block {  
2   flex-basis: 100px;  
3   min-height: 250px;  
4   width: 200px;  
5 }
```

[Live Demo](#)



order

СВОЙСТВО order

Сверстаем адаптивный блок карточки статьи. По дизайну блок с тегами в десктопной версии должен располагаться между картинкой и контентом:



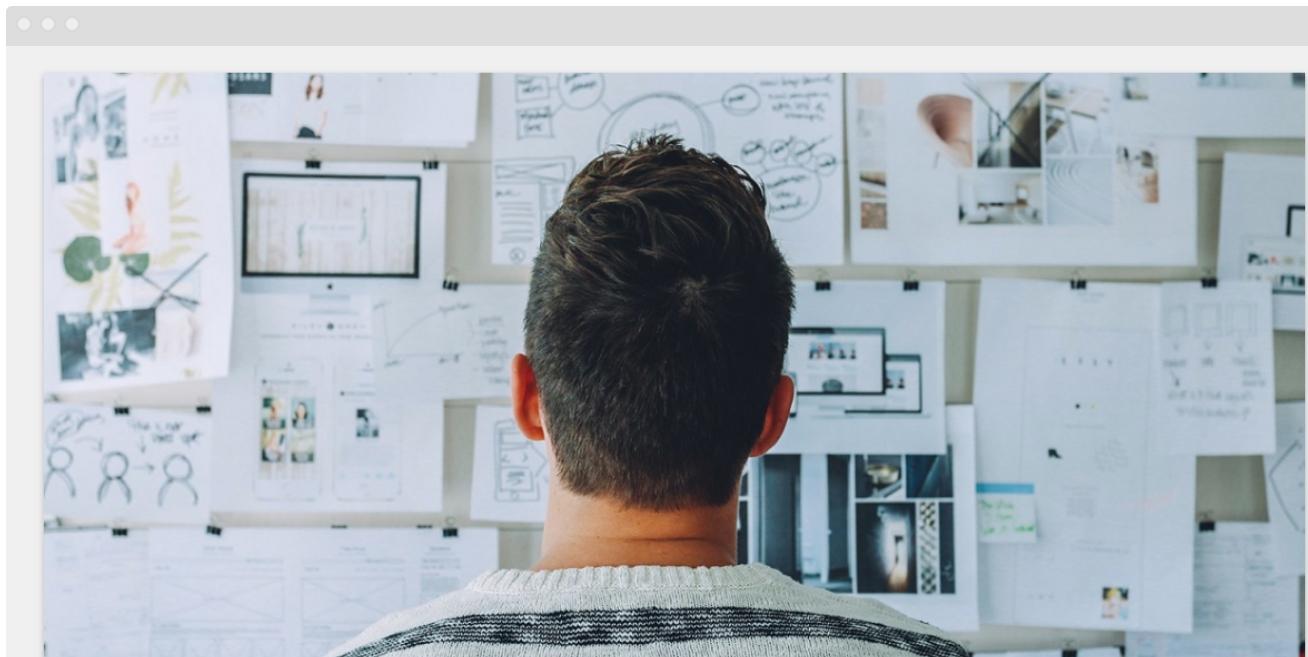
Library-of-plugins is a collection of ready-made decisions for front-end development

html css javascript

They will help you to faster your development, to add functionality and to decorate the interface. It is ok for beginners and professionals. I hope this collection will help you. If you have any questions you can connect with me to social networks.

КАРТОЧКА СТАТЬИ НА МОБИЛЬНОМ

А на мобильных устройствах – после контента:



Library-of-plugins is a collection of ready-made decisions for front-end development

They will help you to faster your development, to add functionality and to decorate the interface. It is ok for beginners and professionals. I hope this collection will help you. If you have any questions you can connect with me to social networks.

html css javascript

РАЗМЕТКА КАРТОЧКИ

```
1 <article class="article">
2   
3   <div class="article__body">
4     <h3 class="article__title">
5       Library-of-plugins is a collection of ready-made decisions
6       for front-end development
7     </h3>
8     <div class="article__tags">
9       <span class="article__tag">html</span>
10      <span class="article__tag">css</span>
11      <span class="article__tag">javascript</span>
12    </div>
13    <div class="article__content">
14      ...
15    </div>
16  </div>
17 </article>
```

ДОБАВИМ СТИЛИ ДЛЯ ДЕСКТОПА

Для нужного расположения блоков на десктопе запишем:

```
1 .article__body {  
2   display: flex;  
3   flex-direction: column;  
4 }  
5  
6 .article__tags {  
7   display: flex;  
8   flex-wrap: wrap;  
9   align-items: center;  
10 }
```

COOTBETCTBYET MAKETY



The image shows a man from behind, looking at a wall covered in various design prototypes and wireframes. The wall is filled with hand-drawn sketches, digital mockups, and printed documents, suggesting a creative workspace or a design studio.

Library-of-plugins is a collection of ready-made decisions for front-end development

[html](#) [css](#) [javascript](#)

They will help you to faster your development, to add functionality and to decorate the interface. It is ok for beginners and professionals. I hope this collection will help you. If you have any questions you can connect with me to social networks.

СИНТАКСИС `order`

Поменять местами блоки для мобильной версии нам поможет свойство `order`.

Свойство имеет вид:

`order: <номер>`

где `<номер>` – это любое целое число, указывающее, каким по порядку должен выводиться блок в потоке элементов.

По умолчанию flex-элементы выводятся в том порядке, в котором они записаны в разметке, и значение `order` каждого из них равно `0`.

Чем меньше значение `order` flex-элемента, тем раньше в потоке будет он стоять, независимо от фактического места в HTML.

ПОМЕНЯЕМ ПОРЯДОК НА МОБИЛЬНОМ

```
1 @media screen and (max-width: 767px) {  
2     .article__tags {  
3         order: 1;  
4     }  
5 }
```

Мы задали `order` только для блока `.article__tags`, а `order` для `.article__content` и `.article__title` по-прежнему равен `0`, поэтому блок тегов будет выводиться последним на мобильной версии.

ВСЕ ПО ДИЗАЙНУ

Блок статьи на десктопе:



Library-of-plugins is a collection of ready-made decisions for front-end development

html css javascript

They will help you to faster your development, to add functionality and to decorate the interface. It is ok for beginners and professionals. I hope this collection will help you. If you have any questions you can connect with me to social networks.

Блок статьи на мобильных:



Library-of-plugins is a collection of ready-made decisions for front-end development

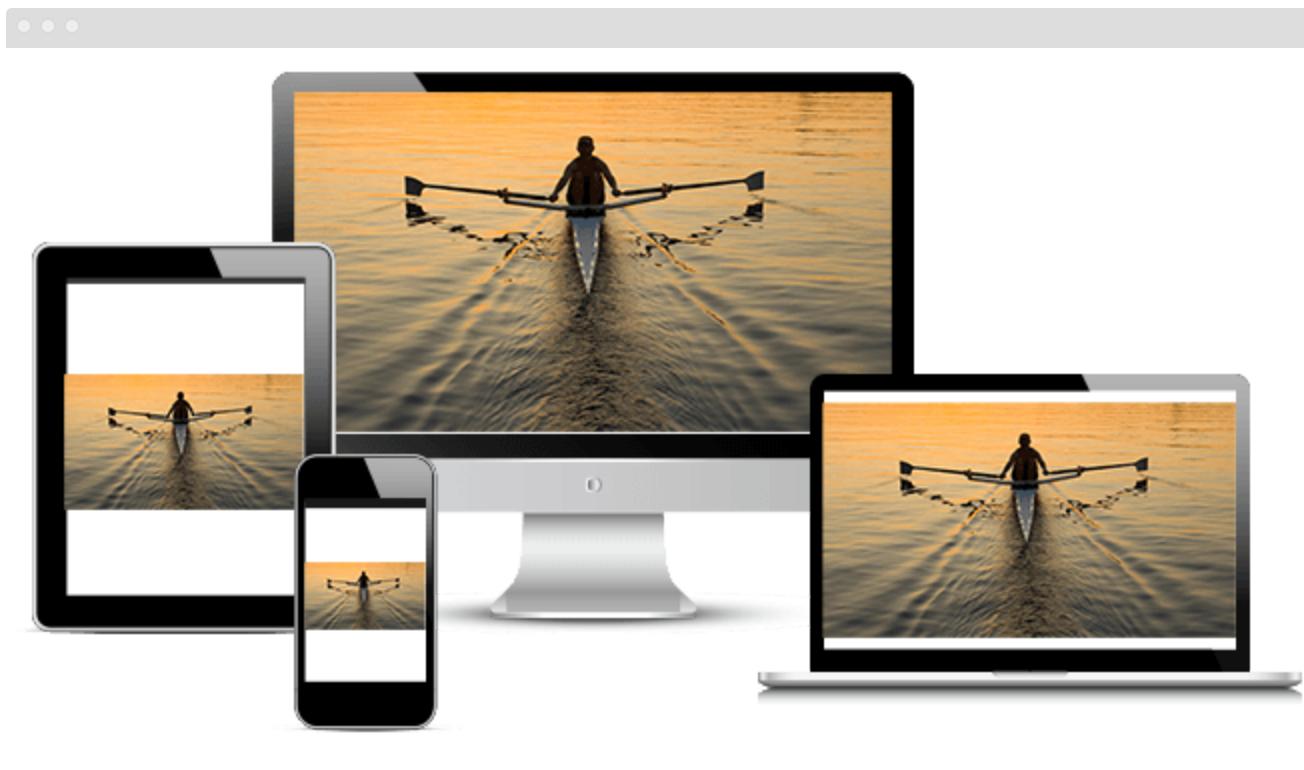
html css javascript

They will help you to faster your development, to add functionality and to decorate the interface. It is ok for beginners and professionals. I hope this collection will help you. If you have any questions you can connect with me to social networks.

TER picture

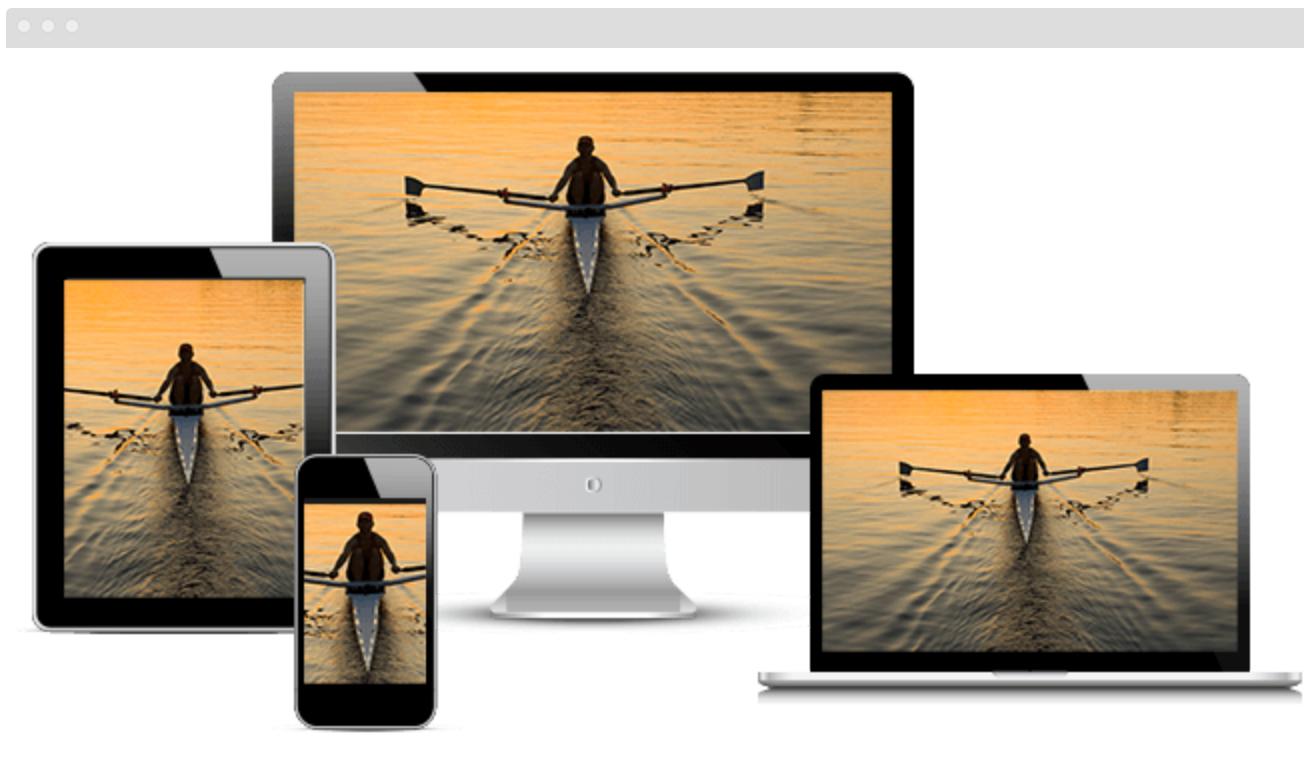
ОДНА КАРТИНКА

Сравните одно изображение, растянутое или сжатое:



НАБОР КАРТИНОК

И несколько изображений разного размера:



АДАПТИВНЫЙ БАННЕР

Рассмотрим задачу адаптивного баннера. Запишем HTML:

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

ДОПИШЕМ СТИЛИ

```
1 .clinic-banner__img {  
2   max-width: 100%;  
3 }
```

БАННЕР НА УЗКИХ ЭКРАНАХ

Баннер на экране 768px:



Баннер на экране 320px:



СТРУКТУРА ТЕГА `picture`

```
1 <picture>
2   <source
3     media="(min-width: 768px)"
4     srcset="images/image-medium.jpg">
5   <source
6     media="(min-width: 465px)"
7     srcset="images/image-small.jpg">
8   
11 </picture>
```

У тега `picture` атрибутов нет – он служит контейнером для тегов `source` и `img`.

ВЛОЖЕННЫЙ ТЕГ `source`

Атрибут `srcset` тега `source` заполняется по тем же правилам, что и у тега `img`. В атрибуте `media` указается условие, по которому браузер выбирает, какую картинку из набора `source` показывать.

ОБЯЗАТЕЛЬНЫЙ `img`

После всех тегов `source` следует указать тег `img`, который будет использован в качестве фоллбэка в старых браузерах, которые будут игнорировать теги `picture` и `source`.

ВЗАИМОДЕЙСТВИЕ С ДИЗАЙНЕРОМ

Важно при приемке адаптивных макетов проверить, подготовил ли дизайнер изображения для разных устройств и экранов с разной плотностью пикселей.

РАЗНЫЕ ИЗОБРАЖЕНИЯ

Для диапазона 480px-768px «обрезан» лишний фон по бокам баннера:



Для экранов мобильных телефонов принципиально другая картинка:

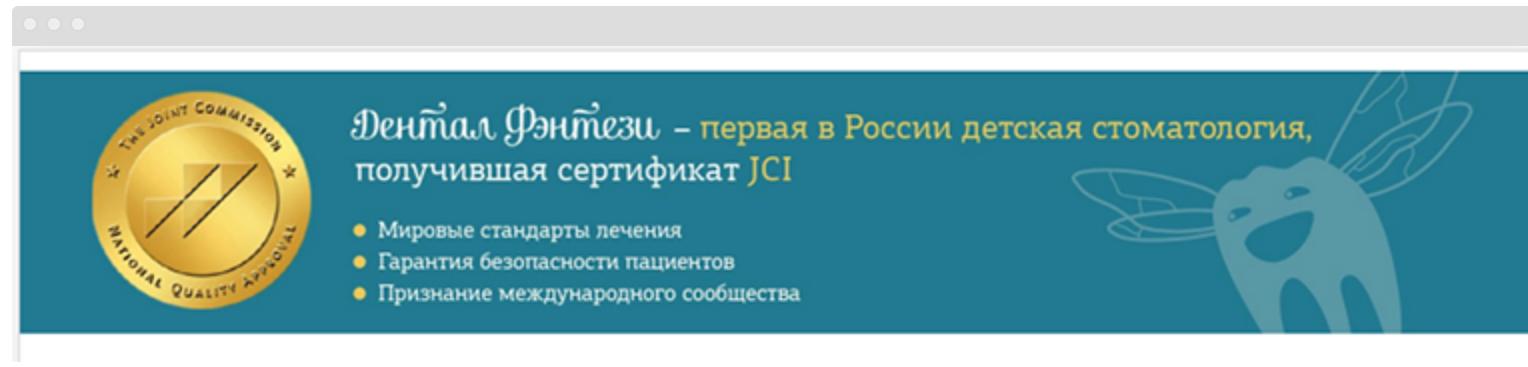


УКАЗЫВАЕМ БРАУЗЕРУ, ЧТО ПОКАЗЫВАТЬ

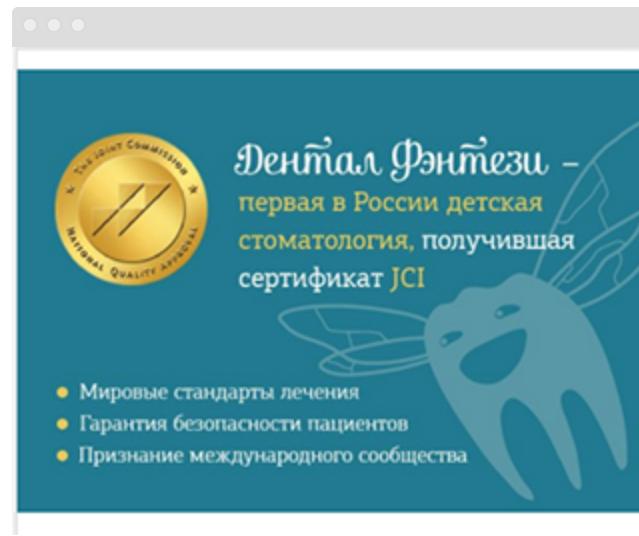
```
1 <picture class="clinic-banner">
2   <source srcset="banner_df_01_300_1x.jpg,
3     banner_df_01_300_2x.jpg 2x" media="(max-width: 480px)">
4
5   <source srcset="banner_df_01_864_1x.jpg,
6     banner_df_01_864_2x.jpg 2x" media="(max-width: 768px)">
7
8   <source srcset="banner_df_01_1080_1x.jpg,
9     banner_df_01_1080_2x.jpg 2x">
10
11  
13 </picture>
```

РЕЗУЛЬТАТ НА РАЗНЫХ РАЗРЕШЕНИЯХ

Баннер на экране 768px:



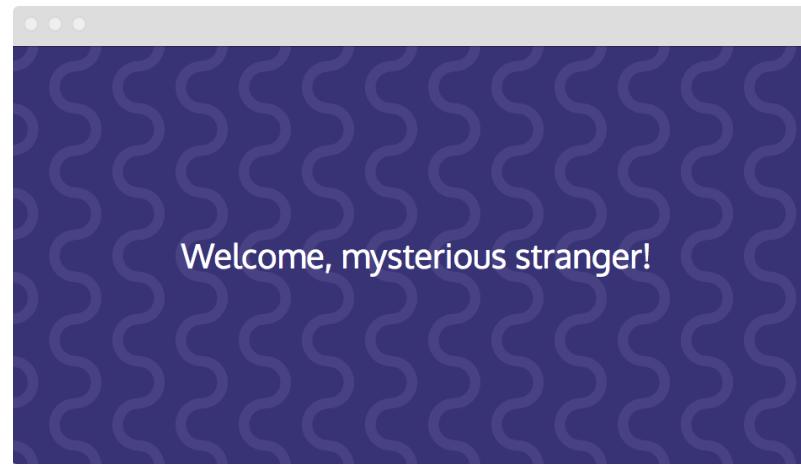
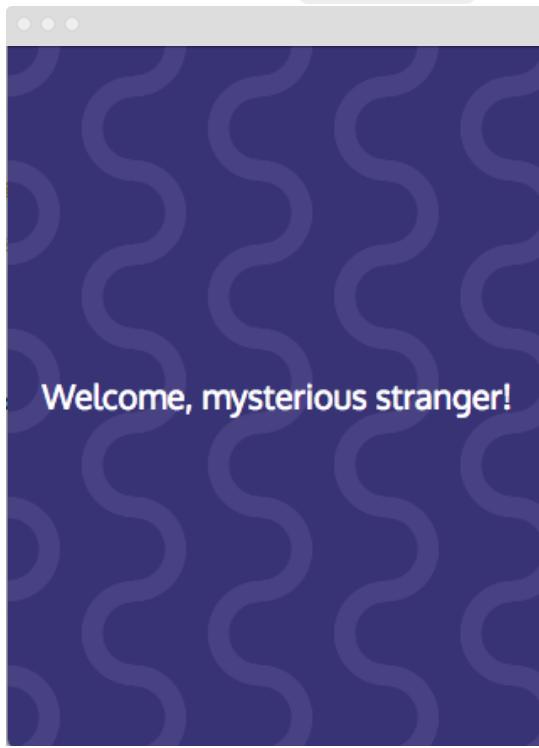
Баннер на экране 320px:



`calc()` с
относительными
единицами
измерения

ЭКРАН ПРИВЕТСТВИЯ

Сверстаем макет приветственного экрана с уточнением от дизайнера:
нужно, чтобы размер шрифта изменялся плавно на интервале от 320px
до 960px , меняя значение от 20px (при ширине 320px) до 40px
(при ширине 960px):



ИСПОЛЬЗУЕМ vw

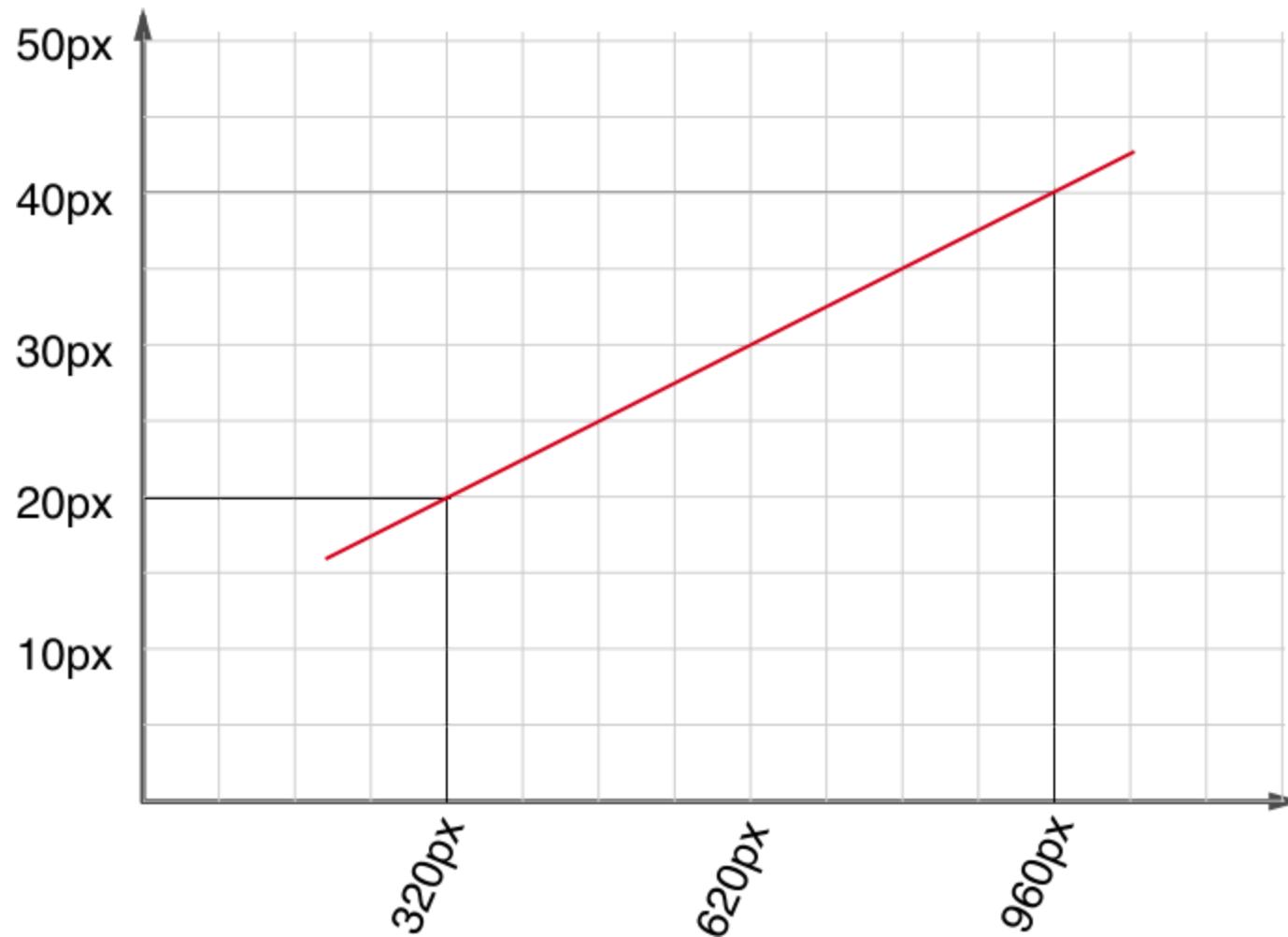
При ширине окна 320px font-size должен быть 20px. 1vw при такой ширине равен $320\text{px}/100 = 3.2\text{px}$, тогда $20\text{px}/3.2\text{px} = 6.25\text{vw}$:

```
1 .greeting__message {  
2   font-size: 6.25vw;  
3 }
```

БОЛЬШЕ, ЧЕМ НУЖНО

Одна проблема: когда ширина окна станет равна `960px`, `1vw` будет равен $960\text{px}/100 = 9.6\text{px}$ и `font-size` станет $9.6\text{px} \times 6.25 = 60\text{px}$, а нам нужно `40px`.

ГРАФИК ПРЯМОЙ



УРАВНЕНИЕ ПРЯМОЙ

$$y = ax + b$$

В нашем случае `y` – это и будет `font-size`, `x` – это `1vw`, а коэффициенты `a` и `b` мы можем найти.

ПОДСТАВИМ ИЗВЕСТНЫЕ ЗНАЧЕНИЯ

Нам известны две точки: при ширине 320px (и соответственно, $1\text{vw} = 3.2\text{px}$) $\text{font-size} = 20\text{px}$, а при ширине 960px ($1\text{vw} = 9.6\text{px}$) $\text{font-size} = 40\text{px}$:

$$20\text{px} = a \cdot 3.2\text{px} + b \quad // \quad 1$$

$$40\text{px} = a \cdot 9.6\text{px} + b \quad // \quad 2$$

ВЫРАЗИМ b

Выразим b из первого уравнения:

$$20px - a*3.2px = b$$

и подставим во второе:

$$40px = a*9.6px + (20px - a*3.2px)$$

ИЩЕМ а

У нас получилось уравнение с одним неизвестным, и теперь мы можем найти а :

$$40\text{px} - 20\text{px} = a * 9.6\text{px} - a * 3.2\text{px}$$

$$20\text{px} = a * 6.4\text{px}$$

$$a = 20\text{px} / 6.4\text{px} = 3.125$$

ИЩЕМ **b**

А теперь подставим получившееся значение в уравнение и найдем **b**:

$$20\text{px} - 3.125 \cdot 3.2\text{px} = b$$

$$b = 20\text{px} - 10\text{px} = 10\text{px}$$

ПОДСТАВИМ ЗНАЧЕНИЯ В ФОРМУЛУ

$$y = 3.125x + 10px$$

ПИШЕМ CSS

А чтобы браузер в каждой точке мог вычислить сумму `3.125vw + 10px`, нам понадобится функция `calc()`:

```
1 .greeting__message {  
2   font-size: calc( 3.125vw + 10px );  
3 }
```

ПРОЦЕСС ВЫЧИСЛЕНИЯ

1. При открытии страницы в браузере, первым делом браузер будет вычислять 1vw в пикселях. Для ширины 900px браузер рассчитает $1\text{vw} = 9\text{px}$.
2. Затем браузер умножит $1\text{vw} * 3.125$, то есть $9\text{px} * 3.125 = 28.125\text{px}$.
3. Потом прибавит 10px . $28.125\text{px} + 10\text{px} = 38.125\text{px}$.

ОКРУГЛЕНИЕ ПИКСЕЛЕЙ

Дробные значения в пикселях используются только для объяснения процесса вычисления браузером размеров. В реальности пиксель представляет из себя физическую «лампочку». Невозможно зажечь 3.5 лампочки или 6.125 лампочки.

По этой причине браузер всегда округляет дробные значения пикселей. Причем в разных браузерах округление происходит по разному: в большую или меньшую сторону.

ПИШЕМ МЕДИАВЫРАЖЕНИЯ

```
1 @media (min-width: 320px) {  
2     .greeting__message {  
3         font-size: calc( 3.125vw + 10px );  
4     }  
5 }  
6  
7 @media (min-width: 960px) {  
8     .greeting__message {  
9         font-size: 40px;  
10    }  
11 }
```

ПЕРЕПИШЕМ НА REM'АХ

```
1 @media (min-width: 320px) {  
2     .greeting__message {  
3         font-size: calc( 3.125vw + 0.625rem );  
4     }  
5 }  
6  
7 @media (min-width: 960px) {  
8     .greeting__message {  
9         font-size: 2.5rem;  
10    }  
11 }
```

[Live Demo](#)

ИТОГИ

FLEXBOX: РАСТЯГИВАНИЕ И СЖАТИЕ

- Свойство `flex-shrink` задает коэффициент гибкого растягивания, определяющий, как будет сужаться элемент относительно остальных flex-элементов во flex-контейнере при распределении отрицательного свободного пространства.
- При значении `flex-shrink: 1` (по умолчанию) flex-элемент может сужаться. При значении `flex-shrink: 0` элемент будет всегда сохранять первоначальный размер.
- Свойство `flex-grow` позволяет управлять алгоритмом растягивания flex-блоков при расширении flex-контейнера.
- Значение `flex-grow: 0` (по умолчанию) запрещает элементу растягиваться при увеличении контейнера. Если всем flex-элементам в контейнере задано одинаковое значение `flex-grow`, то они будут одинакового размера.

СВОЙСТВО `flex-basis`

- Свойство `flex-basis` определяет размеры элемента и может принимать значения: `auto` (по умолчанию) либо значение ширины элемента в любых единицах.
- Три свойства – `flex-grow`, `flex-shrink` и `flex-basis` – могут быть записаны с помощью шортката `flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]`.
- Свойство `width` для flex-элементов имеет смысл, только если `flex-basis` не указан (то есть, имеет значение по умолчанию – `auto`).
- Это относится и к свойству `height`, если изменена основная ось направления flex-элементов (`flex-direction: column` или `column-reverse`)
- `flex-basis` не задает фиксированный размер блока.

order

- Свойство `order` позволяет поменять элементы местами.
- Синтаксис: `order: <номер>`, где `<номер>` – любое целое число, указывающее, каким по порядку будет выводиться блок в потоке элементов.
- Значение `order` по умолчанию – 0.
- Чем меньше значение `order` flex-элемента, тем раньше в потоке будет он стоять, независимо от фактического места в HTML.

АДАПТИВНЫЕ ИЗОБРАЖЕНИЯ: ТЕГ picture

- Тег `picture` служит контейнером для тегов `source` и `img` ; собственных атрибутов этот тег не имеет.
- Атрибут `srcset` тега `source` заполняется по тем же правилам, что и у тега `img` .
- В атрибуте `media` указается условие, по которому браузер выбирает, какую картинку из набора `source` показывать.
- После всех тегов `source` обязательно нужно указать тег `img` , который будет использован как фоллбэк в старых браузерах, которые проигнорируют теги `picture` и `source` .

calc() С ОТНОСИТЕЛЬНЫМИ ЕДИНИЦАМИ ИЗМЕРЕНИЯ

- Для плавного изменения размера шрифта при изменении ширины экрана можно вычислять значение размера шрифта с помощью `calc()`.
- В таком случае размер шрифта рассчитывается с использованием формулы $y = ax + b$, где y – `font-size`, x – `1vw`, а коэффициенты a и b зависят от того, какой размер шрифта необходим при какой ширине экрана.
- При вычислении размеров браузер всегда округляет дробные значения пикселей. В разных браузерах округление происходит по разному: в большую или меньшую сторону



Задавайте вопросы и напишите отзыв о лекции!

МИХАИЛ ЛАРЧЕНКО



larchanka@me.com



[m_larchanka](https://t.me/m_larchanka)